

Tutorial Menjalankan Notebook Python di Google Colab



CO

KASMUI

TUTORIAL MENJALANKAN NOTEBOOK PYTHON DI GOOGLE COLAB

Dari Dasar Pemrograman hingga Optimasi Geometri
dan Visualisasi Molekul Tiga Dimensi

```
from pyscf import gto, dft  
mol = gto.M(atom=..., basis="def2-tzvp")  
mf = dft.RKS(mol).density_fit()  
mf.xc = "b3lyp"  
energy = mf.kernel()
```

KETERANGAN PENERBITAN

Komponen	Keterangan
Judul	Tutorial Menjalankan Notebook Python di Google Colab
Subjudul	Dari Dasar Pemrograman hingga Optimasi Geometri dan Visualisasi Molekul Tiga Dimensi
Penulis	KASMUI
Penerbit	Jaten Press, Semarang
Edisi	Cetakan I, Tahun 2026
Ukuran	A4
Format isi	Times New Roman 12 pt, spasi 1,15; paragraf justify tanpa inden
Jenis buku	Buku ajar, panduan teknis, dan referensi praktis
Cakupan	5 bab dan 8 lampiran utama
Sasaran	Mahasiswa, dosen, guru kimia, peneliti pemula, dan pengguna Python

Hak cipta dilindungi undang-undang. Kutipan ilmiah yang wajar harus mencantumkan sumber. Naskah ini disusun sebagai bahan pengembangan akademik dan tetap memerlukan penyuntingan substansi, telaah sejawat, verifikasi setiap keluaran komputasi, serta pemeriksaan akhir sebelum penerbitan resmi.

Catatan pemutakhiran: Informasi runtime, ketersediaan GPU, dan paket biner dapat berubah. Buku menekankan deteksi lingkungan dan pencatatan versi agar notebook tetap transparan dan mudah diperbarui.

KATA PENGANTAR

Bismillāhirrahmānirrahīm

Alhamdulillahillāhi rabbil ‘ālamīn. Segala puji bagi Allah Swt. atas limpahan rahmat, pertolongan, dan kemudahan-Nya sehingga buku berjudul *Tutorial Menjalankan Notebook Python di Google Colab* ini dapat disusun dan diselesaikan. Selawat dan salam semoga senantiasa tercurah kepada Nabi Muhammad saw., keluarga, para sahabat, serta seluruh umat beliau yang istiqamah mengikuti petunjuknya.

Perkembangan teknologi komputasi telah membuka kesempatan yang semakin luas bagi mahasiswa, dosen, guru, peneliti, dan masyarakat umum untuk mempelajari pemrograman serta melakukan komputasi ilmiah tanpa harus memiliki perangkat dengan spesifikasi tinggi. Salah satu sarana yang dapat dimanfaatkan adalah Google Colab, yaitu lingkungan notebook berbasis awan yang memungkinkan pengguna menulis, menjalankan, mendokumentasikan, dan membagikan kode Python langsung melalui peramban. Buku ini disusun untuk membantu pembaca memahami penggunaan Google Colab secara bertahap, mulai dari konsep paling dasar hingga penerapannya dalam pengolahan data dan kimia komputasi.

Pembahasan dalam buku ini diawali dengan pengenalan ekosistem Python, Jupyter Notebook, dan Google Colab. Pembaca diperkenalkan pada antarmuka notebook, sel kode, sel Markdown, runtime, kernel, pengelolaan berkas, serta cara mengunggah, mengunduh, dan menyimpan hasil pekerjaan. Dasar-dasar Python, seperti variabel, tipe data, operator, struktur kontrol, fungsi, modul, dan penanganan kesalahan, disajikan sebagai fondasi sebelum pembaca melanjutkan ke penggunaan NumPy, Pandas, dan Matplotlib untuk komputasi numerik, pengelolaan data tabular, serta visualisasi hasil.

Buku ini juga menempatkan penyusunan notebook ilmiah yang sistematis dan reproduktif sebagai bagian penting dalam proses pembelajaran. Notebook tidak hanya dipandang sebagai tempat menyalin dan menjalankan kode, tetapi sebagai dokumen ilmiah yang harus memuat tujuan, teori singkat, parameter, input, proses, validasi, keluaran, interpretasi, serta catatan keterbatasan. Pembaca diarahkan untuk menjalankan notebook dari runtime yang bersih, mencatat versi Python dan pustaka, memeriksa lingkungan komputasi, menggunakan urutan eksekusi yang benar, serta melakukan **Restart session** dan **Run all** untuk memastikan bahwa hasil tidak bergantung pada variabel tersembunyi atau state kernel lama.

Pada bagian selanjutnya, pembaca dibimbing membangun dan menampilkan struktur molekul tiga dimensi. Pembahasan mencakup representasi molekul menggunakan SMILES, SDF, MOL, dan XYZ; pemeriksaan muatan, jumlah elektron, multiplicity, dan spin; pembuatan konformer dengan RDKit; optimasi awal menggunakan MMFF94 atau UFF; serta visualisasi molekul menggunakan py3Dmol. Dengan pendekatan tersebut, pembaca tidak hanya memperoleh struktur visual, tetapi juga belajar memeriksa sumber struktur, konsistensi koordinat, dan kesesuaian data kimia sebelum melanjutkan ke perhitungan yang lebih tinggi.

Bagian kimia komputasi membahas optimasi geometri dan analisis sifat molekul menggunakan PySCF dan geomeTRIC. Materi meliputi konsep energi potensial, metode Hartree–Fock dan Density Functional Theory, basis set, parameter numerik, konvergensi

SCF, analisis geometri, frekuensi, orbital molekul, HOMO, LUMO, celah energi, dan momen dipol. Contoh perhitungan disusun secara bertingkat, mulai dari molekul sederhana seperti H_2 dan H_2O hingga proyek lengkap optimasi dan analisis parasetamol menggunakan metode B3LYP/def2-TZVP.

Buku ini juga memperkenalkan pemanfaatan GPU melalui GPU4PySCF. Pembaca diarahkan untuk tidak sekadar mengaktifkan GPU, tetapi juga memeriksa ketersediaan perangkat, versi CUDA, kompatibilitas pustaka, serta kemungkinan penggunaan mode otomatis CPU–GPU. Apabila backend GPU tidak tersedia atau tidak kompatibel, buku menganjurkan penggunaan mekanisme *fallback* ke CPU dengan tetap mencatat perangkat, versi, waktu eksekusi, dan status pengujian. Pendekatan ini penting karena konfigurasi runtime, ketersediaan GPU, dan paket biner dalam Google Colab dapat berubah dari waktu ke waktu.

Salah satu penekanan utama buku ini adalah validasi. Setiap hasil komputasi harus diperiksa dari sisi format, sintaks, input, satuan, muatan, spin, koordinat, konvergensi, kestabilan numerik, serta kemampuan keluaran untuk disimpan dan dibuka kembali. Nilai energi, geometri, orbital, atau sifat molekul tidak seharusnya langsung ditafsirkan hanya karena kode telah selesai dijalankan. Status konvergensi, keberadaan nilai NaN atau tak hingga, konsistensi jumlah atom, pilihan metode, basis set, serta keterbatasan model harus menjadi bagian dari pertimbangan ilmiah.

Untuk membantu pembaca belajar secara mandiri, buku ini dilengkapi dengan berbagai lampiran praktis. Lampiran tersebut mencakup perintah instalasi dan pemulihan runtime, daftar kesalahan umum beserta pemecahannya, format keluaran, notebook lengkap yang siap dipindahkan ke Google Colab, resep-resep Python, latihan dan proyek, rubrik penilaian, skenario troubleshooting, serta kamus perintah dan pola kode penting. Materi lampiran disusun agar pembaca dapat menggunakan buku secara linear dari bab pertama sampai terakhir ataupun secara modular sesuai kebutuhan.

Buku ini ditujukan terutama bagi mahasiswa, dosen, guru kimia, peneliti pemula, dan pengguna Python yang ingin memanfaatkan Google Colab untuk pembelajaran, pengolahan data, komputasi ilmiah, dan kimia komputasi. Pembaca pemula dapat mengikuti tahapan pembelajaran secara berurutan, sedangkan pembaca yang telah memiliki pengalaman dapat langsung menuju bagian instalasi, validasi, troubleshooting, akselerasi GPU, atau proyek akhir.

Penulis menyadari bahwa buku ini masih memiliki keterbatasan. Perangkat lunak, pustaka Python, versi runtime, dukungan GPU, dan kompatibilitas paket terus berkembang. Oleh karena itu, pembaca dianjurkan selalu memeriksa dokumentasi resmi, mencatat versi lingkungan yang digunakan, dan menguji ulang setiap notebook sebelum digunakan dalam pembelajaran, penelitian, atau publikasi. Hasil komputasi juga perlu dibandingkan dengan teori, data eksperimen, literatur ilmiah, atau perangkat lunak lain apabila digunakan untuk menarik kesimpulan akademik yang lebih luas.

Ucapan terima kasih penulis sampaikan kepada semua pihak yang telah memberikan dukungan, inspirasi, kritik, dan saran dalam proses penyusunan buku ini. Penghargaan juga disampaikan kepada para pengembang Python, Project Jupyter, Google Colab, NumPy, Pandas, Matplotlib, RDKit, py3Dmol, PySCF, geomeTRIC, GPU4PySCF, serta

berbagai perangkat lunak sumber terbuka lainnya yang telah memberikan kontribusi besar terhadap perkembangan pembelajaran dan penelitian berbasis komputasi.

Kritik dan saran yang konstruktif sangat diharapkan demi penyempurnaan edisi berikutnya. Semoga buku ini dapat memberikan manfaat, memudahkan pembaca dalam mempelajari notebook Python, serta mendorong lahirnya praktik komputasi ilmiah yang lebih sistematis, kritis, transparan, dan dapat direproduksi.

Semoga Allah Swt. meridai setiap ikhtiar dalam pengembangan ilmu pengetahuan dan menjadikan buku ini sebagai bagian dari amal yang bermanfaat.

Semarang, Juli 2026

Penulis,


KASMUI

PETUNJUK MENGGUNAKAN BUKU

Buku ini dapat dipakai secara linear dari Bab 1 sampai Bab 5 atau secara modular sesuai kebutuhan. Pembaca pemula dianjurkan menyelesaikan setiap halaman praktik, sedangkan pembaca berpengalaman dapat langsung menuju kode, validasi, troubleshooting, dan proyek akhir.

1. Buka notebook baru di Google Colab dan beri nama sesuai pola yang disarankan.
2. Jalankan sel instalasi satu kali pada runtime baru, kemudian catat versi lingkungan.
3. Salin kode secara utuh; jangan menggabungkan baris yang seharusnya terpisah.
4. Lakukan Restart session dan Run all untuk menguji reproduksibilitas.
5. Baca status konvergensi dan validasi sebelum menafsirkan angka.
6. Simpan hasil di Google Drive atau unduh arsip ZIP sebelum runtime berakhir.

Elemen	Fungsi
Ikon/Kotak Biru	Prinsip, catatan, atau informasi penting.
Kotak Hijau	Kriteria keberhasilan dan validasi.
Kotak Jingga	Kesalahan umum dan peringatan.
Blok Monospace	Kode yang dapat ditempatkan dalam sel Colab.
Latihan	Modifikasi kode, analisis, dan refleksi.
Etika eksekusi notebook: Jalankan hanya notebook dari sumber yang dipercaya. Kode notebook dapat membaca, menulis, atau menghapus berkas yang dapat diakses runtime.	

DAFTAR ISI

KETERANGAN PENERBITAN	2
KATA PENGANTAR	3
PETUNJUK MENGGUNAKAN BUKU	6
BAB 1 - MENGENAL GOOGLE COLAB DAN NOTEBOOK PYTHON	21
1.1 Ekosistem Python, Jupyter, dan Google Colab	24
1.1.1 Python sebagai bahasa komputasi ilmiah	25
1.1.2 Konsep Jupyter Notebook.....	30
1.1.3 Google Colab sebagai notebook berbasis awan.....	35
1.2 Membuat dan Mengelola Notebook di Google Colab	41
1.2.1 Membuat, membuka, mengganti nama, dan menyimpan notebook	42
1.2.2 Menjalankan sel dan mengelola runtime	47
1.2.3 Mengunggah dan mengunduh berkas	52
1.3 Dasar Python untuk Komputasi Ilmiah.....	58
1.3.1 Variabel, tipe data, operator, dan struktur kontrol.....	59
1.3.2 Fungsi, modul, dan penanganan kesalahan.....	64
1.3.3 NumPy, Pandas, dan Matplotlib	70
BAB 2 - MENYUSUN NOTEBOOK ILMIAH YANG REPRODUKTIF	78
2.1 Struktur Notebook yang Sistematis	81
2.1.1 Judul, identitas, tujuan, dan teori singkat	82
2.1.2 Konfigurasi parameter dan fungsi modular	87
2.1.3 Urutan eksekusi dan kebersihan notebook	92
2.2 Instalasi Pustaka dan Pemeriksaan Lingkungan	97
2.2.1 Instalasi dengan %pip	98
2.2.2 Pemeriksaan Python, sistem, RAM, dan versi pustaka	103
2.2.3 Deteksi CPU, GPU, CUDA, dan kompatibilitas	109
2.3 Input, Output, dan Pengelolaan Berkas	114
2.3.1 Membaca dan menulis TXT, CSV, JSON, dan XYZ	115
2.3.2 Google Drive dan penyimpanan hasil	120
2.3.3 Membuat laporan dan arsip ZIP	125
BAB 3 - MEMBANGUN DAN MENAMPILKAN STRUKTUR MOLEKUL TIGA DIMENSI.....	133

3.1 Representasi dan Validasi Molekul	136
3.1.1 Representasi SMILES, SDF, MOL, dan XYZ	137
3.1.2 Muatan, jumlah elektron, multiplicity, dan spin	142
3.1.3 Sumber struktur molekul	147
3.2 Pembuatan Konformer dengan RDKit	153
3.2.1 Mengubah SMILES menjadi molekul tiga dimensi	154
3.2.2 ETKDGv3, MMFF94, dan UFF	159
3.2.3 Validasi dan ekspor koordinat	165
3.3 Visualisasi Molekul dengan py3Dmol	170
3.3.1 Menampilkan struktur awal	171
3.3.2 Memberikan label dan informasi geometri	176
3.3.3 Membandingkan struktur awal dan akhir	182
BAB 4 - OPTIMASI GEOMETRI DAN ANALISIS SIFAT MOLEKUL DENGAN PYSCF	189
4.1 Dasar Teori Perhitungan Struktur Elektronik	192
4.1.1 Energi potensial dan optimasi geometri	193
4.1.2 Metode HF dan DFT	198
4.1.3 Basis set dan parameter numerik	203
4.2 Notebook Lengkap Optimasi Molekul H ₂ O	208
4.2.1 Instalasi, struktur awal, dan energi awal	209
4.2.2 Optimasi B3LYP/def2-TZVP	215
4.2.3 Frekuensi, geometri, dan visualisasi hasil	220
4.3 Analisis Keluaran Komputasi	225
4.3.1 Energi dan konvergensi	227
4.3.2 HOMO, LUMO, celah energi, dan momen dipol	232
4.3.3 Interpretasi kritis dan keterbatasan	237
BAB 5 - GPU4PYSCF, VALIDASI, DAN PROYEK LENGKAP PARASETAMOL	245
5.1 Akselerasi GPU di Google Colab	248
5.1.1 Mengaktifkan dan mendeteksi GPU	249
5.1.2 Instalasi GPU4PySCF yang kompatibel	254
5.1.3 Perbandingan CPU, GPU, dan mode otomatis	259
5.2 Validasi Notebook dan Reprodusibilitas	265
5.2.1 Validasi format dan sintaks	266
5.2.2 Validasi kimia dan numerik	271

5.2.3 Dokumentasi versi dan pelaporan pengujian.....	277
5.3 Proyek Akhir: Optimasi dan Analisis Parasetamol	282
5.3.1 Pembuatan struktur awal parasetamol	283
5.3.2 Optimasi DFT B3LYP/def2-TZVP	289
5.3.3 Analisis, visualisasi, dan ekspor seluruh hasil.....	294
DAFTAR PUSTAKA.....	302
LAMPIRAN A - DAFTAR PERINTAH INSTALASI DAN PEMULIHAN RUNTIME	304
LAMPIRAN B - DAFTAR KESALAHAN UMUM DAN PEMECAHANNYA	313
LAMPIRAN C - FORMAT HASIL: XYZ, CSV, JSON, TXT, MOLDEN, DAN ZIP.....	328
LAMPIRAN D - LIMA NOTEBOOK LENGKAP SIAP GOOGLE COLAB.....	335
LAMPIRAN E - 60 RESEP NOTEBOOK PYTHON SIAP GOOGLE COLAB.....	359
E.1 Python Dasar dan Lingkungan Runtime	359
E.01 Mendeteksi Versi Python dan Sistem Runtime.....	359
E.02 Mengelola Variabel dan Satuan Secara Eksplisit.....	360
E.03 Menyusun List, Tuple, dan Dictionary untuk Parameter	362
E.04 Membuat Percabangan untuk Validasi Input	363
E.05 Menggunakan Perulangan dengan Status Proses	364
E.06 Membuat Fungsi dengan Type Hint dan Docstring	366
E.07 Menangani Kesalahan dengan try-except yang Informatif	367
E.08 Membaca dan Menulis Teks UTF-8.....	368
E.09 Menyimpan Konfigurasi sebagai JSON.....	369
E.10 Mengelola Jalur Berkas dengan pathlib	371
E.11 Membuat Log Eksekusi Sederhana	372
E.12 Mengukur Waktu Eksekusi Secara Konsisten	373
E.2 NumPy dan Komputasi Numerik	375
E.13 Membuat Array dan Memeriksa Shape.....	375
E.14 Menghitung Jarak Antaratom dengan Vektor	376
E.15 Menghitung Sudut Ikatan	377
E.16 Menggunakan Masking untuk Menyaring Data.....	379
E.17 Melakukan Reshape dan Transpose	380
E.18 Memahami Broadcasting.....	381
E.19 Menggunakan Aljabar Linear Dasar	382
E.20 Melakukan Interpolasi Satu Dimensi	383

E.21 Menghitung Turunan Numerik	385
E.22 Menghitung Integral Numerik.....	386
E.23 Menggunakan Random Seed untuk Reprodusibilitas	387
E.24 Menyimpan dan Membaca Array.....	388
E.3 Pandas dan Pengelolaan Data.....	389
E.25 Membuat DataFrame dari Dictionary	389
E.26 Membaca CSV dengan Validasi Kolom	391
E.27 Membersihkan Nilai Hilang dan Duplikat	392
E.28 Menyaring dan Mengurutkan Data	393
E.29 Mengelompokkan Data dengan groupby	394
E.30 Menggabungkan Dua Tabel dengan merge.....	396
E.31 Membuat Pivot Table	397
E.32 Menghasilkan Statistik Deskriptif.....	398
E.33 Membuat Grafik dari DataFrame	399
E.34 Mengekspor Data ke CSV dan Excel.....	401
E.4 Visualisasi dan Pelaporan	402
E.35 Membuat Grafik Garis yang Terukur.....	402
E.36 Membuat Diagram Pencar dan Regresi Linear	403
E.37 Membuat Histogram dan Membaca Distribusi	404
E.38 Membuat Heatmap Matriks dengan Matplotlib	406
E.39 Menyusun Ringkasan Hasil dalam Markdown	407
E.40 Membuat Laporan HTML Sederhana	408
E.41 Membuat Arsip ZIP Hasil Proyek.....	409
E.42 Mengunduh Berkas dari Google Colab	411
E.43 Mengunggah Berkas ke Runtime	412
E.44 Menghubungkan Google Drive	413
E.45 Memeriksa Ruang Disk dan Memori	414
E.46 Merekam Versi Pustaka ke requirements.txt.....	416
E.47 Membuat Manifest Hasil dengan Checksum	417
E.48 Mendeteksi GPU dengan nvidia-smi.....	418
E.49 Membuat Fungsi Fallback CPU-GPU.....	419
E.5 Kimia Komputasi Dasar	421
E.50 Mengubah SMILES Menjadi Molekul RDKit.....	421
E.51 Membuat Konformer ETKDGv3	422

E.52 Mengoptimasi Awal dengan MMFF94 atau UFF	423
E.53 Mengekspor Koordinat RDKit ke XYZ	425
E.54 Menampilkan Molekul dengan py3Dmol	426
E.55 Menghitung Energi H2 dengan RHF	427
E.56 Memindai Kurva Energi H2	428
E.57 Menghitung H2O dengan DFT B3LYP	430
E.58 Mengoptimasi Geometri H2O dengan geomeTRIC.....	431
E.59 Menganalisis HOMO, LUMO, dan Celah Energi.....	432
E.60 Menghitung Momen Dipol.....	434
E.61 Menyimpan Ringkasan Perhitungan Kimia	435
LAMPIRAN F - 50 LATIHAN, PROYEK, DAN RUBRIK PENILAIAN	437
F.1 Python dan Notebook	437
F.01 Membandingkan state kernel sebelum dan sesudah Restart session	437
F.02 Membuat fungsi konversi satuan dengan unit test	438
F.03 Menyusun konfigurasi proyek dalam JSON.....	439
F.04 Membuat pencatat waktu untuk beberapa fungsi	440
F.05 Mendeteksi dan menjelaskan tiga jenis exception	441
F.06 Menyusun struktur folder proyek menggunakan pathlib.....	443
F.07 Membuat log eksekusi dengan tiga level pesan	444
F.08 Mengemas notebook dan artefak ke ZIP	445
F.09 Membuat manifest checksum hasil	446
F.10 Menguji reproduksibilitas dengan Run all	447
F.2 NumPy	448
F.11 Menghitung jarak semua pasangan atom	448
F.12 Menghitung sudut ikatan dari koordinat Cartesian	449
F.13 Mendeteksi NaN dan inf pada data energi	451
F.14 Membandingkan loop Python dan operasi vektor	452
F.15 Menyelesaikan sistem persamaan linear sederhana	453
F.16 Menghitung eigenvalue matriks simetris.....	454
F.17 Menginterpolasi kurva energi.....	455
F.18 Mengestimasi turunan numerik	456
F.19 Mengintegrasikan spektrum diskret.....	457
F.20 Menyimpan dan memuat array dalam NPY dan CSV.....	458
F.3 Pandas.....	460

F.21 Membersihkan tabel dengan nilai hilang.....	460
F.22 Menggabungkan tabel parameter dan hasil	461
F.23 Membuat pivot metode versus basis set	462
F.24 Mengelompokkan waktu eksekusi per backend.....	463
F.25 Mendeteksi duplikat job_id	464
F.26 Membuat statistik deskriptif dan outlier.....	465
F.27 Membuat laporan CSV dan Excel	466
F.28 Membuat grafik dari DataFrame	468
F.29 Menyusun tabel perbandingan energi relatif	469
F.30 Membuat pipeline baca-bersih-analisis-ekspor	470
F.4 Visualisasi	471
F.31 Membuat kurva energi dengan anotasi minimum	471
F.32 Membuat residual plot regresi	472
F.33 Membandingkan histogram dengan bin berbeda.....	473
F.34 Membuat heatmap matriks korelasi	474
F.35 Mengekspor grafik ke PNG dan PDF.....	475
F.5 Google Colab.....	477
F.36 Mengelola unggah dan unduh berkas	477
F.37 Menyimpan hasil pada Google Drive.....	478
F.38 Mendeteksi CPU, RAM, disk, dan GPU	479
F.39 Membuat mode auto CPU-GPU	480
F.40 Merekam versi runtime dan pustaka.....	481
F.6 Kimia Komputasi	482
F.41 Menghitung kurva energi H2	482
F.42 Membandingkan basis set H2.....	483
F.43 Menghitung energi H2O dengan HF dan DFT	484
F.44 Mengoptimasi geometri H2O	486
F.45 Menghitung panjang dan sudut ikatan.....	487
F.46 Menganalisis HOMO, LUMO, dan gap	488
F.47 Menghitung momen dipol	489
F.48 Membuat konformer etanol dengan RDKit.....	490
F.49 Memvisualisasikan struktur dengan py3Dmol	491
F.50 Menyusun proyek lengkap parasetamol	492
LAMPIRAN G - 40 SKENARIO TROUBLESHOOTING DAN PEMULIHAN	494

G.01 ModuleNotFoundError	494
G.02 ImportError setelah upgrade	495
G.03 Runtime terputus	496
G.04 Berkas di /content hilang	497
G.05 PermissionError pada Drive.....	497
G.06 FileNotFoundError.....	498
G.07 UnicodeDecodeError	499
G.08 CSV terbaca satu kolom	500
G.09 Kolom numerik menjadi object	501
G.10 DataFrame kosong setelah filter	502
G.11 SettingWithCopyWarning	503
G.12 ValueError shape mismatch.....	503
G.13 NaN atau inf muncul.....	504
G.14 Grafik kosong	505
G.15 Label grafik terpotong	506
G.16 Unggah berkas tidak muncul	507
G.17 Unduhan tidak dimulai.....	508
G.18 nvidia-smi tidak ditemukan	509
G.19 GPU terdeteksi tetapi kode tetap CPU.....	509
G.20 CUDA mismatch.....	510
G.21 RDKit MolFromSmiles menghasilkan None.....	511
G.22 EmbedMolecule gagal	512
G.23 MMFF tidak tersedia	513
G.24 Koordinat XYZ kosong	514
G.25 py3Dmol tidak menampilkan struktur	515
G.26 PySCF gagal membangun molekul.....	515
G.27 SCF tidak konvergen	516
G.28 Energi tampak tidak masuk akal	517
G.29 Optimasi geometri berhenti	518
G.30 Optimasi selesai tetapi geometri aneh	519
G.31 Frekuensi imajiner	520
G.32 HOMO/LUMO salah indeks.....	521
G.33 Momen dipol berubah tanda	522
G.34 Hasil CPU dan GPU sedikit berbeda	522

G.35 Notebook hanya berjalan sebagian	523
G.36 Output lama tertinggal	524
G.37 Versi notebook tidak jelas.....	525
G.38 ZIP tidak memuat semua hasil.....	526
G.39 JSON gagal disimpan.....	527
G.40 Waktu eksekusi sangat berubah	528
LAMPIRAN H - KAMUS 200 PERINTAH DAN POLA KODE PENTING.....	529
H.Python	529
H.001 print()	529
H.002 type().....	529
H.003 isinstance()	529
H.004 len().....	530
H.005 range().....	530
H.006 enumerate()	530
H.007 zip().....	531
H.008 sum()	531
H.009 min().....	531
H.010 max().....	531
H.011 sorted()	532
H.012 any()	532
H.013 all().....	532
H.014 round()	533
H.015 abs()	533
H.016 open().....	533
H.017 with	534
H.018 try-except	534
H.019 raise.....	534
H.020 assert	535
H.Pathlib	535
H.021 Path().....	535
H.022 Path.exists().....	535
H.023 Path.mkdir()	535
H.024 Path.read_text().....	536
H.025 Path.write_text()	536

H.026 Path.glob()	536
H.027 Path.resolve()	537
H.028 Path.stat()	537
H.JSON	537
H.029 json.dumps()	537
H.030 json.loads()	538
H.031 json.dump()	538
H.032 json.load()	538
H.NumPy	539
H.033 np.array()	539
H.034 np.asarray()	539
H.035 np.arange()	539
H.036 np.linspace()	540
H.037 np.zeros()	540
H.038 np.ones()	540
H.039 np.eye()	540
H.040 np.isfinite()	541
H.041 np.isnan()	541
H.042 np.allclose()	541
H.043 np.dot()	542
H.044 np.cross()	542
H.045 np.linalg.norm()	542
H.046 np.linalg.eigh()	542
H.047 np.linalg.solve()	543
H.048 np.mean()	543
H.049 np.median()	543
H.050 np.std()	544
H.051 np.argmin()	544
H.052 np.argmax()	544
H.053 np.where()	544
H.054 np.clip()	545
H.055 np.gradient()	545
H.056 np.trapezoid()	545
H.057 np.interp()	546

H.058 np.save().....	546
H.059 np.load().....	546
H.060 np.savetxt()	546
H.061 np.loadtxt().....	547
H.Pandas	547
H.062 pd.DataFrame()	547
H.063 pd.Series()	547
H.064 pd.read_csv().....	548
H.065 DataFrame.to_csv()	548
H.066 DataFrame.to_excel()	548
H.067 DataFrame.head()	549
H.068 DataFrame.info().....	549
H.069 DataFrame.describe().....	549
H.070 DataFrame.isna().....	549
H.071 DataFrame.dropna()	550
H.072 DataFrame.fillna().....	550
H.073 DataFrame.duplicated()	550
H.074 DataFrame.drop_duplicates()	551
H.075 DataFrame.loc[]	551
H.076 DataFrame.iloc[]	551
H.077 DataFrame.query().....	551
H.078 DataFrame.sort_values().....	552
H.079 DataFrame.groupby().....	552
H.080 DataFrame.agg()	552
H.081 DataFrame.merge()	553
H.082 pd.concat()	553
H.083 DataFrame.pivot_table()	553
H.Matplotlib.....	554
H.084 plt.figure()	554
H.085 plt.plot()	554
H.086 plt.scatter()	554
H.087 plt.hist().....	554
H.088 plt.imshow().....	555
H.089 plt.xlabel().....	555

H.090 plt.ylabel()	555
H.091 plt.title()	556
H.092 plt.legend()	556
H.093 plt.grid()	556
H.094 plt.tight_layout()	556
H.095 plt.savefig()	557
H.096 plt.show()	557
H.Colab	557
H.097 %pip install	557
H.098 !pwd	558
H.099 !ls -lah	558
H.100 files.upload()	558
H.101 files.download()	559
H.102 drive.mount()	559
H.103 nvidia-smi	559
H.104 subprocess.run()	559
H.Archive	560
H.105 ZipFile()	560
H.106 shutil.copy()	560
H.107 shutil.disk_usage()	560
H.Hash	561
H.108 hashlib.sha256()	561
H.Timing	561
H.109 perf_counter()	561
H.Metadata	561
H.110 importlib.metadata.version()	561
H.RDKit	562
H.111 Chem.MolFromSmiles()	562
H.112 Chem.AddHs()	562
H.113 AllChem.ETKDGv3()	562
H.114 AllChem.EmbedMolecule()	563
H.115 AllChem.EmbedMultipleConfs()	563
H.116 AllChem.MMFFHasAllMoleculeParams()	563
H.117 AllChem.MMFFOptimizeMolecule()	564

H.118 AllChem.UFFOptimizeMolecule()	564
H.119 mol.GetNumAtoms()	564
H.120 mol.GetConformer()	564
H.121 atom.GetSymbol()	565
H.122 atom.GetIdx()	565
H.123 Chem.MolToMolBlock()	565
H.py3Dmol	566
H.124 py3Dmol.view()	566
H.125 view.addModel()	566
H.126 view.setStyle()	566
H.127 view.zoomTo()	567
H.128 view.addLabel()	567
H.129 view.show()	567
H.PySCF	568
H.130 gto.M()	568
H.131 mol.natm	568
H.132 mol.nelectron	568
H.133 mol.atom_coords()	569
H.134 mol.atom_symbol()	569
H.135 scf.RHF()	569
H.136 scf.UHF()	569
H.137 dft.RKS()	570
H.138 dft.UKS()	570
H.139 mf.xc	570
H.140 mf.grids.level	571
H.141 mf.conv_tol	571
H.142 mf.max_cycle	571
H.143 mf.kernel()	571
H.144 mf.converged	572
H.145 mf.newton()	572
H.146 mf.make_rdm1()	572
H.147 mf.mo_energy	573
H.148 mf.mo_occ	573
H.149 mf.mo_coeff	573

H.150 mf.dip_moment()	574
H.151 optimize().....	574
H.152 hessian.Hessian()	574
H.Scientific	575
H.153 math.sqrt().....	575
H.154 math.isfinite().....	575
H.155 statistics.mean()	575
H.156 statistics.median()	575
H.157 dataclasses.dataclass	576
H.158 functools.lru_cache	576
H.159 contextlib.redirect_stdout	576
H.Testing	577
H.160 unittest.....	577
H.161 pytest.raises.....	577
H.162 np.testing.assert_allclose	577
H.163 tempfile.TemporaryDirectory	578
H.Formatting.....	578
H.164 f-string.....	578
H.165 textwrap.dedent.....	578
H.166 csv.DictWriter.....	579
H.167 datetime.now()	579
H.168 time.time().....	579
H.System.....	580
H.169 os.getcwd().....	580
H.170 os.cpu_count().....	580
H.171 os.environ.....	580
H.172 platform.platform()	580
H.173 sys.version.....	581
H.174 sys.executable	581
H.175 gc.collect()	581
H.176 warnings.warn()	582
H.177 logging.info()	582
H.178 logging.exception()	582
H.179 subprocess.run().....	583

H.180 shlex.split()	583
H.Data integrity	583
H.181 Path.stat().st_size	583
H.182 hashlib.md5()	584
H.183 hashlib.sha256()	584
H.184 jsonschema.validate	584
H.185 pandas.testing.assert_frame_equal.....	585
H.186 Path.rename().....	585
H.187 Path.unlink()	585
H.Packaging.....	585
H.188 requirements.txt	585
H.189 pip freeze.....	586
H.190 pip show	586
H.191 importlib.util.find_spec.....	586
H.Notebook.....	587
H.192 display()	587
H.193 Markdown()	587
H.194 clear_output().....	587
H.195 %%time.....	588
H.196 %timeit.....	588
H.Pola Notebook.....	588
H.197 checkpoint_01	588
H.198 checkpoint_02	589
H.199 checkpoint_03	589
H.200 checkpoint_04.....	589
LAMPIRAN I Tutorial Google Colab untuk Kimia Komputasi Dasar.....	590
GLOSARIUM 1	613
GLOSARIUM 2	619

1

Mengenal Google Colab dan Notebook Python



BAB 1 - MENGENAL GOOGLE COLAB DAN NOTEBOOK PYTHON

Bab ini membangun fondasi penggunaan Google Colab dan Python. Pembaca belajar membedakan notebook, runtime, sel, output, dan berkas; kemudian berlatih menggunakan tipe data, fungsi, NumPy, Pandas, serta Matplotlib.

Arah Bab: Setiap konsep diikuti prosedur, kode, penjelasan, validasi, troubleshooting, latihan, dan refleksi. Keluaran numerik tidak boleh dipisahkan dari konteks metode dan batas penerapannya.

Komponen	Kriteria
Orientasi	Memahami konsep dan istilah.
Praktik	Menjalankan kode secara bertahap.
Validasi	Memastikan input, proses, dan keluaran sah.
Interpretasi	Menjelaskan hasil tanpa klaim berlebihan.

Capaian Pembelajaran dan Prasyarat

- Menjelaskan konsep utama bab dengan istilah yang tepat.
- Menjalankan notebook dari runtime bersih dan mendokumentasikan versi.
- Memodifikasi parameter tanpa merusak alur validasi.
- Menyimpan hasil dalam format yang dapat dibuka kembali.
- Membedakan keluaran program, interpretasi, dan keterbatasan.

Prasyarat: Kemampuan menggunakan browser, akun Google, operasi berkas dasar, dan kemauan membaca pesan kesalahan secara teliti.

Komponen	Kriteria
Pengetahuan	Konsep sebelumnya dan satuan dasar.
Keterampilan	Menyalin kode, menjalankan sel, dan membaca output.
Sikap ilmiah	Jujur terhadap status pengujian dan ketidakpastian.
Perangkat	Browser modern dan koneksi internet stabil.

Peta Konsep dan Alur Kerja

Subbab	Alur konsep
1.1 Ekosistem Python, Jupyter, dan Google Colab	Python sebagai bahasa komputasi ilmiah → Konsep Jupyter Notebook → Google Colab sebagai notebook berbasis awan
1.2 Membuat dan Mengelola Notebook di Google Colab	Membuat, membuka, mengganti nama, dan menyimpan notebook → Menjalankan sel dan mengelola runtime → Mengunggah dan mengunduh berkas
1.3 Dasar Python untuk Komputasi Ilmiah	Variabel, tipe data, operator, dan struktur kontrol → Fungsi, modul, dan penanganan kesalahan → NumPy, Pandas, dan Matplotlib

Peta konsep dibaca dari kiri ke kanan dan dari input menuju keluaran. Hubungan antarkomponen tidak selalu linear: validasi dapat mengembalikan pembaca ke tahap input, sedangkan troubleshooting dapat memerlukan instalasi ulang atau penggantian metode.

Prinsip alur: Notebook yang baik memperlihatkan jalur keputusan: apa yang diperiksa, kapan proses dihentikan, kapan fallback digunakan, dan berkas apa yang disimpan.

Istilah Kunci dan Konvensi

Istilah	Konvensi penggunaan
notebook	Maknanya dibedakan dari istilah sehari-hari yang mirip.
runtime	Nilai atau statusnya dicatat dalam notebook dan laporan.
kernel	Nilai atau statusnya dicatat dalam notebook dan laporan.
sel Markdown	Nilai atau statusnya dicatat dalam notebook dan laporan.
sel kode	Nilai atau statusnya dicatat dalam notebook dan laporan.
parameter	Maknanya dibedakan dari istilah sehari-hari yang mirip.
validasi	Istilah digunakan secara konsisten dan dijelaskan saat pertama muncul.
konvergensi	Maknanya dibedakan dari istilah sehari-hari yang mirip.
Konvensi satuan: Panjang molekul dinyatakan dalam Angstrom, energi elektronik dalam Hartree, energi orbital dapat dikonversi ke eV, dan perubahan energi dapat dilaporkan dalam kJ mol ⁻¹ .	

Asesmen Diagnostik Awal

7. Apa perbedaan notebook, runtime, dan berkas .ipynb?
8. Mengapa menjalankan sel tidak berurutan dapat menghasilkan keluaran yang menyesatkan?
9. Kapan sebuah nilai numerik boleh ditafsirkan sebagai hasil yang sah?
10. Informasi lingkungan apa yang perlu dicatat agar pekerjaan dapat diulang?
11. Bagaimana membedakan kegagalan instalasi, kegagalan input, dan kegagalan konvergensi?
12. Berkas apa yang harus disimpan sebelum runtime Colab terputus?

Cara menggunakan asesmen: Jawab sebelum memulai bab, tandai tingkat keyakinan, lalu ulangi setelah menyelesaikan proyek bab. Perubahan alasan lebih penting daripada perubahan skor saja.

1.1 Ekosistem Python, Jupyter, dan Google Colab

Ekosistem Python, Jupyter, dan Google Colab mengorganisasi tiga komponen yang saling bergantung. Pembahasan bergerak dari konsep, menuju prosedur, kemudian menghasilkan artefak yang dapat divalidasi dan disimpan.

Sub-subbab	Fokus
1.1.1 Python sebagai bahasa komputasi ilmiah	Mengenal Python, interpreter, pustaka, paket, modul, dan perannya dalam sains.
1.1.2 Konsep Jupyter Notebook	Hubungan sel Markdown, sel kode, kernel, output, variabel, dan urutan eksekusi.
1.1.3 Google Colab sebagai notebook berbasis awan	Runtime, CPU, GPU, TPU, penyimpanan sementara, batas sumber daya, dan keamanan notebook.
Produk belajar: Satu notebook terstruktur, satu berkas keluaran, satu catatan validasi, dan satu interpretasi singkat yang menyatakan batas hasil.	

Rencana Praktik dan Produk Belajar

13. Pelajari Python sebagai bahasa komputasi ilmiah dan jalankan kode dasar yang menyertainya.
14. Pelajari Konsep Jupyter Notebook dan jalankan kode dasar yang menyertainya.
15. Pelajari Google Colab sebagai notebook berbasis awan dan jalankan kode dasar yang menyertainya.

Komponen	Kriteria
Waktu	60-120 menit, bergantung pada komputasi.
Input	Notebook baru atau salinan notebook bab.
Produk	Kode, output, berkas hasil, dan refleksi.

Uji	Restart session lalu Run all.
Dokumentasi	Versi, parameter, status, dan waktu eksekusi.
Kriteria selesai: Semua sel dapat dijalankan berurutan, pesan kesalahan ditangani, dan hasil utama tersimpan di lokasi persisten.	

1.1.1 Python sebagai bahasa komputasi ilmiah

Pembahasan Python sebagai bahasa komputasi ilmiah ditempatkan dalam konteks praktik Google Colab agar pembaca dapat menghubungkan konsep, prosedur, kode, dan interpretasi secara utuh. Mengetahui Python, interpreter, pustaka, paket, modul, dan perannya dalam sains. Hasil yang dituju berupa nilai, teks, tabel, atau grafik.

Prinsip kerja yang digunakan adalah memisahkan input, proses, validasi, dan ekspor. Pemisahan ini membuat kesalahan lebih mudah dilacak dan mencegah hasil lama dianggap sebagai hasil dari parameter baru. Setiap langkah perlu meninggalkan bukti yang dapat dibaca kembali: nilai parameter, versi pustaka, status proses, nama berkas, serta pesan kesalahan bila terjadi kegagalan.

Tujuan operasional: Pembaca mampu menjelaskan Python sebagai bahasa komputasi ilmiah, menjalankan contoh minimal, memeriksa keberhasilan, dan menyimpan bukti hasil.

Komponen	Kriteria
Konsep	Mengenal Python, interpreter, pustaka, paket, modul, dan perannya dalam sains.
Input	Parameter dan data dinyatakan eksplisit.
Proses	Langkah dapat diulang dari runtime bersih.
Output	Nilai, berkas, atau visualisasi yang dapat diperiksa.
Batas	Interpretasi mengikuti metode dan konteks.

Landasan Teori dan Batas Penerapan

Pembahasan Python sebagai bahasa komputasi ilmiah ditempatkan dalam konteks praktik Google Colab agar pembaca dapat menghubungkan konsep, prosedur, kode, dan interpretasi secara utuh. Mengenal Python, interpreter, pustaka, paket, modul, dan perannya dalam sains. Hasil yang dituju berupa nilai, teks, tabel, atau grafik.

Prinsip kerja yang digunakan adalah memisahkan input, proses, validasi, dan ekspor. Pemisahan ini membuat kesalahan lebih mudah dilacak dan mencegah hasil lama dianggap sebagai hasil dari parameter baru. Setiap langkah perlu meninggalkan bukti yang dapat dibaca kembali: nilai parameter, versi pustaka, status proses, nama berkas, serta pesan kesalahan bila terjadi kegagalan.

Risiko utama pada topik ini adalah tipe data dan urutan operasi. Karena itu, notebook harus memuat pemeriksaan eksplisit dan bukan hanya mengandalkan keluaran yang tampak masuk akal. Pembaca dianjurkan menjalankan notebook dari keadaan runtime bersih, membandingkan hasil dengan batas kewajaran, dan menyimpan artefak penting di lokasi persisten.

Batas penerapan: Contoh pada buku berfungsi sebagai pola kerja. Ukuran molekul, biaya komputasi, versi pustaka, dan perangkat keras dapat mengubah waktu serta strategi yang diperlukan.

Prosedur Bertahap

Pembahasan Python sebagai bahasa komputasi ilmiah ditempatkan dalam konteks praktik Google Colab agar pembaca dapat menghubungkan konsep, prosedur, kode, dan interpretasi secara utuh. Mengenal Python, interpreter, pustaka, paket, modul, dan perannya dalam sains. Hasil yang dituju berupa nilai, teks, tabel, atau grafik.

16. Tuliskan tujuan operasional Python sebagai bahasa komputasi ilmiah dan tentukan keluaran yang harus tersedia pada akhir sel.
17. Tetapkan parameter pada satu sel konfigurasi agar perubahan dapat ditelusuri dan tidak tersebar di banyak tempat.
18. Siapkan skrip Python dengan input yang eksplisit; hindari nilai tersembunyi yang hanya tersimpan dalam state kernel.
19. Jalankan pemeriksaan prasyarat, termasuk keberadaan pustaka, bentuk data, tipe nilai, serta konsistensi satuan.
20. Eksekusi proses utama, tangkap status konvergensi atau keberhasilan, dan hentikan alur apabila syarat minimum tidak terpenuhi.
21. Simpan nilai, teks, tabel, atau grafik dengan nama berkas informatif, kemudian tulis interpretasi yang membedakan fakta komputasi dan kesimpulan ilmiah.

Titik kontrol: Jangan melanjutkan ke interpretasi apabila validasi input atau status proses belum memenuhi kriteria.

Kode Dasar Siap Colab

Python sebagai bahasa komputasi ilmiah merupakan simpul penting dalam alur kerja notebook ilmiah karena menentukan bagaimana skrip Python disiapkan, dijalankan, dan diperiksa. Mengetahui Python, interpreter, pustaka, paket, modul, dan perannya dalam sains. Hasil yang dituju berupa nilai, teks, tabel, atau grafik.

Sel kode contoh

```
import sys
import math

print("Versi Python:", sys.version.split()[0])
energi_hartree = -1.117349
energi_ev = energi_hartree * 27.211386245988
print(f"Energi = {energi_ev:.6f} eV")
```

Cara menjalankan: Tempatkan kode dalam satu atau beberapa sel sesuai komentar. Untuk baris %pip atau perintah shell, gunakan runtime Google Colab.

Penjelasan Baris Kode Penting

Potongan kode	Fungsi
import sys	Mendefinisikan input atau parameter yang akan digunakan pada langkah berikutnya.
import math	Mencetak status agar pengguna tidak hanya bergantung pada keluaran visual.
print("Versi Python:", sys.version.split()[0])	Menjalankan operasi utama dan menyimpan hasil pada variabel yang jelas.
energi_hartree = -1.117349	Mencetak status agar pengguna tidak hanya bergantung pada keluaran visual.
energi_ev = energi_hartree * 27.211386245988	Menjalankan operasi utama dan menyimpan hasil pada variabel yang jelas.

```
print(f'Energi = {energi_ev:.6f} eV')
```

Menjalankan operasi utama dan menyimpan hasil pada variabel yang jelas.

Nama variabel dipilih agar maknanya terbaca. Hindari singkatan yang tidak dijelaskan, nilai konstan tanpa satuan, serta penggunaan variabel dari sel lama. Komentar harus menerangkan alasan atau asumsi, bukan sekadar mengulang sintaks.

Validasi dan Interpretasi Hasil

Prinsip kerja yang digunakan adalah memisahkan input, proses, validasi, dan ekspor. Pemisahan ini membuat kesalahan lebih mudah dilacak dan mencegah hasil lama dianggap sebagai hasil dari parameter baru. Setiap langkah perlu meninggalkan bukti yang dapat dibaca kembali: nilai parameter, versi pustaka, status proses, nama berkas, serta pesan kesalahan bila terjadi kegagalan.

Aspek validasi	Kriteria lulus
Input	Tipe, bentuk, satuan, muatan, spin, dan jalur berkas dinyatakan eksplisit.
Lingkungan	Versi Python dan pustaka tercatat; backend CPU/GPU terdeteksi.
Proses	Status eksekusi atau konvergensi diperiksa sebelum hasil dipakai.
Numerik	Tidak ada NaN/inf; ukuran array dan jumlah atom konsisten.
Keluaran	Nilai, teks, tabel, atau grafik disimpan dan dapat dibuka kembali.
Interpretasi	Klaim tidak melampaui metode, basis set, atau data yang dihitung.
Interpretasi aman: Sebutkan metode, parameter, satuan, status konvergensi, dan keterbatasan saat menuliskan kesimpulan.	

Kesalahan Umum dan Troubleshooting

Risiko utama pada topik ini adalah tipe data dan urutan operasi. Karena itu, notebook harus memuat pemeriksaan eksplisit dan bukan hanya mengandalkan keluaran yang

tampak masuk akal. Pembaca dianjurkan menjalankan notebook dari keadaan runtime bersih, membandingkan hasil dengan batas kewajaran, dan menyimpan artefak penting di lokasi persisten.

Gejala	Tindakan perbaikan
Sel dijalankan tidak berurutan	Restart session, lalu gunakan Run all dan periksa variabel prasyarat.
Pustaka tidak ditemukan	Jalankan kembali sel %pip, periksa nama paket, lalu restart bila dependensi inti berubah.
Input tampak benar tetapi gagal	Cetak tipe, shape, satuan, muatan, spin, dan beberapa baris pertama data.
Tipe data dan urutan operasi	Tambahkan validasi sebelum proses utama dan hentikan eksekusi dengan pesan yang spesifik.
Hasil berbeda antar-runtime	Catat versi runtime/paket, random seed, parameter, dan perangkat yang digunakan.
Aturan troubleshooting: Perbaiki satu hal pada satu waktu, simpan pesan kesalahan lengkap, dan catat perubahan versi atau parameter.	

Latihan Modifikasi dan Refleksi

- Ubah satu parameter pada contoh Python sebagai bahasa komputasi ilmiah, jalankan ulang dari runtime bersih, dan jelaskan perubahan keluaran.
- Tambahkan satu validasi baru yang mencegah tipe data dan urutan operasi.
- Buat berkas ringkasan TXT atau JSON yang memuat input, status proses, waktu, dan hasil utama.
- Jelaskan bagian mana yang merupakan fakta keluaran program dan bagian mana yang merupakan interpretasi ilmiah.
- Susun satu kasus uji yang sengaja salah, lalu pastikan notebook menghasilkan pesan kesalahan yang informatif.

Refleksi: Tuliskan apa yang berubah setelah kode dimodifikasi, bukti apa yang menunjukkan keberhasilan, dan bagian mana yang masih memerlukan verifikasi eksternal.

Rubrik

Deskripsi

Skor 4	Kode berjalan dari runtime bersih, validasi lengkap, interpretasi tepat.
Skor 3	Kode berjalan dan hasil tersimpan, tetapi dokumentasi belum lengkap.
Skor 2	Kode hanya berjalan setelah intervensi atau state tertentu.
Skor 1	Kode belum berjalan dan diagnosis kesalahan belum sistematis.

1.1.2 Konsep Jupyter Notebook

Konsep Jupyter Notebook merupakan simpul penting dalam alur kerja notebook ilmiah karena menentukan bagaimana notebook disiapkan, dijalankan, dan diperiksa. Hubungan sel Markdown, sel kode, kernel, output, variabel, dan urutan eksekusi. Hasil yang dituju berupa sel Markdown, sel kode, output, dan metadata.

Prinsip kerja yang digunakan adalah memisahkan input, proses, validasi, dan ekspor. Pemisahan ini membuat kesalahan lebih mudah dilacak dan mencegah hasil lama dianggap sebagai hasil dari parameter baru. Setiap langkah perlu meninggalkan bukti yang dapat dibaca kembali: nilai parameter, versi pustaka, status proses, nama berkas, serta pesan kesalahan bila terjadi kegagalan.

Tujuan operasional: Pembaca mampu menjelaskan Konsep Jupyter Notebook, menjalankan contoh minimal, memeriksa keberhasilan, dan menyimpan bukti hasil.

Komponen	Kriteria
Konsep	Hubungan sel Markdown, sel kode, kernel, output, variabel, dan urutan eksekusi.
Input	Parameter dan data dinyatakan eksplisit.
Proses	Langkah dapat diulang dari runtime bersih.
Output	Nilai, berkas, atau visualisasi yang dapat diperiksa.

Batas	Interpretasi mengikuti metode dan konteks.

Landasan Teori dan Batas Penerapan

Pembahasan Konsep Jupyter Notebook ditempatkan dalam konteks praktik Google Colab agar pembaca dapat menghubungkan konsep, prosedur, kode, dan interpretasi secara utuh. Hubungan sel Markdown, sel kode, kernel, output, variabel, dan urutan eksekusi. Hasil yang dituju berupa sel Markdown, sel kode, output, dan metadata.

Prinsip kerja yang digunakan adalah memisahkan input, proses, validasi, dan ekspor. Pemisahan ini membuat kesalahan lebih mudah dilacak dan mencegah hasil lama dianggap sebagai hasil dari parameter baru. Setiap langkah perlu meninggalkan bukti yang dapat dibaca kembali: nilai parameter, versi pustaka, status proses, nama berkas, serta pesan kesalahan bila terjadi kegagalan.

Risiko utama pada topik ini adalah state kernel tersembunyi dan urutan eksekusi. Karena itu, notebook harus memuat pemeriksaan eksplisit dan bukan hanya mengandalkan keluaran yang tampak masuk akal. Pembaca dianjurkan menjalankan notebook dari keadaan runtime bersih, membandingkan hasil dengan batas kewajaran, dan menyimpan artefak penting di lokasi persisten.

Batas penerapan: Contoh pada buku berfungsi sebagai pola kerja. Ukuran molekul, biaya komputasi, versi pustaka, dan perangkat keras dapat mengubah waktu serta strategi yang diperlukan.

Prosedur Bertahap

Pembahasan Konsep Jupyter Notebook ditempatkan dalam konteks praktik Google Colab agar pembaca dapat menghubungkan konsep, prosedur, kode, dan interpretasi secara utuh. Hubungan sel Markdown, sel kode, kernel, output, variabel, dan urutan eksekusi. Hasil yang dituju berupa sel Markdown, sel kode, output, dan metadata.

27. Tuliskan tujuan operasional Konsep Jupyter Notebook dan tentukan keluaran yang harus tersedia pada akhir sel.
28. Tetapkan parameter pada satu sel konfigurasi agar perubahan dapat ditelusuri dan tidak tersebar di banyak tempat.
29. Siapkan notebook dengan input yang eksplisit; hindari nilai tersembunyi yang hanya tersimpan dalam state kernel.
30. Jalankan pemeriksaan prasyarat, termasuk keberadaan pustaka, bentuk data, tipe nilai, serta konsistensi satuan.
31. Eksekusi proses utama, tangkap status konvergensi atau keberhasilan, dan hentikan alur apabila syarat minimum tidak terpenuhi.

32. Simpan sel Markdown, sel kode, output, dan metadata dengan nama berkas informatif, kemudian tulis interpretasi yang membedakan fakta komputasi dan kesimpulan ilmiah.

Titik kontrol: Jangan melanjutkan ke interpretasi apabila validasi input atau status proses belum memenuhi kriteria.

Kode Dasar Siap Colab

Pembahasan Konsep Jupyter Notebook ditempatkan dalam konteks praktik Google Colab agar pembaca dapat menghubungkan konsep, prosedur, kode, dan interpretasi secara utuh. Hubungan sel Markdown, sel kode, kernel, output, variabel, dan urutan eksekusi. Hasil yang dituju berupa sel Markdown, sel kode, output, dan metadata.

Sel kode contoh

```
# Sel 1: membuat state
a = 12
b = 5

# Sel 2: menggunakan state
hasil = a**2 + b**2
print("Hasil:", hasil)

# Uji reproduksibilitas: Restart session, lalu Run all.
```

Cara menjalankan: Tempatkan kode dalam satu atau beberapa sel sesuai komentar. Untuk baris %pip atau perintah shell, gunakan runtime Google Colab.

Penjelasan Baris Kode Penting

Potongan kode	Fungsi
# Sel 1: membuat state	Mendefinisikan input atau parameter yang akan digunakan pada langkah berikutnya.
a = 12	Mendefinisikan input atau parameter yang akan digunakan pada langkah berikutnya.
b = 5	Menambahkan pemeriksaan untuk mencegah hasil tidak valid diteruskan.
# Sel 2: menggunakan state	Menambahkan pemeriksaan untuk mencegah hasil tidak valid diteruskan.

<code>hasil = a**2 + b**2</code>	Menjalankan operasi utama dan menyimpan hasil pada variabel yang jelas.
<code>print("Hasil:", hasil)</code>	Mencetak status agar pengguna tidak hanya bergantung pada keluaran visual.
<code># Uji reproduksibilitas: Restart session, lalu R</code>	Mendefinisikan input atau parameter yang akan digunakan pada langkah berikutnya.

Nama variabel dipilih agar maknanya terbaca. Hindari singkatan yang tidak dijelaskan, nilai konstan tanpa satuan, serta penggunaan variabel dari sel lama. Komentar harus menerangkan alasan atau asumsi, bukan sekadar mengulang sintaks.

Validasi dan Interpretasi Hasil

Prinsip kerja yang digunakan adalah memisahkan input, proses, validasi, dan ekspor. Pemisahan ini membuat kesalahan lebih mudah dilacak dan mencegah hasil lama dianggap sebagai hasil dari parameter baru. Setiap langkah perlu meninggalkan bukti yang dapat dibaca kembali: nilai parameter, versi pustaka, status proses, nama berkas, serta pesan kesalahan bila terjadi kegagalan.

Aspek validasi	Kriteria lulus
Input	Tipe, bentuk, satuan, muatan, spin, dan jalur berkas dinyatakan eksplisit.
Lingkungan	Versi Python dan pustaka tercatat; backend CPU/GPU terdeteksi.
Proses	Status eksekusi atau konvergensi diperiksa sebelum hasil dipakai.
Numerik	Tidak ada NaN/inf; ukuran array dan jumlah atom konsisten.
Keluaran	Sel markdown, sel kode, output, dan metadata disimpan dan dapat dibuka kembali.

Interpretasi

Klaim tidak melampaui metode, basis set, atau data yang dihitung.

Interpretasi aman: Sebutkan metode, parameter, satuan, status konvergensi, dan keterbatasan saat menuliskan kesimpulan.

Kesalahan Umum dan Troubleshooting

Risiko utama pada topik ini adalah state kernel tersembunyi dan urutan eksekusi. Karena itu, notebook harus memuat pemeriksaan eksplisit dan bukan hanya mengandalkan keluaran yang tampak masuk akal. Pembaca dianjurkan menjalankan notebook dari keadaan runtime bersih, membandingkan hasil dengan batas kewajaran, dan menyimpan artefak penting di lokasi persisten.

Gejala	Tindakan perbaikan
Sel dijalankan tidak berurutan	Restart session, lalu gunakan Run all dan periksa variabel prasyarat.
Pustaka tidak ditemukan	Jalankan kembali sel %pip, periksa nama paket, lalu restart bila dependensi inti berubah.
Input tampak benar tetapi gagal	Cetak tipe, shape, satuan, muatan, spin, dan beberapa baris pertama data.
State kernel tersembunyi dan urutan eksekusi	Tambahkan validasi sebelum proses utama dan hentikan eksekusi dengan pesan yang spesifik.
Hasil berbeda antar-runtime	Catat versi runtime/paket, random seed, parameter, dan perangkat yang digunakan.
Aturan troubleshooting: Perbaiki satu hal pada satu waktu, simpan pesan kesalahan lengkap, dan catat perubahan versi atau parameter.	

Latihan Modifikasi dan Refleksi

- Ubah satu parameter pada contoh Konsep Jupyter Notebook, jalankan ulang dari runtime bersih, dan jelaskan perubahan keluaran.
- Tambahkan satu validasi baru yang mencegah state kernel tersembunyi dan urutan eksekusi.

35. Buat berkas ringkasan TXT atau JSON yang memuat input, status proses, waktu, dan hasil utama.
36. Jelaskan bagian mana yang merupakan fakta keluaran program dan bagian mana yang merupakan interpretasi ilmiah.
37. Susun satu kasus uji yang sengaja salah, lalu pastikan notebook menghasilkan pesan kesalahan yang informatif.

Refleksi: Tuliskan apa yang berubah setelah kode dimodifikasi, bukti apa yang menunjukkan keberhasilan, dan bagian mana yang masih memerlukan verifikasi eksternal.

Rubrik	Deskripsi
Skor 4	Kode berjalan dari runtime bersih, validasi lengkap, interpretasi tepat.
Skor 3	Kode berjalan dan hasil tersimpan, tetapi dokumentasi belum lengkap.
Skor 2	Kode hanya berjalan setelah intervensi atau state tertentu.
Skor 1	Kode belum berjalan dan diagnosis kesalahan belum sistematis.

1.1.3 Google Colab sebagai notebook berbasis awan

Google Colab sebagai notebook berbasis awan merupakan simpul penting dalam alur kerja notebook ilmiah karena menentukan bagaimana notebook disiapkan, dijalankan, dan diperiksa. Runtime, CPU, GPU, TPU, penyimpanan sementara, batas sumber daya, dan keamanan notebook. Hasil yang dituju berupa sel Markdown, sel kode, output, dan metadata.

Prinsip kerja yang digunakan adalah memisahkan input, proses, validasi, dan ekspor. Pemisahan ini membuat kesalahan lebih mudah dilacak dan mencegah hasil lama dianggap sebagai hasil dari parameter baru. Setiap langkah perlu meninggalkan bukti yang dapat dibaca kembali: nilai parameter, versi pustaka, status proses, nama berkas, serta pesan kesalahan bila terjadi kegagalan.

Tujuan operasional: Pembaca mampu menjelaskan Google Colab sebagai notebook berbasis awan, menjalankan contoh minimal, memeriksa keberhasilan, dan menyimpan bukti hasil.

Komponen	Kriteria
Konsep	Runtime, CPU, GPU, TPU, penyimpanan sementara, batas sumber daya, dan keamanan notebook.
Input	Parameter dan data dinyatakan eksplisit.
Proses	Langkah dapat diulang dari runtime bersih.
Output	Nilai, berkas, atau visualisasi yang dapat diperiksa.
Batas	Interpretasi mengikuti metode dan konteks.

Landasan Teori dan Batas Penerapan

Google Colab sebagai notebook berbasis awan merupakan simpul penting dalam alur kerja notebook ilmiah karena menentukan bagaimana notebook disiapkan, dijalankan, dan diperiksa. Runtime, CPU, GPU, TPU, penyimpanan sementara, batas sumber daya, dan keamanan notebook. Hasil yang dituju berupa sel Markdown, sel kode, output, dan metadata.

Prinsip kerja yang digunakan adalah memisahkan input, proses, validasi, dan ekspor. Pemisahan ini membuat kesalahan lebih mudah dilacak dan mencegah hasil lama dianggap sebagai hasil dari parameter baru. Setiap langkah perlu meninggalkan bukti yang dapat dibaca kembali: nilai parameter, versi pustaka, status proses, nama berkas, serta pesan kesalahan bila terjadi kegagalan.

Risiko utama pada topik ini adalah state kernel tersembunyi dan urutan eksekusi. Karena itu, notebook harus memuat pemeriksaan eksplisit dan bukan hanya mengandalkan keluaran yang tampak masuk akal. Pembaca dianjurkan menjalankan notebook dari keadaan runtime bersih, membandingkan hasil dengan batas kewajaran, dan menyimpan artefak penting di lokasi persisten.

Batas penerapan: Contoh pada buku berfungsi sebagai pola kerja. Ukuran molekul, biaya komputasi, versi pustaka, dan perangkat keras dapat mengubah waktu serta strategi yang diperlukan.

Prosedur Bertahap

Google Colab sebagai notebook berbasis awan merupakan simpul penting dalam alur kerja notebook ilmiah karena menentukan bagaimana notebook disiapkan, dijalankan, dan diperiksa. Runtime, CPU, GPU, TPU, penyimpanan sementara, batas sumber daya, dan keamanan notebook. Hasil yang dituju berupa sel Markdown, sel kode, output, dan metadata.

38. Tuliskan tujuan operasional Google Colab sebagai notebook berbasis awan dan tentukan keluaran yang harus tersedia pada akhir sel.
39. Tetapkan parameter pada satu sel konfigurasi agar perubahan dapat ditelusuri dan tidak tersebar di banyak tempat.
40. Siapkan notebook dengan input yang eksplisit; hindari nilai tersembunyi yang hanya tersimpan dalam state kernel.
41. Jalankan pemeriksaan prasyarat, termasuk keberadaan pustaka, bentuk data, tipe nilai, serta konsistensi satuan.
42. Eksekusi proses utama, tangkap status konvergensi atau keberhasilan, dan hentikan alur apabila syarat minimum tidak terpenuhi.
43. Simpan sel Markdown, sel kode, output, dan metadata dengan nama berkas informatif, kemudian tulis interpretasi yang membedakan fakta komputasi dan kesimpulan ilmiah.

Titik kontrol: Jangan melanjutkan ke interpretasi apabila validasi input atau status proses belum memenuhi kriteria.

Kode Dasar Siap Colab

Google Colab sebagai notebook berbasis awan merupakan simpul penting dalam alur kerja notebook ilmiah karena menentukan bagaimana notebook disiapkan, dijalankan, dan diperiksa. Runtime, CPU, GPU, TPU, penyimpanan sementara, batas sumber daya, dan keamanan notebook. Hasil yang dituju berupa sel Markdown, sel kode, output, dan metadata.

Sel kode contoh

```
import os, platform, sys

print("Python :", sys.version.split()[0])
print("Sistem :", platform.platform())
print("Folder :", os.getcwd())
print("CPU    :", os.cpu_count())

# Di Colab, ruang kerja utama biasanya /content.
```


Cara menjalankan: Tempatkan kode dalam satu atau beberapa sel sesuai komentar. Untuk baris %pip atau perintah shell, gunakan runtime Google Colab.

Penjelasan Baris Kode Penting

Potongan kode	Fungsi
<code>import os, platform, sys</code>	Mencetak status agar pengguna tidak hanya bergantung pada keluaran visual.
<code>print("Python :", sys.version.split()[0])</code>	Mendefinisikan input atau parameter yang akan digunakan pada langkah berikutnya.
<code>print("Sistem :", platform.platform())</code>	Menjalankan operasi utama dan menyimpan hasil pada variabel yang jelas.
<code>print("Folder :", os.getcwd())</code>	Menambahkan pemeriksaan untuk mencegah hasil tidak valid diteruskan.
<code>print("CPU :", os.cpu_count())</code>	Mencetak status agar pengguna tidak hanya bergantung pada keluaran visual.
<code># Di Colab, ruang kerja utama biasanya /content.</code>	Mendefinisikan input atau parameter yang akan digunakan pada langkah berikutnya.

Nama variabel dipilih agar maknanya terbaca. Hindari singkatan yang tidak dijelaskan, nilai konstan tanpa satuan, serta penggunaan variabel dari sel lama. Komentar harus menerangkan alasan atau asumsi, bukan sekadar mengulang sintaks.

Validasi dan Interpretasi Hasil

Prinsip kerja yang digunakan adalah memisahkan input, proses, validasi, dan ekspor. Pemisahan ini membuat kesalahan lebih mudah dilacak dan mencegah hasil lama dianggap sebagai hasil dari parameter baru. Setiap langkah perlu meninggalkan bukti yang dapat dibaca kembali: nilai parameter, versi pustaka, status proses, nama berkas, serta pesan kesalahan bila terjadi kegagalan.

Aspek validasi

Kriteria lulus

Input	Tipe, bentuk, satuan, muatan, spin, dan jalur berkas dinyatakan eksplisit.
Lingkungan	Versi Python dan pustaka tercatat; backend CPU/GPU terdeteksi.
Proses	Status eksekusi atau konvergensi diperiksa sebelum hasil dipakai.
Numerik	Tidak ada NaN/inf; ukuran array dan jumlah atom konsisten.
Keluaran	Sel markdown, sel kode, output, dan metadata disimpan dan dapat dibuka kembali.
Interpretasi	Klaim tidak melampaui metode, basis set, atau data yang dihitung.
Interpretasi aman: Sebutkan metode, parameter, satuan, status konvergensi, dan keterbatasan saat menuliskan kesimpulan.	

Kesalahan Umum dan Troubleshooting

Risiko utama pada topik ini adalah state kernel tersembunyi dan urutan eksekusi. Karena itu, notebook harus memuat pemeriksaan eksplisit dan bukan hanya mengandalkan keluaran yang tampak masuk akal. Pembaca dianjurkan menjalankan notebook dari keadaan runtime bersih, membandingkan hasil dengan batas kewajaran, dan menyimpan artefak penting di lokasi persisten.

Gejala	Tindakan perbaikan
Sel dijalankan tidak berurutan	Restart session, lalu gunakan Run all dan periksa variabel prasyarat.
Pustaka tidak ditemukan	Jalankan kembali sel %pip, periksa nama paket, lalu restart bila dependensi inti berubah.
Input tampak benar tetapi gagal	Cetak tipe, shape, satuan, muatan, spin, dan beberapa baris pertama data.

State kernel tersembunyi dan urutan eksekusi	Tambahkan validasi sebelum proses utama dan hentikan eksekusi dengan pesan yang spesifik.
Hasil berbeda antar-runtime	Catat versi runtime/paket, random seed, parameter, dan perangkat yang digunakan.
Aturan troubleshooting: Perbaiki satu hal pada satu waktu, simpan pesan kesalahan lengkap, dan catat perubahan versi atau parameter.	

Latihan Modifikasi dan Refleksi

44. Ubah satu parameter pada contoh Google Colab sebagai notebook berbasis awan, jalankan ulang dari runtime bersih, dan jelaskan perubahan keluaran.
45. Tambahkan satu validasi baru yang mencegah state kernel tersembunyi dan urutan eksekusi.
46. Buat berkas ringkasan TXT atau JSON yang memuat input, status proses, waktu, dan hasil utama.
47. Jelaskan bagian mana yang merupakan fakta keluaran program dan bagian mana yang merupakan interpretasi ilmiah.
48. Susun satu kasus uji yang sengaja salah, lalu pastikan notebook menghasilkan pesan kesalahan yang informatif.

Refleksi: Tuliskan apa yang berubah setelah kode dimodifikasi, bukti apa yang menunjukkan keberhasilan, dan bagian mana yang masih memerlukan verifikasi eksternal.

Rubrik	Deskripsi
Skor 4	Kode berjalan dari runtime bersih, validasi lengkap, interpretasi tepat.
Skor 3	Kode berjalan dan hasil tersimpan, tetapi dokumentasi belum lengkap.
Skor 2	Kode hanya berjalan setelah intervensi atau state tertentu.
Skor 1	Kode belum berjalan dan diagnosis kesalahan belum sistematis.

1.2 Membuat dan Mengelola Notebook di Google Colab

Membuat dan Mengelola Notebook di Google Colab mengorganisasi tiga komponen yang saling bergantung. Pembahasan bergerak dari konsep, menuju prosedur, kemudian menghasilkan artefak yang dapat divalidasi dan disimpan.

Sub-subbab	Fokus
1.2.1 Membuat, membuka, mengganti nama, dan menyimpan notebook	Pembuatan notebook baru, pembukaan .ipynb, penyimpanan, salinan, dan berbagi notebook.
1.2.2 Menjalankan sel dan mengelola runtime	Menjalankan satu sel, seluruh sel, menghentikan proses, memulai ulang runtime, dan menghapus output.
1.2.3 Mengunggah dan mengunduh berkas	Mengunggah data, menghubungkan Google Drive, mengunduh hasil, dan memahami direktori /content.
Produk belajar: Satu notebook terstruktur, satu berkas keluaran, satu catatan validasi, dan satu interpretasi singkat yang menyatakan batas hasil.	

Rencana Praktik dan Produk Belajar

49. Pelajari Membuat, membuka, mengganti nama, dan menyimpan notebook dan jalankan kode dasar yang menyertainya.
50. Pelajari Menjalankan sel dan mengelola runtime dan jalankan kode dasar yang menyertainya.
51. Pelajari Mengunggah dan mengunduh berkas dan jalankan kode dasar yang menyertainya.

Komponen	Kriteria
Waktu	60-120 menit, bergantung pada komputasi.
Input	Notebook baru atau salinan notebook bab.
Produk	Kode, output, berkas hasil, dan refleksi.
Uji	Restart session lalu Run all.

Dokumentasi

Versi, parameter, status, dan waktu eksekusi.

Kriteria selesai: Semua sel dapat dijalankan berurutan, pesan kesalahan ditangani, dan hasil utama tersimpan di lokasi persisten.

1.2.1 Membuat, membuka, mengganti nama, dan menyimpan notebook

Pada bagian Membuat, membuka, mengganti nama, dan menyimpan notebook, perhatian tidak hanya diarahkan pada keberhasilan menjalankan kode, tetapi juga pada ketepatan asumsi, keterlacakan parameter, dan mutu keluaran. Pembuatan notebook baru, pembukaan .ipynb, penyimpanan, salinan, dan berbagi notebook. Hasil yang dituju berupa sel Markdown, sel kode, output, dan metadata.

Prinsip kerja yang digunakan adalah memisahkan input, proses, validasi, dan ekspor. Pemisahan ini membuat kesalahan lebih mudah dilacak dan mencegah hasil lama dianggap sebagai hasil dari parameter baru. Setiap langkah perlu meninggalkan bukti yang dapat dibaca kembali: nilai parameter, versi pustaka, status proses, nama berkas, serta pesan kesalahan bila terjadi kegagalan.

Tujuan operasional: Pembaca mampu menjelaskan Membuat, membuka, mengganti nama, dan menyimpan notebook, menjalankan contoh minimal, memeriksa keberhasilan, dan menyimpan bukti hasil.

Komponen	Kriteria
Konsep	Pembuatan notebook baru, pembukaan .ipynb, penyimpanan, salinan, dan berbagi notebook.
Input	Parameter dan data dinyatakan eksplisit.
Proses	Langkah dapat diulang dari runtime bersih.
Output	Nilai, berkas, atau visualisasi yang dapat diperiksa.
Batas	Interpretasi mengikuti metode dan konteks.

Landasan Teori dan Batas Penerapan

Membuat, membuka, mengganti nama, dan menyimpan notebook merupakan simpul penting dalam alur kerja notebook ilmiah karena menentukan bagaimana notebook

disiapkan, dijalankan, dan diperiksa. Pembuatan notebook baru, pembukaan .ipynb, penyimpanan, salinan, dan berbagi notebook. Hasil yang dituju berupa sel Markdown, sel kode, output, dan metadata.

Prinsip kerja yang digunakan adalah memisahkan input, proses, validasi, dan ekspor. Pemisahan ini membuat kesalahan lebih mudah dilacak dan mencegah hasil lama dianggap sebagai hasil dari parameter baru. Setiap langkah perlu meninggalkan bukti yang dapat dibaca kembali: nilai parameter, versi pustaka, status proses, nama berkas, serta pesan kesalahan bila terjadi kegagalan.

Risiko utama pada topik ini adalah state kernel tersembunyi dan urutan eksekusi. Karena itu, notebook harus memuat pemeriksaan eksplisit dan bukan hanya mengandalkan keluaran yang tampak masuk akal. Pembaca dianjurkan menjalankan notebook dari keadaan runtime bersih, membandingkan hasil dengan batas kewajaran, dan menyimpan artefak penting di lokasi persisten.

Batas penerapan: Contoh pada buku berfungsi sebagai pola kerja. Ukuran molekul, biaya komputasi, versi pustaka, dan perangkat keras dapat mengubah waktu serta strategi yang diperlukan.

Prosedur Bertahap

Membuat, membuka, mengganti nama, dan menyimpan notebook merupakan simpul penting dalam alur kerja notebook ilmiah karena menentukan bagaimana notebook disiapkan, dijalankan, dan diperiksa. Pembuatan notebook baru, pembukaan .ipynb, penyimpanan, salinan, dan berbagi notebook. Hasil yang dituju berupa sel Markdown, sel kode, output, dan metadata.

52. Tuliskan tujuan operasional Membuat, membuka, mengganti nama, dan menyimpan notebook dan tentukan keluaran yang harus tersedia pada akhir sel.
53. Tetapkan parameter pada satu sel konfigurasi agar perubahan dapat ditelusuri dan tidak tersebar di banyak tempat.
54. Siapkan notebook dengan input yang eksplisit; hindari nilai tersembunyi yang hanya tersimpan dalam state kernel.
55. Jalankan pemeriksaan prasyarat, termasuk keberadaan pustaka, bentuk data, tipe nilai, serta konsistensi satuan.
56. Eksekusi proses utama, tangkap status konvergensi atau keberhasilan, dan hentikan alur apabila syarat minimum tidak terpenuhi.
57. Simpan sel Markdown, sel kode, output, dan metadata dengan nama berkas informatif, kemudian tulis interpretasi yang membedakan fakta komputasi dan kesimpulan ilmiah.

Titik kontrol: Jangan melanjutkan ke interpretasi apabila validasi input atau status proses belum memenuhi kriteria.

Kode Dasar Siap Colab

Pembahasan Membuat, membuka, mengganti nama, dan menyimpan notebook ditempatkan dalam konteks praktik Google Colab agar pembaca dapat menghubungkan konsep, prosedur, kode, dan interpretasi secara utuh. Pembuatan notebook baru, pembukaan .ipynb, penyimpanan, salinan, dan berbagi notebook. Hasil yang dituju berupa sel Markdown, sel kode, output, dan metadata.

Sel kode contoh

```
# Penamaan notebook yang konsisten
PROJECT = "kimia-komputasi-dasar"
NOTEBOOK = "01-dasar-google-colab-python.ipynb"

print(f"Proyek : {PROJECT}")
print(f"Notebook : {NOTEBOOK}")

# Gunakan File > Save a copy in Drive untuk membuat salinan.
```

Cara menjalankan: Tempatkan kode dalam satu atau beberapa sel sesuai komentar. Untuk baris %pip atau perintah shell, gunakan runtime Google Colab.

Penjelasan Baris Kode Penting

Potongan kode	Fungsi
# Penamaan notebook yang konsisten	Mendefinisikan input atau parameter yang akan digunakan pada langkah berikutnya.
PROJECT = "kimia-komputasi-dasar"	Mendefinisikan input atau parameter yang akan digunakan pada langkah berikutnya.
NOTEBOOK = "01-dasar-google-colab-python.ipynb"	Menambahkan pemeriksaan untuk mencegah hasil tidak valid diteruskan.
print(f"Proyek : {PROJECT}")	Mencetak status agar pengguna tidak hanya bergantung pada keluaran visual.
print(f"Notebook : {NOTEBOOK}")	Menjalankan operasi utama dan menyimpan hasil pada variabel yang jelas.
# Gunakan File > Save a copy in Drive untuk memb	Mendefinisikan input atau parameter yang akan digunakan pada langkah berikutnya.

Nama variabel dipilih agar maknanya terbaca. Hindari singkatan yang tidak dijelaskan, nilai konstan tanpa satuan, serta penggunaan variabel dari sel lama. Komentar harus menerangkan alasan atau asumsi, bukan sekadar mengulang sintaks.

Validasi dan Interpretasi Hasil

Prinsip kerja yang digunakan adalah memisahkan input, proses, validasi, dan ekspor. Pemisahan ini membuat kesalahan lebih mudah dilacak dan mencegah hasil lama dianggap sebagai hasil dari parameter baru. Setiap langkah perlu meninggalkan bukti yang dapat dibaca kembali: nilai parameter, versi pustaka, status proses, nama berkas, serta pesan kesalahan bila terjadi kegagalan.

Aspek validasi	Kriteria lulus
Input	Tipe, bentuk, satuan, muatan, spin, dan jalur berkas dinyatakan eksplisit.
Lingkungan	Versi Python dan pustaka tercatat; backend CPU/GPU terdeteksi.
Proses	Status eksekusi atau konvergensi diperiksa sebelum hasil dipakai.
Numerik	Tidak ada NaN/inf; ukuran array dan jumlah atom konsisten.
Keluaran	Sel markdown, sel kode, output, dan metadata disimpan dan dapat dibuka kembali.
Interpretasi	Klaim tidak melampaui metode, basis set, atau data yang dihitung.
Interpretasi aman: Sebutkan metode, parameter, satuan, status konvergensi, dan keterbatasan saat menuliskan kesimpulan.	

Kesalahan Umum dan Troubleshooting

Risiko utama pada topik ini adalah state kernel tersembunyi dan urutan eksekusi. Karena itu, notebook harus memuat pemeriksaan eksplisit dan bukan hanya mengandalkan keluaran yang tampak masuk akal. Pembaca dianjurkan menjalankan notebook dari keadaan runtime bersih, membandingkan hasil dengan batas kewajaran, dan menyimpan artefak penting di lokasi persisten.

Gejala	Tindakan perbaikan
Sel dijalankan tidak berurutan	Restart session, lalu gunakan Run all dan periksa variabel prasyarat.
Pustaka tidak ditemukan	Jalankan kembali sel %pip, periksa nama paket, lalu restart bila dependensi inti berubah.
Input tampak benar tetapi gagal	Cetak tipe, shape, satuan, muatan, spin, dan beberapa baris pertama data.
State kernel tersembunyi dan urutan eksekusi	Tambahkan validasi sebelum proses utama dan hentikan eksekusi dengan pesan yang spesifik.
Hasil berbeda antar-runtime	Catat versi runtime/paket, random seed, parameter, dan perangkat yang digunakan.
Aturan troubleshooting: Perbaiki satu hal pada satu waktu, simpan pesan kesalahan lengkap, dan catat perubahan versi atau parameter.	

Latihan Modifikasi dan Refleksi

58. Ubah satu parameter pada contoh Membuat, membuka, mengganti nama, dan menyimpan notebook, jalankan ulang dari runtime bersih, dan jelaskan perubahan keluaran.
59. Tambahkan satu validasi baru yang mencegah state kernel tersembunyi dan urutan eksekusi.
60. Buat berkas ringkasan TXT atau JSON yang memuat input, status proses, waktu, dan hasil utama.
61. Jelaskan bagian mana yang merupakan fakta keluaran program dan bagian mana yang merupakan interpretasi ilmiah.
62. Susun satu kasus uji yang sengaja salah, lalu pastikan notebook menghasilkan pesan kesalahan yang informatif.

Refleksi: Tuliskan apa yang berubah setelah kode dimodifikasi, bukti apa yang menunjukkan keberhasilan, dan bagian mana yang masih memerlukan verifikasi eksternal.

Rubrik

Deskripsi

Skor 4	Kode berjalan dari runtime bersih, validasi lengkap, interpretasi tepat.
Skor 3	Kode berjalan dan hasil tersimpan, tetapi dokumentasi belum lengkap.
Skor 2	Kode hanya berjalan setelah intervensi atau state tertentu.
Skor 1	Kode belum berjalan dan diagnosis kesalahan belum sistematis.

1.2.2 Menjalankan sel dan mengelola runtime

Pada bagian Menjalankan sel dan mengelola runtime, perhatian tidak hanya diarahkan pada keberhasilan menjalankan kode, tetapi juga pada ketepatan asumsi, keterlacakan parameter, dan mutu keluaran. Menjalankan satu sel, seluruh sel, menghentikan proses, memulai ulang runtime, dan menghapus output. Hasil yang dituju berupa informasi lingkungan dan berkas hasil.

Prinsip kerja yang digunakan adalah memisahkan input, proses, validasi, dan ekspor. Pemisahan ini membuat kesalahan lebih mudah dilacak dan mencegah hasil lama dianggap sebagai hasil dari parameter baru. Setiap langkah perlu meninggalkan bukti yang dapat dibaca kembali: nilai parameter, versi pustaka, status proses, nama berkas, serta pesan kesalahan bila terjadi kegagalan.

Tujuan operasional: Pembaca mampu menjelaskan Menjalankan sel dan mengelola runtime, menjalankan contoh minimal, memeriksa keberhasilan, dan menyimpan bukti hasil.

Komponen	Kriteria
Konsep	Menjalankan satu sel, seluruh sel, menghentikan proses, memulai ulang runtime, dan menghapus output.
Input	Parameter dan data dinyatakan eksplisit.
Proses	Langkah dapat diulang dari runtime bersih.

Output	Nilai, berkas, atau visualisasi yang dapat diperiksa.
Batas	Interpretasi mengikuti metode dan konteks.

Landasan Teori dan Batas Penerapan

Pada bagian Menjalankan sel dan mengelola runtime, perhatian tidak hanya diarahkan pada keberhasilan menjalankan kode, tetapi juga pada ketepatan asumsi, keterlacakan parameter, dan mutu keluaran. Menjalankan satu sel, seluruh sel, menghentikan proses, memulai ulang runtime, dan menghapus output. Hasil yang dituju berupa informasi lingkungan dan berkas hasil.

Prinsip kerja yang digunakan adalah memisahkan input, proses, validasi, dan ekspor. Pemisahan ini membuat kesalahan lebih mudah dilacak dan mencegah hasil lama dianggap sebagai hasil dari parameter baru. Setiap langkah perlu meninggalkan bukti yang dapat dibaca kembali: nilai parameter, versi pustaka, status proses, nama berkas, serta pesan kesalahan bila terjadi kegagalan.

Risiko utama pada topik ini adalah runtime terputus, sumber daya berubah, atau berkas sementara hilang. Karena itu, notebook harus memuat pemeriksaan eksplisit dan bukan hanya mengandalkan keluaran yang tampak masuk akal. Pembaca dianjurkan menjalankan notebook dari keadaan runtime bersih, membandingkan hasil dengan batas kewajaran, dan menyimpan artefak penting di lokasi persisten.

Batas penerapan: Contoh pada buku berfungsi sebagai pola kerja. Ukuran molekul, biaya komputasi, versi pustaka, dan perangkat keras dapat mengubah waktu serta strategi yang diperlukan.

Prosedur Bertahap

Pada bagian Menjalankan sel dan mengelola runtime, perhatian tidak hanya diarahkan pada keberhasilan menjalankan kode, tetapi juga pada ketepatan asumsi, keterlacakan parameter, dan mutu keluaran. Menjalankan satu sel, seluruh sel, menghentikan proses, memulai ulang runtime, dan menghapus output. Hasil yang dituju berupa informasi lingkungan dan berkas hasil.

63. Tuliskan tujuan operasional Menjalankan sel dan mengelola runtime dan tentukan keluaran yang harus tersedia pada akhir sel.
64. Tetapkan parameter pada satu sel konfigurasi agar perubahan dapat ditelusuri dan tidak tersebar di banyak tempat.
65. Siapkan runtime Google Colab dengan input yang eksplisit; hindari nilai tersembunyi yang hanya tersimpan dalam state kernel.

66. Jalankan pemeriksaan prasyarat, termasuk keberadaan pustaka, bentuk data, tipe nilai, serta konsistensi satuan.
67. Eksekusi proses utama, tangkap status konvergensi atau keberhasilan, dan hentikan alur apabila syarat minimum tidak terpenuhi.
68. Simpan informasi lingkungan dan berkas hasil dengan nama berkas informatif, kemudian tulis interpretasi yang membedakan fakta komputasi dan kesimpulan ilmiah.

Titik kontrol: Jangan melanjutkan ke interpretasi apabila validasi input atau status proses belum memenuhi kriteria.

Kode Dasar Siap Colab

Pembahasan Menjalankan sel dan mengelola runtime ditempatkan dalam konteks praktik Google Colab agar pembaca dapat menghubungkan konsep, prosedur, kode, dan interpretasi secara utuh. Menjalankan satu sel, seluruh sel, menghentikan proses, memulai ulang runtime, dan menghapus output. Hasil yang dituju berupa informasi lingkungan dan berkas hasil.

Sel kode contoh

```
from time import perf_counter, sleep

awal = perf_counter()
for tahap in range(1, 6):
    print(f"Menjalankan tahap {tahap}/5")
    sleep(0.2)
akhir = perf_counter()
print(f"Selesai dalam {akhir-awal:.2f} s")
```

Cara menjalankan: Tempatkan kode dalam satu atau beberapa sel sesuai komentar. Untuk baris %pip atau perintah shell, gunakan runtime Google Colab.

Penjelasan Baris Kode Penting

Potongan kode	Fungsi
<code>from time import perf_counter, sleep</code>	Mencetak status agar pengguna tidak hanya bergantung pada keluaran visual.
<code>awal = perf_counter()</code>	Mendefinisikan input atau parameter yang akan digunakan pada langkah berikutnya.

<code>for tahap in range(1, 6):</code>	Mendefinisikan input atau parameter yang akan digunakan pada langkah berikutnya.
<code>print(f'Menjalankan tahap {tahap}/5')</code>	Menambahkan pemeriksaan untuk mencegah hasil tidak valid diteruskan.
<code>sleep(0.2)</code>	Menjalankan operasi utama dan menyimpan hasil pada variabel yang jelas.
<code>akhir = perf_counter()</code>	Menjalankan operasi utama dan menyimpan hasil pada variabel yang jelas.
<code>print(f'Selesai dalam {akhir-awal:.2f} s')</code>	Mendefinisikan input atau parameter yang akan digunakan pada langkah berikutnya.

Nama variabel dipilih agar maknanya terbaca. Hindari singkatan yang tidak dijelaskan, nilai konstan tanpa satuan, serta penggunaan variabel dari sel lama. Komentar harus menerangkan alasan atau asumsi, bukan sekadar mengulang sintaks.

Validasi dan Interpretasi Hasil

Prinsip kerja yang digunakan adalah memisahkan input, proses, validasi, dan ekspor. Pemisahan ini membuat kesalahan lebih mudah dilacak dan mencegah hasil lama dianggap sebagai hasil dari parameter baru. Setiap langkah perlu meninggalkan bukti yang dapat dibaca kembali: nilai parameter, versi pustaka, status proses, nama berkas, serta pesan kesalahan bila terjadi kegagalan.

Aspek validasi	Kriteria lulus
Input	Tipe, bentuk, satuan, muatan, spin, dan jalur berkas dinyatakan eksplisit.
Lingkungan	Versi Python dan pustaka tercatat; backend CPU/GPU terdeteksi.
Proses	Status eksekusi atau konvergensi diperiksa sebelum hasil dipakai.
Numerik	Tidak ada NaN/inf; ukuran array dan jumlah atom konsisten.

Keluaran	Informasi lingkungan dan berkas hasil disimpan dan dapat dibuka kembali.
Interpretasi	Klaim tidak melampaui metode, basis set, atau data yang dihitung.
Interpretasi aman: Sebutkan metode, parameter, satuan, status konvergensi, dan keterbatasan saat menuliskan kesimpulan.	

Kesalahan Umum dan Troubleshooting

Risiko utama pada topik ini adalah runtime terputus, sumber daya berubah, atau berkas sementara hilang. Karena itu, notebook harus memuat pemeriksaan eksplisit dan bukan hanya mengandalkan keluaran yang tampak masuk akal. Pembaca dianjurkan menjalankan notebook dari keadaan runtime bersih, membandingkan hasil dengan batas kewajaran, dan menyimpan artefak penting di lokasi persisten.

Gejala	Tindakan perbaikan
Sel dijalankan tidak berurutan	Restart session, lalu gunakan Run all dan periksa variabel prasyarat.
Pustaka tidak ditemukan	Jalankan kembali sel %pip, periksa nama paket, lalu restart bila dependensi inti berubah.
Input tampak benar tetapi gagal	Cetak tipe, shape, satuan, muatan, spin, dan beberapa baris pertama data.
Runtime terputus, sumber daya berubah, atau berkas sementara hilang	Tambahkan validasi sebelum proses utama dan hentikan eksekusi dengan pesan yang spesifik.
Hasil berbeda antar-runtime	Catat versi runtime/paket, random seed, parameter, dan perangkat yang digunakan.
Aturan troubleshooting: Perbaiki satu hal pada satu waktu, simpan pesan kesalahan lengkap, dan catat perubahan versi atau parameter.	

Latihan Modifikasi dan Refleksi

69. Ubah satu parameter pada contoh Menjalankan sel dan mengelola runtime, jalankan ulang dari runtime bersih, dan jelaskan perubahan keluaran.

70. Tambahkan satu validasi baru yang mencegah runtime terputus, sumber daya berubah, atau berkas sementara hilang.
71. Buat berkas ringkasan TXT atau JSON yang memuat input, status proses, waktu, dan hasil utama.
72. Jelaskan bagian mana yang merupakan fakta keluaran program dan bagian mana yang merupakan interpretasi ilmiah.
73. Susun satu kasus uji yang sengaja salah, lalu pastikan notebook menghasilkan pesan kesalahan yang informatif.

Refleksi: Tuliskan apa yang berubah setelah kode dimodifikasi, bukti apa yang menunjukkan keberhasilan, dan bagian mana yang masih memerlukan verifikasi eksternal.

Rubrik	Deskripsi
Skor 4	Kode berjalan dari runtime bersih, validasi lengkap, interpretasi tepat.
Skor 3	Kode berjalan dan hasil tersimpan, tetapi dokumentasi belum lengkap.
Skor 2	Kode hanya berjalan setelah intervensi atau state tertentu.
Skor 1	Kode belum berjalan dan diagnosis kesalahan belum sistematis.

1.2.3 Mengunggah dan mengunduh berkas

Pembahasan Mengunggah dan mengunduh berkas ditempatkan dalam konteks praktik Google Colab agar pembaca dapat menghubungkan konsep, prosedur, kode, dan interpretasi secara utuh. Mengunggah data, menghubungkan Google Drive, mengunduh hasil, dan memahami direktori `/content`. Hasil yang dituju berupa hasil yang dapat diperiksa dan disimpan.

Prinsip kerja yang digunakan adalah memisahkan input, proses, validasi, dan ekspor. Pemisahan ini membuat kesalahan lebih mudah dilacak dan mencegah hasil lama dianggap sebagai hasil dari parameter baru. Setiap langkah perlu meninggalkan bukti yang dapat dibaca kembali: nilai parameter, versi pustaka, status proses, nama berkas, serta pesan kesalahan bila terjadi kegagalan.

Tujuan operasional: Pembaca mampu menjelaskan Mengunggah dan mengunduh berkas, menjalankan contoh minimal, memeriksa keberhasilan, dan menyimpan bukti hasil.

Komponen	Kriteria
Konsep	Mengunggah data, menghubungkan Google Drive, mengunduh hasil, dan memahami direktori /content.
Input	Parameter dan data dinyatakan eksplisit.
Proses	Langkah dapat diulang dari runtime bersih.
Output	Nilai, berkas, atau visualisasi yang dapat diperiksa.
Batas	Interpretasi mengikuti metode dan konteks.

Landasan Teori dan Batas Penerapan

Mengunggah dan mengunduh berkas merupakan simpul penting dalam alur kerja notebook ilmiah karena menentukan bagaimana prosedur komputasi disiapkan, dijalankan, dan diperiksa. Mengunggah data, menghubungkan Google Drive, mengunduh hasil, dan memahami direktori /content. Hasil yang dituju berupa hasil yang dapat diperiksa dan disimpan.

Prinsip kerja yang digunakan adalah memisahkan input, proses, validasi, dan ekspor. Pemisahan ini membuat kesalahan lebih mudah dilacak dan mencegah hasil lama dianggap sebagai hasil dari parameter baru. Setiap langkah perlu meninggalkan bukti yang dapat dibaca kembali: nilai parameter, versi pustaka, status proses, nama berkas, serta pesan kesalahan bila terjadi kegagalan.

Risiko utama pada topik ini adalah parameter tidak terdokumentasi dan keluaran tidak tervalidasi. Karena itu, notebook harus memuat pemeriksaan eksplisit dan bukan hanya mengandalkan keluaran yang tampak masuk akal. Pembaca dianjurkan menjalankan notebook dari keadaan runtime bersih, membandingkan hasil dengan batas kewajaran, dan menyimpan artefak penting di lokasi persisten.

Batas penerapan: Contoh pada buku berfungsi sebagai pola kerja. Ukuran molekul, biaya komputasi, versi pustaka, dan perangkat keras dapat mengubah waktu serta strategi yang diperlukan.

Prosedur Bertahap

Pada bagian Mengunggah dan mengunduh berkas, perhatian tidak hanya diarahkan pada keberhasilan menjalankan kode, tetapi juga pada ketepatan asumsi, keterlacakan parameter, dan mutu keluaran. Mengunggah data, menghubungkan Google Drive, mengunduh hasil, dan memahami direktori /content. Hasil yang dituju berupa hasil yang dapat diperiksa dan disimpan.

74. Tuliskan tujuan operasional Mengunggah dan mengunduh berkas dan tentukan keluaran yang harus tersedia pada akhir sel.
75. Tetapkan parameter pada satu sel konfigurasi agar perubahan dapat ditelusuri dan tidak tersebar di banyak tempat.
76. Siapkan prosedur komputasi dengan input yang eksplisit; hindari nilai tersembunyi yang hanya tersimpan dalam state kernel.
77. Jalankan pemeriksaan prasyarat, termasuk keberadaan pustaka, bentuk data, tipe nilai, serta konsistensi satuan.
78. Eksekusi proses utama, tangkap status konvergensi atau keberhasilan, dan hentikan alur apabila syarat minimum tidak terpenuhi.
79. Simpan hasil yang dapat diperiksa dan disimpan dengan nama berkas informatif, kemudian tulis interpretasi yang membedakan fakta komputasi dan kesimpulan ilmiah.

Titik kontrol: Jangan melanjutkan ke interpretasi apabila validasi input atau status proses belum memenuhi kriteria.

Kode Dasar Siap Colab

Pada bagian Mengunggah dan mengunduh berkas, perhatian tidak hanya diarahkan pada keberhasilan menjalankan kode, tetapi juga pada ketepatan asumsi, keterlacakan parameter, dan mutu keluaran. Mengunggah data, menghubungkan Google Drive, mengunduh hasil, dan memahami direktori /content. Hasil yang dituju berupa hasil yang dapat diperiksa dan disimpan.

Sel kode contoh

```
from pathlib import Path
from google.colab import files

unggah = files.upload()
for nama in unggahan:
    print("Berkas diterima:", nama)

Path("hasil.txt").write_text("Contoh hasil Colab", encoding="utf-8")
files.download("hasil.txt")
```

Cara menjalankan: Tempatkan kode dalam satu atau beberapa sel sesuai komentar. Untuk baris %pip atau perintah shell, gunakan runtime Google Colab.

Penjelasan Baris Kode Penting

Potongan kode	Fungsi
<code>from pathlib import Path</code>	Mencetak status agar pengguna tidak hanya bergantung pada keluaran visual.
<code>from google.colab import files</code>	Mencetak status agar pengguna tidak hanya bergantung pada keluaran visual.
<code>unggahan = files.upload()</code>	Menjalankan operasi utama dan menyimpan hasil pada variabel yang jelas.
<code>for nama in unggahan:</code>	Menjalankan operasi utama dan menyimpan hasil pada variabel yang jelas.
<code>print("Berkas diterima:", nama)</code>	Mencetak status agar pengguna tidak hanya bergantung pada keluaran visual.
<code>Path("hasil.txt").write_text("Contoh hasil Colab</code>	Mendefinisikan input atau parameter yang akan digunakan pada langkah berikutnya.
<code>files.download("hasil.txt")</code>	Mencetak status agar pengguna tidak hanya bergantung pada keluaran visual.

Nama variabel dipilih agar maknanya terbaca. Hindari singkatan yang tidak dijelaskan, nilai konstan tanpa satuan, serta penggunaan variabel dari sel lama. Komentar harus menerangkan alasan atau asumsi, bukan sekadar mengulang sintaks.

Validasi dan Interpretasi Hasil

Prinsip kerja yang digunakan adalah memisahkan input, proses, validasi, dan ekspor. Pemisahan ini membuat kesalahan lebih mudah dilacak dan mencegah hasil lama dianggap sebagai hasil dari parameter baru. Setiap langkah perlu meninggalkan bukti yang dapat dibaca kembali: nilai parameter, versi pustaka, status proses, nama berkas, serta pesan kesalahan bila terjadi kegagalan.

Aspek validasi	Kriteria lulus
Input	Tipe, bentuk, satuan, muatan, spin, dan jalur berkas dinyatakan eksplisit.
Lingkungan	Versi Python dan pustaka tercatat; backend CPU/GPU terdeteksi.
Proses	Status eksekusi atau konvergensi diperiksa sebelum hasil dipakai.
Numerik	Tidak ada NaN/inf; ukuran array dan jumlah atom konsisten.
Keluaran	Hasil yang dapat diperiksa dan disimpan disimpan dan dapat dibuka kembali.
Interpretasi	Klaim tidak melampaui metode, basis set, atau data yang dihitung.
Interpretasi aman: Sebutkan metode, parameter, satuan, status konvergensi, dan keterbatasan saat menuliskan kesimpulan.	

Kesalahan Umum dan Troubleshooting

Risiko utama pada topik ini adalah parameter tidak terdokumentasi dan keluaran tidak tervalidasi. Karena itu, notebook harus memuat pemeriksaan eksplisit dan bukan hanya mengandalkan keluaran yang tampak masuk akal. Pembaca dianjurkan menjalankan notebook dari keadaan runtime bersih, membandingkan hasil dengan batas kewajaran, dan menyimpan artefak penting di lokasi persisten.

Gejala	Tindakan perbaikan
Sel dijalankan tidak berurutan	Restart session, lalu gunakan Run all dan periksa variabel prasyarat.
Pustaka tidak ditemukan	Jalankan kembali sel %pip, periksa nama paket, lalu restart bila dependensi inti berubah.

Input tampak benar tetapi gagal	Cetak tipe, shape, satuan, muatan, spin, dan beberapa baris pertama data.
Parameter tidak terdokumentasi dan keluaran tidak tervalidasi	Tambahkan validasi sebelum proses utama dan hentikan eksekusi dengan pesan yang spesifik.
Hasil berbeda antar-runtime	Catat versi runtime/paket, random seed, parameter, dan perangkat yang digunakan.
Aturan troubleshooting: Perbaiki satu hal pada satu waktu, simpan pesan kesalahan lengkap, dan catat perubahan versi atau parameter.	

Latihan Modifikasi dan Refleksi

80. Ubah satu parameter pada contoh Mengunggah dan mengunduh berkas, jalankan ulang dari runtime bersih, dan jelaskan perubahan keluaran.
81. Tambahkan satu validasi baru yang mencegah parameter tidak terdokumentasi dan keluaran tidak tervalidasi.
82. Buat berkas ringkasan TXT atau JSON yang memuat input, status proses, waktu, dan hasil utama.
83. Jelaskan bagian mana yang merupakan fakta keluaran program dan bagian mana yang merupakan interpretasi ilmiah.
84. Susun satu kasus uji yang sengaja salah, lalu pastikan notebook menghasilkan pesan kesalahan yang informatif.

Refleksi: Tuliskan apa yang berubah setelah kode dimodifikasi, bukti apa yang menunjukkan keberhasilan, dan bagian mana yang masih memerlukan verifikasi eksternal.

Rubrik	Deskripsi
Skor 4	Kode berjalan dari runtime bersih, validasi lengkap, interpretasi tepat.
Skor 3	Kode berjalan dan hasil tersimpan, tetapi dokumentasi belum lengkap.
Skor 2	Kode hanya berjalan setelah intervensi atau state tertentu.

Skor 1	Kode belum berjalan dan diagnosis kesalahan belum sistematis.

1.3 Dasar Python untuk Komputasi Ilmiah

Dasar Python untuk Komputasi Ilmiah mengorganisasi tiga komponen yang saling bergantung. Pembahasan bergerak dari konsep, menuju prosedur, kemudian menghasilkan artefak yang dapat divalidasi dan disimpan.

Sub-subbab	Fokus
1.3.1 Variabel, tipe data, operator, dan struktur kontrol	String, integer, float, Boolean, list, tuple, dictionary, percabangan, dan perulangan.
1.3.2 Fungsi, modul, dan penanganan kesalahan	Pembuatan fungsi, parameter, nilai balik, impor modul, try-except, dan pesan kesalahan informatif.
1.3.3 NumPy, Pandas, dan Matplotlib	Array numerik, tabel data, statistik sederhana, grafik garis, diagram pencar, dan penyimpanan gambar.
Produk belajar: Satu notebook terstruktur, satu berkas keluaran, satu catatan validasi, dan satu interpretasi singkat yang menyatakan batas hasil.	

Rencana Praktik dan Produk Belajar

85. Pelajari Variabel, tipe data, operator, dan struktur kontrol dan jalankan kode dasar yang menyertainya.
86. Pelajari Fungsi, modul, dan penanganan kesalahan dan jalankan kode dasar yang menyertainya.
87. Pelajari NumPy, Pandas, dan Matplotlib dan jalankan kode dasar yang menyertainya.

Komponen	Kriteria
Waktu	60-120 menit, bergantung pada komputasi.
Input	Notebook baru atau salinan notebook bab.
Produk	Kode, output, berkas hasil, dan refleksi.

Uji	Restart session lalu Run all.
Dokumentasi	Versi, parameter, status, dan waktu eksekusi.
Kriteria selesai: Semua sel dapat dijalankan berurutan, pesan kesalahan ditangani, dan hasil utama tersimpan di lokasi persisten.	

1.3.1 Variabel, tipe data, operator, dan struktur kontrol

Variabel, tipe data, operator, dan struktur kontrol merupakan simpul penting dalam alur kerja notebook ilmiah karena menentukan bagaimana prosedur komputasi disiapkan, dijalankan, dan diperiksa. String, integer, float, Boolean, list, tuple, dictionary, percabangan, dan perulangan. Hasil yang dituju berupa hasil yang dapat diperiksa dan disimpan.

Prinsip kerja yang digunakan adalah memisahkan input, proses, validasi, dan ekspor. Pemisahan ini membuat kesalahan lebih mudah dilacak dan mencegah hasil lama dianggap sebagai hasil dari parameter baru. Setiap langkah perlu meninggalkan bukti yang dapat dibaca kembali: nilai parameter, versi pustaka, status proses, nama berkas, serta pesan kesalahan bila terjadi kegagalan.

Tujuan operasional: Pembaca mampu menjelaskan Variabel, tipe data, operator, dan struktur kontrol, menjalankan contoh minimal, memeriksa keberhasilan, dan menyimpan bukti hasil.

Komponen	Kriteria
Konsep	String, integer, float, Boolean, list, tuple, dictionary, percabangan, dan perulangan.
Input	Parameter dan data dinyatakan eksplisit.
Proses	Langkah dapat diulang dari runtime bersih.
Output	Nilai, berkas, atau visualisasi yang dapat diperiksa.
Batas	Interpretasi mengikuti metode dan konteks.

Landasan Teori dan Batas Penerapan

Pada bagian Variabel, tipe data, operator, dan struktur kontrol, perhatian tidak hanya diarahkan pada keberhasilan menjalankan kode, tetapi juga pada ketepatan asumsi, keterlacakan parameter, dan mutu keluaran. String, integer, float, Boolean, list, tuple, dictionary, percabangan, dan perulangan. Hasil yang dituju berupa hasil yang dapat diperiksa dan disimpan.

Prinsip kerja yang digunakan adalah memisahkan input, proses, validasi, dan ekspor. Pemisahan ini membuat kesalahan lebih mudah dilacak dan mencegah hasil lama dianggap sebagai hasil dari parameter baru. Setiap langkah perlu meninggalkan bukti yang dapat dibaca kembali: nilai parameter, versi pustaka, status proses, nama berkas, serta pesan kesalahan bila terjadi kegagalan.

Risiko utama pada topik ini adalah parameter tidak terdokumentasi dan keluaran tidak tervalidasi. Karena itu, notebook harus memuat pemeriksaan eksplisit dan bukan hanya mengandalkan keluaran yang tampak masuk akal. Pembaca dianjurkan menjalankan notebook dari keadaan runtime bersih, membandingkan hasil dengan batas kewajaran, dan menyimpan artefak penting di lokasi persisten.

Batas penerapan: Contoh pada buku berfungsi sebagai pola kerja. Ukuran molekul, biaya komputasi, versi pustaka, dan perangkat keras dapat mengubah waktu serta strategi yang diperlukan.

Prosedur Bertahap

Pada bagian Variabel, tipe data, operator, dan struktur kontrol, perhatian tidak hanya diarahkan pada keberhasilan menjalankan kode, tetapi juga pada ketepatan asumsi, keterlacakan parameter, dan mutu keluaran. String, integer, float, Boolean, list, tuple, dictionary, percabangan, dan perulangan. Hasil yang dituju berupa hasil yang dapat diperiksa dan disimpan.

88. Tuliskan tujuan operasional Variabel, tipe data, operator, dan struktur kontrol dan tentukan keluaran yang harus tersedia pada akhir sel.
89. Tetapkan parameter pada satu sel konfigurasi agar perubahan dapat ditelusuri dan tidak tersebar di banyak tempat.
90. Siapkan prosedur komputasi dengan input yang eksplisit; hindari nilai tersembunyi yang hanya tersimpan dalam state kernel.
91. Jalankan pemeriksaan prasyarat, termasuk keberadaan pustaka, bentuk data, tipe nilai, serta konsistensi satuan.
92. Eksekusi proses utama, tangkap status konvergensi atau keberhasilan, dan hentikan alur apabila syarat minimum tidak terpenuhi.
93. Simpan hasil yang dapat diperiksa dan disimpan dengan nama berkas informatif, kemudian tulis interpretasi yang membedakan fakta komputasi dan kesimpulan ilmiah.

Titik kontrol: Jangan melanjutkan ke interpretasi apabila validasi input atau status proses belum memenuhi kriteria.

Kode Dasar Siap Colab

Pembahasan Variabel, tipe data, operator, dan struktur kontrol ditempatkan dalam konteks praktik Google Colab agar pembaca dapat menghubungkan konsep, prosedur, kode, dan interpretasi secara utuh. String, integer, float, Boolean, list, tuple, dictionary, percabangan, dan perulangan. Hasil yang dituju berupa hasil yang dapat diperiksa dan disimpan.

Sel kode contoh

```
nama = "H2O"
muatan = 0
energi = -76.4
konvergen = True
atom = ["O", "H", "H"]
parameter = {"metode": "B3LYP", "basis": "def2-TZVP"}

if konvergen:
    for simbol in atom:
        print(simbol, parameter["metode"])
```

Cara menjalankan: Tempatkan kode dalam satu atau beberapa sel sesuai komentar. Untuk baris %pip atau perintah shell, gunakan runtime Google Colab.

Penjelasan Baris Kode Penting

Potongan kode	Fungsi
nama = "H2O"	Menjalankan operasi utama dan menyimpan hasil pada variabel yang jelas.
muatan = 0	Menambahkan pemeriksaan untuk mencegah hasil tidak valid diteruskan.
energi = -76.4	Mendefinisikan input atau parameter yang akan digunakan pada langkah berikutnya.
konvergen = True	Menambahkan pemeriksaan untuk mencegah hasil tidak valid diteruskan.

<code>atom = ["O", "H", "H"]</code>	Mencetak status agar pengguna tidak hanya bergantung pada keluaran visual.
<code>parameter = {"metode": "B3LYP", "basis": "def2-T</code>	Menjalankan operasi utama dan menyimpan hasil pada variabel yang jelas.
<code>if konvergen:</code>	Mencetak status agar pengguna tidak hanya bergantung pada keluaran visual.
<code>for simbol in atom:</code>	Mencetak status agar pengguna tidak hanya bergantung pada keluaran visual.

Nama variabel dipilih agar maknanya terbaca. Hindari singkatan yang tidak dijelaskan, nilai konstan tanpa satuan, serta penggunaan variabel dari sel lama. Komentar harus menerangkan alasan atau asumsi, bukan sekadar mengulang sintaks.

Validasi dan Interpretasi Hasil

Prinsip kerja yang digunakan adalah memisahkan input, proses, validasi, dan ekspor. Pemisahan ini membuat kesalahan lebih mudah dilacak dan mencegah hasil lama dianggap sebagai hasil dari parameter baru. Setiap langkah perlu meninggalkan bukti yang dapat dibaca kembali: nilai parameter, versi pustaka, status proses, nama berkas, serta pesan kesalahan bila terjadi kegagalan.

Aspek validasi	Kriteria lulus
Input	Tipe, bentuk, satuan, muatan, spin, dan jalur berkas dinyatakan eksplisit.
Lingkungan	Versi Python dan pustaka tercatat; backend CPU/GPU terdeteksi.
Proses	Status eksekusi atau konvergensi diperiksa sebelum hasil dipakai.
Numerik	Tidak ada NaN/inf; ukuran array dan jumlah atom konsisten.
Keluaran	Hasil yang dapat diperiksa dan disimpan disimpan dan dapat dibuka kembali.

Interpretasi

Klaim tidak melampaui metode, basis set, atau data yang dihitung.

Interpretasi aman: Sebutkan metode, parameter, satuan, status konvergensi, dan keterbatasan saat menuliskan kesimpulan.

Kesalahan Umum dan Troubleshooting

Risiko utama pada topik ini adalah parameter tidak terdokumentasi dan keluaran tidak tervalidasi. Karena itu, notebook harus memuat pemeriksaan eksplisit dan bukan hanya mengandalkan keluaran yang tampak masuk akal. Pembaca dianjurkan menjalankan notebook dari keadaan runtime bersih, membandingkan hasil dengan batas kewajaran, dan menyimpan artefak penting di lokasi persisten.

Gejala	Tindakan perbaikan
Sel dijalankan tidak berurutan	Restart session, lalu gunakan Run all dan periksa variabel prasyarat.
Pustaka tidak ditemukan	Jalankan kembali sel %pip, periksa nama paket, lalu restart bila dependensi inti berubah.
Input tampak benar tetapi gagal	Cetak tipe, shape, satuan, muatan, spin, dan beberapa baris pertama data.
Parameter tidak terdokumentasi dan keluaran tidak tervalidasi	Tambahkan validasi sebelum proses utama dan hentikan eksekusi dengan pesan yang spesifik.
Hasil berbeda antar-runtime	Catat versi runtime/paket, random seed, parameter, dan perangkat yang digunakan.
Aturan troubleshooting: Perbaiki satu hal pada satu waktu, simpan pesan kesalahan lengkap, dan catat perubahan versi atau parameter.	

Latihan Modifikasi dan Refleksi

- Ubah satu parameter pada contoh Variabel, tipe data, operator, dan struktur kontrol, jalankan ulang dari runtime bersih, dan jelaskan perubahan keluaran.
- Tambahkan satu validasi baru yang mencegah parameter tidak terdokumentasi dan keluaran tidak tervalidasi.

96. Buat berkas ringkasan TXT atau JSON yang memuat input, status proses, waktu, dan hasil utama.
97. Jelaskan bagian mana yang merupakan fakta keluaran program dan bagian mana yang merupakan interpretasi ilmiah.
98. Susun satu kasus uji yang sengaja salah, lalu pastikan notebook menghasilkan pesan kesalahan yang informatif.

Refleksi: Tuliskan apa yang berubah setelah kode dimodifikasi, bukti apa yang menunjukkan keberhasilan, dan bagian mana yang masih memerlukan verifikasi eksternal.

Rubrik	Deskripsi
Skor 4	Kode berjalan dari runtime bersih, validasi lengkap, interpretasi tepat.
Skor 3	Kode berjalan dan hasil tersimpan, tetapi dokumentasi belum lengkap.
Skor 2	Kode hanya berjalan setelah intervensi atau state tertentu.
Skor 1	Kode belum berjalan dan diagnosis kesalahan belum sistematis.

1.3.2 Fungsi, modul, dan penanganan kesalahan

Pada bagian Fungsi, modul, dan penanganan kesalahan, perhatian tidak hanya diarahkan pada keberhasilan menjalankan kode, tetapi juga pada ketepatan asumsi, keterlacakan parameter, dan mutu keluaran. Pembuatan fungsi, parameter, nilai balik, impor modul, try-except, dan pesan kesalahan informatif. Hasil yang dituju berupa hasil yang dapat diperiksa dan disimpan.

Prinsip kerja yang digunakan adalah memisahkan input, proses, validasi, dan ekspor. Pemisahan ini membuat kesalahan lebih mudah dilacak dan mencegah hasil lama dianggap sebagai hasil dari parameter baru. Setiap langkah perlu meninggalkan bukti yang dapat dibaca kembali: nilai parameter, versi pustaka, status proses, nama berkas, serta pesan kesalahan bila terjadi kegagalan.

Tujuan operasional: Pembaca mampu menjelaskan Fungsi, modul, dan penanganan kesalahan, menjalankan contoh minimal, memeriksa keberhasilan, dan menyimpan bukti hasil.

Komponen	Kriteria
Konsep	Pembuatan fungsi, parameter, nilai balik, impor modul, try-except, dan pesan kesalahan informatif.
Input	Parameter dan data dinyatakan eksplisit.
Proses	Langkah dapat diulang dari runtime bersih.
Output	Nilai, berkas, atau visualisasi yang dapat diperiksa.
Batas	Interpretasi mengikuti metode dan konteks.

Landasan Teori dan Batas Penerapan

Pembahasan Fungsi, modul, dan penanganan kesalahan ditempatkan dalam konteks praktik Google Colab agar pembaca dapat menghubungkan konsep, prosedur, kode, dan interpretasi secara utuh. Pembuatan fungsi, parameter, nilai balik, impor modul, try-except, dan pesan kesalahan informatif. Hasil yang dituju berupa hasil yang dapat diperiksa dan disimpan.

Prinsip kerja yang digunakan adalah memisahkan input, proses, validasi, dan ekspor. Pemisahan ini membuat kesalahan lebih mudah dilacak dan mencegah hasil lama dianggap sebagai hasil dari parameter baru. Setiap langkah perlu meninggalkan bukti yang dapat dibaca kembali: nilai parameter, versi pustaka, status proses, nama berkas, serta pesan kesalahan bila terjadi kegagalan.

Risiko utama pada topik ini adalah parameter tidak terdokumentasi dan keluaran tidak tervalidasi. Karena itu, notebook harus memuat pemeriksaan eksplisit dan bukan hanya mengandalkan keluaran yang tampak masuk akal. Pembaca dianjurkan menjalankan notebook dari keadaan runtime bersih, membandingkan hasil dengan batas kewajaran, dan menyimpan artefak penting di lokasi persisten.

Batas penerapan: Contoh pada buku berfungsi sebagai pola kerja. Ukuran molekul, biaya komputasi, versi pustaka, dan perangkat keras dapat mengubah waktu serta strategi yang diperlukan.

Prosedur Bertahap

Fungsi, modul, dan penanganan kesalahan merupakan simpul penting dalam alur kerja notebook ilmiah karena menentukan bagaimana prosedur komputasi disiapkan,

dijalankan, dan diperiksa. Pembuatan fungsi, parameter, nilai balik, impor modul, try-except, dan pesan kesalahan informatif. Hasil yang dituju berupa hasil yang dapat diperiksa dan disimpan.

99. Tuliskan tujuan operasional Fungsi, modul, dan penanganan kesalahan dan tentukan keluaran yang harus tersedia pada akhir sel.
100. Tetapkan parameter pada satu sel konfigurasi agar perubahan dapat ditelusuri dan tidak tersebar di banyak tempat.
101. Siapkan prosedur komputasi dengan input yang eksplisit; hindari nilai tersembunyi yang hanya tersimpan dalam state kernel.
102. Jalankan pemeriksaan prasyarat, termasuk keberadaan pustaka, bentuk data, tipe nilai, serta konsistensi satuan.
103. Eksekusi proses utama, tangkap status konvergensi atau keberhasilan, dan hentikan alur apabila syarat minimum tidak terpenuhi.
104. Simpan hasil yang dapat diperiksa dan disimpan dengan nama berkas informatif, kemudian tulis interpretasi yang membedakan fakta komputasi dan kesimpulan ilmiah.

Titik kontrol: Jangan melanjutkan ke interpretasi apabila validasi input atau status proses belum memenuhi kriteria.

Kode Dasar Siap Colab

Pada bagian Fungsi, modul, dan penanganan kesalahan, perhatian tidak hanya diarahkan pada keberhasilan menjalankan kode, tetapi juga pada ketepatan asumsi, keterlacakan parameter, dan mutu keluaran. Pembuatan fungsi, parameter, nilai balik, impor modul, try-except, dan pesan kesalahan informatif. Hasil yang dituju berupa hasil yang dapat diperiksa dan disimpan.

Sel kode contoh

```
from math import sqrt

def jarak_3d(a, b):
    if len(a) != 3 or len(b) != 3:
        raise ValueError("Koordinat harus memiliki tiga komponen")
    return sqrt(sum((x-y)**2 for x, y in zip(a, b)))

try:
    print(jarak_3d((0,0,0), (0,0,0.74)))
except ValueError as exc:
    print("Input tidak valid:", exc)
```

Cara menjalankan: Tempatkan kode dalam satu atau beberapa sel sesuai komentar. Untuk baris %pip atau perintah shell, gunakan runtime Google Colab.

Penjelasan Baris Kode Penting

Potongan kode	Fungsi
<code>from math import sqrt</code>	Mencetak status agar pengguna tidak hanya bergantung pada keluaran visual.
<code>def jarak_3d(a, b):</code>	Mencetak status agar pengguna tidak hanya bergantung pada keluaran visual.
<code>if len(a) != 3 or len(b) != 3:</code>	Mencetak status agar pengguna tidak hanya bergantung pada keluaran visual.
<code>raise ValueError("Koordinat harus memiliki</code>	Mendefinisikan input atau parameter yang akan digunakan pada langkah berikutnya.
<code>return sqrt(sum((x-y)**2 for x, y in zip(a,</code>	Mencetak status agar pengguna tidak hanya bergantung pada keluaran visual.
<code>try:</code>	Menambahkan pemeriksaan untuk mencegah hasil tidak valid diteruskan.
<code>print(jarak_3d((0,0,0), (0,0,0.74)))</code>	Menambahkan pemeriksaan untuk mencegah hasil tidak valid diteruskan.
<code>except ValueError as exc:</code>	Mencetak status agar pengguna tidak hanya bergantung pada keluaran visual.

Nama variabel dipilih agar maknanya terbaca. Hindari singkatan yang tidak dijelaskan, nilai konstan tanpa satuan, serta penggunaan variabel dari sel lama. Komentar harus menerangkan alasan atau asumsi, bukan sekadar mengulang sintaks.

Validasi dan Interpretasi Hasil

Prinsip kerja yang digunakan adalah memisahkan input, proses, validasi, dan ekspor. Pemisahan ini membuat kesalahan lebih mudah dilacak dan mencegah hasil lama dianggap sebagai hasil dari parameter baru. Setiap langkah perlu meninggalkan bukti yang dapat dibaca kembali: nilai parameter, versi pustaka, status proses, nama berkas, serta pesan kesalahan bila terjadi kegagalan.

Aspek validasi**Kriteria lulus**

Input	Tipe, bentuk, satuan, muatan, spin, dan jalur berkas dinyatakan eksplisit.
Lingkungan	Versi Python dan pustaka tercatat; backend CPU/GPU terdeteksi.
Proses	Status eksekusi atau konvergensi diperiksa sebelum hasil dipakai.
Numerik	Tidak ada NaN/inf; ukuran array dan jumlah atom konsisten.
Keluaran	Hasil yang dapat diperiksa dan disimpan disimpan dan dapat dibuka kembali.
Interpretasi	Klaim tidak melampaui metode, basis set, atau data yang dihitung.
Interpretasi aman: Sebutkan metode, parameter, satuan, status konvergensi, dan keterbatasan saat menuliskan kesimpulan.	

Kesalahan Umum dan Troubleshooting

Risiko utama pada topik ini adalah parameter tidak terdokumentasi dan keluaran tidak tervalidasi. Karena itu, notebook harus memuat pemeriksaan eksplisit dan bukan hanya mengandalkan keluaran yang tampak masuk akal. Pembaca dianjurkan menjalankan notebook dari keadaan runtime bersih, membandingkan hasil dengan batas kewajaran, dan menyimpan artefak penting di lokasi persisten.

Gejala	Tindakan perbaikan
Sel dijalankan tidak berurutan	Restart session, lalu gunakan Run all dan periksa variabel prasyarat.
Pustaka tidak ditemukan	Jalankan kembali sel %pip, periksa nama paket, lalu restart bila dependensi inti berubah.
Input tampak benar tetapi gagal	Cetak tipe, shape, satuan, muatan, spin, dan beberapa baris pertama data.

Parameter tidak terdokumentasi dan keluaran tidak tervalidasi	Tambahkan validasi sebelum proses utama dan hentikan eksekusi dengan pesan yang spesifik.
Hasil berbeda antar-runtime	Catat versi runtime/paket, random seed, parameter, dan perangkat yang digunakan.
Aturan troubleshooting: Perbaiki satu hal pada satu waktu, simpan pesan kesalahan lengkap, dan catat perubahan versi atau parameter.	

Latihan Modifikasi dan Refleksi

105. Ubah satu parameter pada contoh Fungsi, modul, dan penanganan kesalahan, jalankan ulang dari runtime bersih, dan jelaskan perubahan keluaran.
106. Tambahkan satu validasi baru yang mencegah parameter tidak terdokumentasi dan keluaran tidak tervalidasi.
107. Buat berkas ringkasan TXT atau JSON yang memuat input, status proses, waktu, dan hasil utama.
108. Jelaskan bagian mana yang merupakan fakta keluaran program dan bagian mana yang merupakan interpretasi ilmiah.
109. Susun satu kasus uji yang sengaja salah, lalu pastikan notebook menghasilkan pesan kesalahan yang informatif.

Refleksi: Tuliskan apa yang berubah setelah kode dimodifikasi, bukti apa yang menunjukkan keberhasilan, dan bagian mana yang masih memerlukan verifikasi eksternal.

Rubrik	Deskripsi
Skor 4	Kode berjalan dari runtime bersih, validasi lengkap, interpretasi tepat.
Skor 3	Kode berjalan dan hasil tersimpan, tetapi dokumentasi belum lengkap.
Skor 2	Kode hanya berjalan setelah intervensi atau state tertentu.
Skor 1	Kode belum berjalan dan diagnosis kesalahan belum sistematis.

1.3.3 NumPy, Pandas, dan Matplotlib

NumPy, Pandas, dan Matplotlib merupakan simpul penting dalam alur kerja notebook ilmiah karena menentukan bagaimana prosedur komputasi disiapkan, dijalankan, dan diperiksa. Array numerik, tabel data, statistik sederhana, grafik garis, diagram pencar, dan penyimpanan gambar. Hasil yang dituju berupa hasil yang dapat diperiksa dan disimpan.

Prinsip kerja yang digunakan adalah memisahkan input, proses, validasi, dan ekspor. Pemisahan ini membuat kesalahan lebih mudah dilacak dan mencegah hasil lama dianggap sebagai hasil dari parameter baru. Setiap langkah perlu meninggalkan bukti yang dapat dibaca kembali: nilai parameter, versi pustaka, status proses, nama berkas, serta pesan kesalahan bila terjadi kegagalan.

Tujuan operasional: Pembaca mampu menjelaskan NumPy, Pandas, dan Matplotlib, menjalankan contoh minimal, memeriksa keberhasilan, dan menyimpan bukti hasil.

Komponen	Kriteria
Konsep	Array numerik, tabel data, statistik sederhana, grafik garis, diagram pencar, dan penyimpanan gambar.
Input	Parameter dan data dinyatakan eksplisit.
Proses	Langkah dapat diulang dari runtime bersih.
Output	Nilai, berkas, atau visualisasi yang dapat diperiksa.
Batas	Interpretasi mengikuti metode dan konteks.

Landasan Teori dan Batas Penerapan

Pada bagian NumPy, Pandas, dan Matplotlib, perhatian tidak hanya diarahkan pada keberhasilan menjalankan kode, tetapi juga pada ketepatan asumsi, keterlacakan parameter, dan mutu keluaran. Array numerik, tabel data, statistik sederhana, grafik garis, diagram pencar, dan penyimpanan gambar. Hasil yang dituju berupa hasil yang dapat diperiksa dan disimpan.

Prinsip kerja yang digunakan adalah memisahkan input, proses, validasi, dan ekspor. Pemisahan ini membuat kesalahan lebih mudah dilacak dan mencegah hasil lama dianggap sebagai hasil dari parameter baru. Setiap langkah perlu meninggalkan bukti yang dapat dibaca kembali: nilai parameter, versi pustaka, status proses, nama berkas, serta pesan kesalahan bila terjadi kegagalan.

Risiko utama pada topik ini adalah parameter tidak terdokumentasi dan keluaran tidak tervalidasi. Karena itu, notebook harus memuat pemeriksaan eksplisit dan bukan hanya mengandalkan keluaran yang tampak masuk akal. Pembaca dianjurkan menjalankan notebook dari keadaan runtime bersih, membandingkan hasil dengan batas kewajaran, dan menyimpan artefak penting di lokasi persisten.

Batas penerapan: Contoh pada buku berfungsi sebagai pola kerja. Ukuran molekul, biaya komputasi, versi pustaka, dan perangkat keras dapat mengubah waktu serta strategi yang diperlukan.

Prosedur Bertahap

Pada bagian NumPy, Pandas, dan Matplotlib, perhatian tidak hanya diarahkan pada keberhasilan menjalankan kode, tetapi juga pada ketepatan asumsi, keterlacakan parameter, dan mutu keluaran. Array numerik, tabel data, statistik sederhana, grafik garis, diagram pencar, dan penyimpanan gambar. Hasil yang dituju berupa hasil yang dapat diperiksa dan disimpan.

110. Tuliskan tujuan operasional NumPy, Pandas, dan Matplotlib dan tentukan keluaran yang harus tersedia pada akhir sel.
111. Tetapkan parameter pada satu sel konfigurasi agar perubahan dapat ditelusuri dan tidak tersebar di banyak tempat.
112. Siapkan prosedur komputasi dengan input yang eksplisit; hindari nilai tersembunyi yang hanya tersimpan dalam state kernel.
113. Jalankan pemeriksaan prasyarat, termasuk keberadaan pustaka, bentuk data, tipe nilai, serta konsistensi satuan.
114. Eksekusi proses utama, tangkap status konvergensi atau keberhasilan, dan hentikan alur apabila syarat minimum tidak terpenuhi.
115. Simpan hasil yang dapat diperiksa dan disimpan dengan nama berkas informatif, kemudian tulis interpretasi yang membedakan fakta komputasi dan kesimpulan ilmiah.

Titik kontrol: Jangan melanjutkan ke interpretasi apabila validasi input atau status proses belum memenuhi kriteria.

Kode Dasar Siap Colab

Pembahasan NumPy, Pandas, dan Matplotlib ditempatkan dalam konteks praktik Google Colab agar pembaca dapat menghubungkan konsep, prosedur, kode, dan interpretasi secara utuh. Array numerik, tabel data, statistik sederhana, grafik garis, diagram pencar, dan penyimpanan gambar. Hasil yang dituju berupa hasil yang dapat diperiksa dan disimpan.

Sel kode contoh

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

x = np.linspace(0.4, 2.0, 17)
y = (x - 0.74)**2 - 1.1
df = pd.DataFrame({"jarak_A": x, "energi_rel": y})
print(df.head())

plt.plot(x, y, marker="o")
plt.xlabel("Jarak H-H (Å)")
plt.ylabel("Energi relatif")
plt.savefig("kurva_energi.png", dpi=200, bbox_inches="tight")
```

Cara menjalankan: Tempatkan kode dalam satu atau beberapa sel sesuai komentar. Untuk baris %pip atau perintah shell, gunakan runtime Google Colab.

Penjelasan Baris Kode Penting

Potongan kode	Fungsi
import numpy as np	Mendefinisikan input atau parameter yang akan digunakan pada langkah berikutnya.
import pandas as pd	Mencetak status agar pengguna tidak hanya bergantung pada keluaran visual.
import matplotlib.pyplot as plt	Menjalankan operasi utama dan menyimpan hasil pada variabel yang jelas.
x = np.linspace(0.4, 2.0, 17)	Menambahkan pemeriksaan untuk mencegah hasil tidak valid diteruskan.
y = (x - 0.74)**2 - 1.1	Mencetak status agar pengguna tidak hanya bergantung pada keluaran visual.
df = pd.DataFrame({"jarak_A": x, "energi_rel": y})	Mendefinisikan input atau parameter yang akan digunakan pada langkah berikutnya.
print(df.head())	Mencetak status agar pengguna tidak hanya bergantung pada keluaran visual.

```
plt.plot(x, y, marker="o")
```

Mencetak status agar pengguna tidak hanya bergantung pada keluaran visual.

Nama variabel dipilih agar maknanya terbaca. Hindari singkatan yang tidak dijelaskan, nilai konstan tanpa satuan, serta penggunaan variabel dari sel lama. Komentar harus menerangkan alasan atau asumsi, bukan sekadar mengulang sintaks.

Validasi dan Interpretasi Hasil

Prinsip kerja yang digunakan adalah memisahkan input, proses, validasi, dan ekspor. Pemisahan ini membuat kesalahan lebih mudah dilacak dan mencegah hasil lama dianggap sebagai hasil dari parameter baru. Setiap langkah perlu meninggalkan bukti yang dapat dibaca kembali: nilai parameter, versi pustaka, status proses, nama berkas, serta pesan kesalahan bila terjadi kegagalan.

Aspek validasi	Kriteria lulus
Input	Tipe, bentuk, satuan, muatan, spin, dan jalur berkas dinyatakan eksplisit.
Lingkungan	Versi Python dan pustaka tercatat; backend CPU/GPU terdeteksi.
Proses	Status eksekusi atau konvergensi diperiksa sebelum hasil dipakai.
Numerik	Tidak ada NaN/inf; ukuran array dan jumlah atom konsisten.
Keluaran	Hasil yang dapat diperiksa dan disimpan disimpan dan dapat dibuka kembali.
Interpretasi	Klaim tidak melampaui metode, basis set, atau data yang dihitung.
Interpretasi aman: Sebutkan metode, parameter, satuan, status konvergensi, dan keterbatasan saat menuliskan kesimpulan.	

Kesalahan Umum dan Troubleshooting

Risiko utama pada topik ini adalah parameter tidak terdokumentasi dan keluaran tidak tervalidasi. Karena itu, notebook harus memuat pemeriksaan eksplisit dan bukan hanya mengandalkan keluaran yang tampak masuk akal. Pembaca dianjurkan menjalankan

notebook dari keadaan runtime bersih, membandingkan hasil dengan batas kewajaran, dan menyimpan artefak penting di lokasi persisten.

Gejala	Tindakan perbaikan
Sel dijalankan tidak berurutan	Restart session, lalu gunakan Run all dan periksa variabel prasyarat.
Pustaka tidak ditemukan	Jalankan kembali sel %pip, periksa nama paket, lalu restart bila dependensi inti berubah.
Input tampak benar tetapi gagal	Cetak tipe, shape, satuan, muatan, spin, dan beberapa baris pertama data.
Parameter tidak terdokumentasi dan keluaran tidak tervalidasi	Tambahkan validasi sebelum proses utama dan hentikan eksekusi dengan pesan yang spesifik.
Hasil berbeda antar-runtime	Catat versi runtime/paket, random seed, parameter, dan perangkat yang digunakan.
Aturan troubleshooting: Perbaiki satu hal pada satu waktu, simpan pesan kesalahan lengkap, dan catat perubahan versi atau parameter.	

Latihan Modifikasi dan Refleksi

116. Ubah satu parameter pada contoh NumPy, Pandas, dan Matplotlib, jalankan ulang dari runtime bersih, dan jelaskan perubahan keluaran.
117. Tambahkan satu validasi baru yang mencegah parameter tidak terdokumentasi dan keluaran tidak tervalidasi.
118. Buat berkas ringkasan TXT atau JSON yang memuat input, status proses, waktu, dan hasil utama.
119. Jelaskan bagian mana yang merupakan fakta keluaran program dan bagian mana yang merupakan interpretasi ilmiah.
120. Susun satu kasus uji yang sengaja salah, lalu pastikan notebook menghasilkan pesan kesalahan yang informatif.

Refleksi: Tuliskan apa yang berubah setelah kode dimodifikasi, bukti apa yang menunjukkan keberhasilan, dan bagian mana yang masih memerlukan verifikasi eksternal.

Rubrik

Deskripsi

Skor 4	Kode berjalan dari runtime bersih, validasi lengkap, interpretasi tepat.
Skor 3	Kode berjalan dan hasil tersimpan, tetapi dokumentasi belum lengkap.
Skor 2	Kode hanya berjalan setelah intervensi atau state tertentu.
Skor 1	Kode belum berjalan dan diagnosis kesalahan belum sistematis.

Rangkuman Bab

Bab ini membangun fondasi penggunaan Google Colab dan Python. Pembaca belajar membedakan notebook, runtime, sel, output, dan berkas; kemudian berlatih menggunakan tipe data, fungsi, NumPy, Pandas, serta Matplotlib.

- Python sebagai bahasa komputasi ilmiah
- Konsep Jupyter Notebook
- Google Colab sebagai notebook berbasis awan
- Membuat, membuka, mengganti nama, dan menyimpan notebook
- Menjalankan sel dan mengelola runtime
- Mengunggah dan mengunduh berkas
- Variabel, tipe data, operator, dan struktur kontrol
- Fungsi, modul, dan penanganan kesalahan
- NumPy, Pandas, dan Matplotlib

Inti bab: Keberhasilan teknis harus selalu disertai validasi, dokumentasi, dan interpretasi yang proporsional.

Proyek Praktik Terpadu

Proyek terpadu menggabungkan seluruh keterampilan bab dalam satu notebook. Gunakan satu sel konfigurasi, fungsi modular, validasi, logging sederhana, penyimpanan hasil, dan bagian pembahasan.

121. Tentukan pertanyaan atau tujuan komputasi.
122. Siapkan input dan parameter.
123. Jalankan proses utama.
124. Validasi status dan nilai.
125. Buat satu visualisasi atau tabel.
126. Ekspor hasil dan tulis pembahasan.

127. Restart session dan Run all.

Produk akhir: Notebook .ipynb, satu folder hasil, arsip ZIP, dan ringkasan pengujian.

Pertanyaan Analitis

- 128. Mengapa kode yang menghasilkan angka belum tentu menghasilkan bukti ilmiah?
- 129. Bagaimana versi runtime dapat memengaruhi reproduksibilitas?
- 130. Apa perbedaan pemeriksaan sintaks, pemeriksaan numerik, dan validasi kimia?
- 131. Kapan fallback dapat diterima, dan bagaimana harus dilaporkan?
- 132. Apa batas interpretasi dari metode atau contoh pada bab ini?
- 133. Bagaimana merancang notebook agar mudah direview oleh orang lain?

Panduan Jawaban dan Rubrik

Komponen	Bobot dan indikator
Ketepatan konsep	25% - definisi, asumsi, dan satuan benar.
Kualitas kode	25% - modular, terbaca, dan menangani kesalahan.
Validasi	25% - input, proses, dan keluaran diperiksa.
Interpretasi	15% - klaim sesuai bukti dan keterbatasan.
Reproduksibilitas	10% - dapat dijalankan dari runtime bersih.

Panduan jawaban menilai alur penalaran, bukan hanya keluaran akhir. Perbedaan nilai kecil dapat diterima bila versi atau perangkat berbeda dan semua parameter dilaporkan.

Daftar Periksa Penguasaan Bab

Pernyataan	Cek
Saya dapat menjelaskan konsep utama	☐
Saya dapat menjalankan kode dari runtime bersih	☐
Saya memeriksa status dan nilai numerik	☐

Saya menyimpan hasil di lokasi persisten	☐
Saya dapat menjelaskan kesalahan umum	☐
Saya menuliskan batas interpretasi	☐
Saya mencatat versi dan waktu pengujian	☐

Tindak lanjut: Ulangi halaman yang belum dikuasai, lalu kerjakan proyek bab tanpa melihat kode contoh secara langsung.

2

Menyusun Notebook Ilmiah yang Reproduksi?ktif

BAB 2 - MENYUSUN NOTEBOOK ILMIAH YANG REPRODUKTIF

Bab ini menempatkan reproduksibilitas sebagai prinsip utama. Struktur notebook, instalasi paket, pemeriksaan lingkungan, pencatatan versi, penyimpanan persisten, dan arsip hasil dibahas sebagai satu alur.

Arah Bab: Setiap konsep diikuti prosedur, kode, penjelasan, validasi, troubleshooting, latihan, dan refleksi. Keluaran numerik tidak boleh dipisahkan dari konteks metode dan batas penerapannya.

Komponen	Kriteria
Orientasi	Memahami konsep dan istilah.
Praktik	Menjalankan kode secara bertahap.
Validasi	Memastikan input, proses, dan keluaran sah.
Interpretasi	Menjelaskan hasil tanpa klaim berlebihan.

Capaian Pembelajaran dan Prasyarat

- Menjelaskan konsep utama bab dengan istilah yang tepat.
- Menjalankan notebook dari runtime bersih dan mendokumentasikan versi.
- Memodifikasi parameter tanpa merusak alur validasi.
- Menyimpan hasil dalam format yang dapat dibuka kembali.
- Membedakan keluaran program, interpretasi, dan keterbatasan.

Prasyarat: Kemampuan menggunakan browser, akun Google, operasi berkas dasar, dan kemauan membaca pesan kesalahan secara teliti.

Komponen	Kriteria
Pengetahuan	Konsep sebelumnya dan satuan dasar.
Keterampilan	Menyalin kode, menjalankan sel, dan membaca output.
Sikap ilmiah	Jujur terhadap status pengujian dan ketidakpastian.
Perangkat	Browser modern dan koneksi internet stabil.

Peta Konsep dan Alur Kerja

Subbab	Alur konsep
2.1 Struktur Notebook yang Sistematis	Judul, identitas, tujuan, dan teori singkat → Konfigurasi parameter dan fungsi modular → Urutan eksekusi dan kebersihan notebook
2.2 Instalasi Pustaka dan Pemeriksaan Lingkungan	Instalasi dengan %pip → Pemeriksaan Python, sistem, RAM, dan versi pustaka → Deteksi CPU, GPU, CUDA, dan kompatibilitas
2.3 Input, Output, dan Pengelolaan Berkas	Membaca dan menulis TXT, CSV, JSON, dan XYZ → Google Drive dan penyimpanan hasil → Membuat laporan dan arsip ZIP

Peta konsep dibaca dari kiri ke kanan dan dari input menuju keluaran. Hubungan antarkomponen tidak selalu linear: validasi dapat mengembalikan pembaca ke tahap input, sedangkan troubleshooting dapat memerlukan instalasi ulang atau penggantian metode.

Prinsip alur: Notebook yang baik memperlihatkan jalur keputusan: apa yang diperiksa, kapan proses dihentikan, kapan fallback digunakan, dan berkas apa yang disimpan.

Istilah Kunci dan Konvensi

Istilah	Konvensi penggunaan
notebook	Istilah digunakan secara konsisten dan dijelaskan saat pertama muncul.
runtime	Nilai atau statusnya dicatat dalam notebook dan laporan.
kernel	Maknanya dibedakan dari istilah sehari-hari yang mirip.
sel Markdown	Maknanya dibedakan dari istilah sehari-hari yang mirip.
sel kode	Istilah digunakan secara konsisten dan dijelaskan saat pertama muncul.
parameter	Nilai atau statusnya dicatat dalam notebook dan laporan.
validasi	Nilai atau statusnya dicatat dalam notebook dan laporan.
konvergensi	Nilai atau statusnya dicatat dalam notebook dan laporan.
Konvensi satuan: Panjang molekul dinyatakan dalam Angstrom, energi elektronik dalam Hartree, energi orbital dapat dikonversi ke eV, dan perubahan energi dapat dilaporkan dalam kJ mol ⁻¹ .	

Asesmen Diagnostik Awal

134. Apa perbedaan notebook, runtime, dan berkas .ipynb?
135. Mengapa menjalankan sel tidak berurutan dapat menghasilkan keluaran yang menyesatkan?
136. Kapan sebuah nilai numerik boleh ditafsirkan sebagai hasil yang sah?
137. Informasi lingkungan apa yang perlu dicatat agar pekerjaan dapat diulang?
138. Bagaimana membedakan kegagalan instalasi, kegagalan input, dan kegagalan konvergensi?
139. Berkas apa yang harus disimpan sebelum runtime Colab terputus?

Cara menggunakan asesmen: Jawab sebelum memulai bab, tandai tingkat keyakinan, lalu ulangi setelah menyelesaikan proyek bab. Perubahan alasan lebih penting daripada perubahan skor saja.

2.1 Struktur Notebook yang Sistematis

Struktur Notebook yang Sistematis mengorganisasi tiga komponen yang saling bergantung. Pembahasan bergerak dari konsep, menuju prosedur, kemudian menghasilkan artefak yang dapat divalidasi dan disimpan.

Sub-subbab	Fokus
2.1.1 Judul, identitas, tujuan, dan teori singkat	Penyusunan bagian awal notebook dan penulisan dokumentasi menggunakan Markdown.
2.1.2 Konfigurasi parameter dan fungsi modular	Memisahkan parameter, fungsi, perhitungan utama, analisis hasil, dan ekspor.
2.1.3 Urutan eksekusi dan kebersihan notebook	Menghindari variabel tersembunyi, sel tidak berurutan, output lama, dan konflik state kernel.
Produk belajar: Satu notebook terstruktur, satu berkas keluaran, satu catatan validasi, dan satu interpretasi singkat yang menyatakan batas hasil.	

Rencana Praktik dan Produk Belajar

140. Pelajari Judul, identitas, tujuan, dan teori singkat dan jalankan kode dasar yang menyertainya.
141. Pelajari Konfigurasi parameter dan fungsi modular dan jalankan kode dasar yang menyertainya.
142. Pelajari Urutan eksekusi dan kebersihan notebook dan jalankan kode dasar yang menyertainya.

Komponen	Kriteria
Waktu	60-120 menit, bergantung pada komputasi.
Input	Notebook baru atau salinan notebook bab.

Produk	Kode, output, berkas hasil, dan refleksi.
Uji	Restart session lalu Run all.
Dokumentasi	Versi, parameter, status, dan waktu eksekusi.
Kriteria selesai: Semua sel dapat dijalankan berurutan, pesan kesalahan ditangani, dan hasil utama tersimpan di lokasi persisten.	

2.1.1 Judul, identitas, tujuan, dan teori singkat

Pada bagian Judul, identitas, tujuan, dan teori singkat, perhatian tidak hanya diarahkan pada keberhasilan menjalankan kode, tetapi juga pada ketepatan asumsi, keterlacakan parameter, dan mutu keluaran. Penyusunan bagian awal notebook dan penulisan dokumentasi menggunakan Markdown. Hasil yang dituju berupa sel Markdown, sel kode, output, dan metadata.

Prinsip kerja yang digunakan adalah memisahkan input, proses, validasi, dan ekspor. Pemisahan ini membuat kesalahan lebih mudah dilacak dan mencegah hasil lama dianggap sebagai hasil dari parameter baru. Setiap langkah perlu meninggalkan bukti yang dapat dibaca kembali: nilai parameter, versi pustaka, status proses, nama berkas, serta pesan kesalahan bila terjadi kegagalan.

Tujuan operasional: Pembaca mampu menjelaskan Judul, identitas, tujuan, dan teori singkat, menjalankan contoh minimal, memeriksa keberhasilan, dan menyimpan bukti hasil.

Komponen	Kriteria
Konsep	Penyusunan bagian awal notebook dan penulisan dokumentasi menggunakan Markdown.
Input	Parameter dan data dinyatakan eksplisit.
Proses	Langkah dapat diulang dari runtime bersih.
Output	Nilai, berkas, atau visualisasi yang dapat diperiksa.
Batas	Interpretasi mengikuti metode dan konteks.

Landasan Teori dan Batas Penerapan

Pembahasan Judul, identitas, tujuan, dan teori singkat ditempatkan dalam konteks praktik Google Colab agar pembaca dapat menghubungkan konsep, prosedur, kode, dan interpretasi secara utuh. Penyusunan bagian awal notebook dan penulisan dokumentasi menggunakan Markdown. Hasil yang dituju berupa sel Markdown, sel kode, output, dan metadata.

Prinsip kerja yang digunakan adalah memisahkan input, proses, validasi, dan ekspor. Pemisahan ini membuat kesalahan lebih mudah dilacak dan mencegah hasil lama dianggap sebagai hasil dari parameter baru. Setiap langkah perlu meninggalkan bukti yang dapat dibaca kembali: nilai parameter, versi pustaka, status proses, nama berkas, serta pesan kesalahan bila terjadi kegagalan.

Risiko utama pada topik ini adalah state kernel tersembunyi dan urutan eksekusi. Karena itu, notebook harus memuat pemeriksaan eksplisit dan bukan hanya mengandalkan keluaran yang tampak masuk akal. Pembaca dianjurkan menjalankan notebook dari keadaan runtime bersih, membandingkan hasil dengan batas kewajaran, dan menyimpan artefak penting di lokasi persisten.

Batas penerapan: Contoh pada buku berfungsi sebagai pola kerja. Ukuran molekul, biaya komputasi, versi pustaka, dan perangkat keras dapat mengubah waktu serta strategi yang diperlukan.

Prosedur Bertahap

Pembahasan Judul, identitas, tujuan, dan teori singkat ditempatkan dalam konteks praktik Google Colab agar pembaca dapat menghubungkan konsep, prosedur, kode, dan interpretasi secara utuh. Penyusunan bagian awal notebook dan penulisan dokumentasi menggunakan Markdown. Hasil yang dituju berupa sel Markdown, sel kode, output, dan metadata.

143. Tuliskan tujuan operasional Judul, identitas, tujuan, dan teori singkat dan tentukan keluaran yang harus tersedia pada akhir sel.
144. Tetapkan parameter pada satu sel konfigurasi agar perubahan dapat ditelusuri dan tidak tersebar di banyak tempat.
145. Siapkan notebook dengan input yang eksplisit; hindari nilai tersembunyi yang hanya tersimpan dalam state kernel.
146. Jalankan pemeriksaan prasyarat, termasuk keberadaan pustaka, bentuk data, tipe nilai, serta konsistensi satuan.
147. Eksekusi proses utama, tangkap status konvergensi atau keberhasilan, dan hentikan alur apabila syarat minimum tidak terpenuhi.
148. Simpan sel Markdown, sel kode, output, dan metadata dengan nama berkas informatif, kemudian tulis interpretasi yang membedakan fakta komputasi dan kesimpulan ilmiah.

Titik kontrol: Jangan melanjutkan ke interpretasi apabila validasi input atau status proses belum memenuhi kriteria.

Kode Dasar Siap Colab

Pada bagian Judul, identitas, tujuan, dan teori singkat, perhatian tidak hanya diarahkan pada keberhasilan menjalankan kode, tetapi juga pada ketepatan asumsi, keterlacakan parameter, dan mutu keluaran. Penyusunan bagian awal notebook dan penulisan dokumentasi menggunakan Markdown. Hasil yang dituju berupa sel Markdown, sel kode, output, dan metadata.

Sel kode contoh

```
# Judul Notebook

**Penulis:** KASMUI
**Tujuan:** Menjalankan perhitungan yang dapat direproduksi.
**Metode:** B3LYP/def2-TZVP
**Input utama:** muatan, spin, dan koordinat molekul.

> Jalankan notebook dari atas ke bawah setelah Restart session.
```

Cara menjalankan: Tempatkan kode dalam satu atau beberapa sel sesuai komentar. Untuk baris %pip atau perintah shell, gunakan runtime Google Colab.

Penjelasan Baris Kode Penting

Potongan kode	Fungsi
# Judul Notebook	Mendefinisikan input atau parameter yang akan digunakan pada langkah berikutnya.
Penulis: KASMUI	Mendefinisikan input atau parameter yang akan digunakan pada langkah berikutnya.
Tujuan: Menjalankan perhitungan yang dapat d	Menjalankan operasi utama dan menyimpan hasil pada variabel yang jelas.
Metode: B3LYP/def2-TZVP	Mencetak status agar pengguna tidak hanya bergantung pada keluaran visual.
Input utama: muatan, spin, dan koordinat mol	Mendefinisikan input atau parameter yang akan digunakan pada langkah berikutnya.

> Jalankan notebook dari atas ke bawah setelah R	Mencetak status agar pengguna tidak hanya bergantung pada keluaran visual.

Nama variabel dipilih agar maknanya terbaca. Hindari singkatan yang tidak dijelaskan, nilai konstan tanpa satuan, serta penggunaan variabel dari sel lama. Komentar harus menerangkan alasan atau asumsi, bukan sekadar mengulang sintaks.

Validasi dan Interpretasi Hasil

Prinsip kerja yang digunakan adalah memisahkan input, proses, validasi, dan ekspor. Pemisahan ini membuat kesalahan lebih mudah dilacak dan mencegah hasil lama dianggap sebagai hasil dari parameter baru. Setiap langkah perlu meninggalkan bukti yang dapat dibaca kembali: nilai parameter, versi pustaka, status proses, nama berkas, serta pesan kesalahan bila terjadi kegagalan.

Aspek validasi	Kriteria lulus
Input	Tipe, bentuk, satuan, muatan, spin, dan jalur berkas dinyatakan eksplisit.
Lingkungan	Versi Python dan pustaka tercatat; backend CPU/GPU terdeteksi.
Proses	Status eksekusi atau konvergensi diperiksa sebelum hasil dipakai.
Numerik	Tidak ada NaN/inf; ukuran array dan jumlah atom konsisten.
Keluaran	Sel markdown, sel kode, output, dan metadata disimpan dan dapat dibuka kembali.
Interpretasi	Klaim tidak melampaui metode, basis set, atau data yang dihitung.
Interpretasi aman: Sebutkan metode, parameter, satuan, status konvergensi, dan keterbatasan saat menuliskan kesimpulan.	

Kesalahan Umum dan Troubleshooting

Risiko utama pada topik ini adalah state kernel tersembunyi dan urutan eksekusi. Karena itu, notebook harus memuat pemeriksaan eksplisit dan bukan hanya mengandalkan

keluaran yang tampak masuk akal. Pembaca dianjurkan menjalankan notebook dari keadaan runtime bersih, membandingkan hasil dengan batas kewajaran, dan menyimpan artefak penting di lokasi persisten.

Gejala	Tindakan perbaikan
Sel dijalankan tidak berurutan	Restart session, lalu gunakan Run all dan periksa variabel prasyarat.
Pustaka tidak ditemukan	Jalankan kembali sel %pip, periksa nama paket, lalu restart bila dependensi inti berubah.
Input tampak benar tetapi gagal	Cetak tipe, shape, satuan, muatan, spin, dan beberapa baris pertama data.
State kernel tersembunyi dan urutan eksekusi	Tambahkan validasi sebelum proses utama dan hentikan eksekusi dengan pesan yang spesifik.
Hasil berbeda antar-runtime	Catat versi runtime/paket, random seed, parameter, dan perangkat yang digunakan.
Aturan troubleshooting: Perbaiki satu hal pada satu waktu, simpan pesan kesalahan lengkap, dan catat perubahan versi atau parameter.	

Latihan Modifikasi dan Refleksi

149. Ubah satu parameter pada contoh Judul, identitas, tujuan, dan teori singkat, jalankan ulang dari runtime bersih, dan jelaskan perubahan keluaran.
150. Tambahkan satu validasi baru yang mencegah state kernel tersembunyi dan urutan eksekusi.
151. Buat berkas ringkasan TXT atau JSON yang memuat input, status proses, waktu, dan hasil utama.
152. Jelaskan bagian mana yang merupakan fakta keluaran program dan bagian mana yang merupakan interpretasi ilmiah.
153. Susun satu kasus uji yang sengaja salah, lalu pastikan notebook menghasilkan pesan kesalahan yang informatif.

Refleksi: Tuliskan apa yang berubah setelah kode dimodifikasi, bukti apa yang menunjukkan keberhasilan, dan bagian mana yang masih memerlukan verifikasi eksternal.

Rubrik	Deskripsi
Skor 4	Kode berjalan dari runtime bersih, validasi lengkap, interpretasi tepat.
Skor 3	Kode berjalan dan hasil tersimpan, tetapi dokumentasi belum lengkap.
Skor 2	Kode hanya berjalan setelah intervensi atau state tertentu.
Skor 1	Kode belum berjalan dan diagnosis kesalahan belum sistematis.

2.1.2 Konfigurasi parameter dan fungsi modular

Pembahasan Konfigurasi parameter dan fungsi modular ditempatkan dalam konteks praktik Google Colab agar pembaca dapat menghubungkan konsep, prosedur, kode, dan interpretasi secara utuh. Memisahkan parameter, fungsi, perhitungan utama, analisis hasil, dan ekspor. Hasil yang dituju berupa hasil yang dapat diperiksa dan disimpan.

Prinsip kerja yang digunakan adalah memisahkan input, proses, validasi, dan ekspor. Pemisahan ini membuat kesalahan lebih mudah dilacak dan mencegah hasil lama dianggap sebagai hasil dari parameter baru. Setiap langkah perlu meninggalkan bukti yang dapat dibaca kembali: nilai parameter, versi pustaka, status proses, nama berkas, serta pesan kesalahan bila terjadi kegagalan.

Tujuan operasional: Pembaca mampu menjelaskan Konfigurasi parameter dan fungsi modular, menjalankan contoh minimal, memeriksa keberhasilan, dan menyimpan bukti hasil.

Komponen	Kriteria
Konsep	Memisahkan parameter, fungsi, perhitungan utama, analisis hasil, dan ekspor.
Input	Parameter dan data dinyatakan eksplisit.
Proses	Langkah dapat diulang dari runtime bersih.

Output	Nilai, berkas, atau visualisasi yang dapat diperiksa.
Batas	Interpretasi mengikuti metode dan konteks.

Landasan Teori dan Batas Penerapan

Pembahasan Konfigurasi parameter dan fungsi modular ditempatkan dalam konteks praktik Google Colab agar pembaca dapat menghubungkan konsep, prosedur, kode, dan interpretasi secara utuh. Memisahkan parameter, fungsi, perhitungan utama, analisis hasil, dan ekspor. Hasil yang dituju berupa hasil yang dapat diperiksa dan disimpan.

Prinsip kerja yang digunakan adalah memisahkan input, proses, validasi, dan ekspor. Pemisahan ini membuat kesalahan lebih mudah dilacak dan mencegah hasil lama dianggap sebagai hasil dari parameter baru. Setiap langkah perlu meninggalkan bukti yang dapat dibaca kembali: nilai parameter, versi pustaka, status proses, nama berkas, serta pesan kesalahan bila terjadi kegagalan.

Risiko utama pada topik ini adalah parameter tidak terdokumentasi dan keluaran tidak tervalidasi. Karena itu, notebook harus memuat pemeriksaan eksplisit dan bukan hanya mengandalkan keluaran yang tampak masuk akal. Pembaca dianjurkan menjalankan notebook dari keadaan runtime bersih, membandingkan hasil dengan batas kewajaran, dan menyimpan artefak penting di lokasi persisten.

Batas penerapan: Contoh pada buku berfungsi sebagai pola kerja. Ukuran molekul, biaya komputasi, versi pustaka, dan perangkat keras dapat mengubah waktu serta strategi yang diperlukan.

Prosedur Bertahap

Pembahasan Konfigurasi parameter dan fungsi modular ditempatkan dalam konteks praktik Google Colab agar pembaca dapat menghubungkan konsep, prosedur, kode, dan interpretasi secara utuh. Memisahkan parameter, fungsi, perhitungan utama, analisis hasil, dan ekspor. Hasil yang dituju berupa hasil yang dapat diperiksa dan disimpan.

154. Tuliskan tujuan operasional Konfigurasi parameter dan fungsi modular dan tentukan keluaran yang harus tersedia pada akhir sel.
155. Tetapkan parameter pada satu sel konfigurasi agar perubahan dapat ditelusuri dan tidak tersebar di banyak tempat.
156. Siapkan prosedur komputasi dengan input yang eksplisit; hindari nilai tersembunyi yang hanya tersimpan dalam state kernel.
157. Jalankan pemeriksaan prasyarat, termasuk keberadaan pustaka, bentuk data, tipe nilai, serta konsistensi satuan.

158. Eksekusi proses utama, tangkap status konvergensi atau keberhasilan, dan hentikan alur apabila syarat minimum tidak terpenuhi.
159. Simpan hasil yang dapat diperiksa dan disimpan dengan nama berkas informatif, kemudian tulis interpretasi yang membedakan fakta komputasi dan kesimpulan ilmiah.

Titik kontrol: Jangan melanjutkan ke interpretasi apabila validasi input atau status proses belum memenuhi kriteria.

Kode Dasar Siap Colab

Pada bagian Konfigurasi parameter dan fungsi modular, perhatian tidak hanya diarahkan pada keberhasilan menjalankan kode, tetapi juga pada ketepatan asumsi, keterlacakan parameter, dan mutu keluaran. Memisahkan parameter, fungsi, perhitungan utama, analisis hasil, dan ekspor. Hasil yang dituju berupa hasil yang dapat diperiksa dan disimpan.

Sel kode contoh

```
from dataclasses import dataclass

@dataclass(frozen=True)
class Config:
    method: str = "b3lyp"
    basis: str = "def2-tzvp"
    charge: int = 0
    spin: int = 0
    max_cycle: int = 200

CFG = Config()
print(CFG)
```

Cara menjalankan: Tempatkan kode dalam satu atau beberapa sel sesuai komentar. Untuk baris %pip atau perintah shell, gunakan runtime Google Colab.

Penjelasan Baris Kode Penting

Potongan kode	Fungsi
<code>from dataclasses import dataclass</code>	Menambahkan pemeriksaan untuk mencegah hasil tidak valid diteruskan.
<code>@dataclass(frozen=True)</code>	Mencetak status agar pengguna tidak hanya bergantung pada keluaran visual.

<code>class Config:</code>	Menjalankan operasi utama dan menyimpan hasil pada variabel yang jelas.
<code>method: str = "b3lyp"</code>	Menjalankan operasi utama dan menyimpan hasil pada variabel yang jelas.
<code>basis: str = "def2-tzvp"</code>	Menjalankan operasi utama dan menyimpan hasil pada variabel yang jelas.
<code>charge: int = 0</code>	Menambahkan pemeriksaan untuk mencegah hasil tidak valid diteruskan.
<code>spin: int = 0</code>	Menjalankan operasi utama dan menyimpan hasil pada variabel yang jelas.
<code>max_cycle: int = 200</code>	Menambahkan pemeriksaan untuk mencegah hasil tidak valid diteruskan.

Nama variabel dipilih agar maknanya terbaca. Hindari singkatan yang tidak dijelaskan, nilai konstan tanpa satuan, serta penggunaan variabel dari sel lama. Komentar harus menerangkan alasan atau asumsi, bukan sekadar mengulang sintaks.

Validasi dan Interpretasi Hasil

Prinsip kerja yang digunakan adalah memisahkan input, proses, validasi, dan ekspor. Pemisahan ini membuat kesalahan lebih mudah dilacak dan mencegah hasil lama dianggap sebagai hasil dari parameter baru. Setiap langkah perlu meninggalkan bukti yang dapat dibaca kembali: nilai parameter, versi pustaka, status proses, nama berkas, serta pesan kesalahan bila terjadi kegagalan.

Aspek validasi	Kriteria lulus
Input	Tipe, bentuk, satuan, muatan, spin, dan jalur berkas dinyatakan eksplisit.
Lingkungan	Versi Python dan pustaka tercatat; backend CPU/GPU terdeteksi.
Proses	Status eksekusi atau konvergensi diperiksa sebelum hasil dipakai.

Numerik	Tidak ada NaN/inf; ukuran array dan jumlah atom konsisten.
Keluaran	Hasil yang dapat diperiksa dan disimpan disimpan dan dapat dibuka kembali.
Interpretasi	Klaim tidak melampaui metode, basis set, atau data yang dihitung.

Interpretasi aman: Sebutkan metode, parameter, satuan, status konvergensi, dan keterbatasan saat menuliskan kesimpulan.

Kesalahan Umum dan Troubleshooting

Risiko utama pada topik ini adalah parameter tidak terdokumentasi dan keluaran tidak tervalidasi. Karena itu, notebook harus memuat pemeriksaan eksplisit dan bukan hanya mengandalkan keluaran yang tampak masuk akal. Pembaca dianjurkan menjalankan notebook dari keadaan runtime bersih, membandingkan hasil dengan batas kewajaran, dan menyimpan artefak penting di lokasi persisten.

Gejala	Tindakan perbaikan
Sel dijalankan tidak berurutan	Restart session, lalu gunakan Run all dan periksa variabel prasyarat.
Pustaka tidak ditemukan	Jalankan kembali sel %pip, periksa nama paket, lalu restart bila dependensi inti berubah.
Input tampak benar tetapi gagal	Cetak tipe, shape, satuan, muatan, spin, dan beberapa baris pertama data.
Parameter tidak terdokumentasi dan keluaran tidak tervalidasi	Tambahkan validasi sebelum proses utama dan hentikan eksekusi dengan pesan yang spesifik.
Hasil berbeda antar-runtime	Catat versi runtime/paket, random seed, parameter, dan perangkat yang digunakan.

Aturan troubleshooting: Perbaiki satu hal pada satu waktu, simpan pesan kesalahan lengkap, dan catat perubahan versi atau parameter.

Latihan Modifikasi dan Refleksi

160. Ubah satu parameter pada contoh Konfigurasi parameter dan fungsi modular, jalankan ulang dari runtime bersih, dan jelaskan perubahan keluaran.
161. Tambahkan satu validasi baru yang mencegah parameter tidak terdokumentasi dan keluaran tidak tervalidasi.
162. Buat berkas ringkasan TXT atau JSON yang memuat input, status proses, waktu, dan hasil utama.
163. Jelaskan bagian mana yang merupakan fakta keluaran program dan bagian mana yang merupakan interpretasi ilmiah.
164. Susun satu kasus uji yang sengaja salah, lalu pastikan notebook menghasilkan pesan kesalahan yang informatif.

Refleksi: Tuliskan apa yang berubah setelah kode dimodifikasi, bukti apa yang menunjukkan keberhasilan, dan bagian mana yang masih memerlukan verifikasi eksternal.

Rubrik	Deskripsi
Skor 4	Kode berjalan dari runtime bersih, validasi lengkap, interpretasi tepat.
Skor 3	Kode berjalan dan hasil tersimpan, tetapi dokumentasi belum lengkap.
Skor 2	Kode hanya berjalan setelah intervensi atau state tertentu.
Skor 1	Kode belum berjalan dan diagnosis kesalahan belum sistematis.

2.1.3 Urutan eksekusi dan kebersihan notebook

Urutan eksekusi dan kebersihan notebook merupakan simpul penting dalam alur kerja notebook ilmiah karena menentukan bagaimana notebook disiapkan, dijalankan, dan diperiksa. Menghindari variabel tersembunyi, sel tidak berurutan, output lama, dan konflik state kernel. Hasil yang dituju berupa sel Markdown, sel kode, output, dan metadata.

Prinsip kerja yang digunakan adalah memisahkan input, proses, validasi, dan ekspor. Pemisahan ini membuat kesalahan lebih mudah dilacak dan mencegah hasil lama dianggap sebagai hasil dari parameter baru. Setiap langkah perlu meninggalkan bukti yang dapat dibaca kembali: nilai parameter, versi pustaka, status proses, nama berkas, serta pesan kesalahan bila terjadi kegagalan.

Tujuan operasional: Pembaca mampu menjelaskan Urutan eksekusi dan kebersihan notebook, menjalankan contoh minimal, memeriksa keberhasilan, dan menyimpan bukti hasil.

Komponen	Kriteria
Konsep	Menghindari variabel tersembunyi, sel tidak berurutan, output lama, dan konflik state kernel.
Input	Parameter dan data dinyatakan eksplisit.
Proses	Langkah dapat diulang dari runtime bersih.
Output	Nilai, berkas, atau visualisasi yang dapat diperiksa.
Batas	Interpretasi mengikuti metode dan konteks.

Landasan Teori dan Batas Penerapan

Urutan eksekusi dan kebersihan notebook merupakan simpul penting dalam alur kerja notebook ilmiah karena menentukan bagaimana notebook disiapkan, dijalankan, dan diperiksa. Menghindari variabel tersembunyi, sel tidak berurutan, output lama, dan konflik state kernel. Hasil yang dituju berupa sel Markdown, sel kode, output, dan metadata.

Prinsip kerja yang digunakan adalah memisahkan input, proses, validasi, dan ekspor. Pemisahan ini membuat kesalahan lebih mudah dilacak dan mencegah hasil lama dianggap sebagai hasil dari parameter baru. Setiap langkah perlu meninggalkan bukti yang dapat dibaca kembali: nilai parameter, versi pustaka, status proses, nama berkas, serta pesan kesalahan bila terjadi kegagalan.

Risiko utama pada topik ini adalah state kernel tersembunyi dan urutan eksekusi. Karena itu, notebook harus memuat pemeriksaan eksplisit dan bukan hanya mengandalkan keluaran yang tampak masuk akal. Pembaca dianjurkan menjalankan notebook dari keadaan runtime bersih, membandingkan hasil dengan batas kewajaran, dan menyimpan artefak penting di lokasi persisten.

Batas penerapan: Contoh pada buku berfungsi sebagai pola kerja. Ukuran molekul, biaya komputasi, versi pustaka, dan perangkat keras dapat mengubah waktu serta strategi yang diperlukan.

Prosedur Bertahap

Urutan eksekusi dan kebersihan notebook merupakan simpul penting dalam alur kerja notebook ilmiah karena menentukan bagaimana notebook disiapkan, dijalankan, dan diperiksa. Menghindari variabel tersembunyi, sel tidak berurutan, output lama, dan konflik state kernel. Hasil yang dituju berupa sel Markdown, sel kode, output, dan metadata.

165. Tuliskan tujuan operasional Urutan eksekusi dan kebersihan notebook dan tentukan keluaran yang harus tersedia pada akhir sel.
166. Tetapkan parameter pada satu sel konfigurasi agar perubahan dapat ditelusuri dan tidak tersebar di banyak tempat.
167. Siapkan notebook dengan input yang eksplisit; hindari nilai tersembunyi yang hanya tersimpan dalam state kernel.
168. Jalankan pemeriksaan prasyarat, termasuk keberadaan pustaka, bentuk data, tipe nilai, serta konsistensi satuan.
169. Eksekusi proses utama, tangkap status konvergensi atau keberhasilan, dan hentikan alur apabila syarat minimum tidak terpenuhi.
170. Simpan sel Markdown, sel kode, output, dan metadata dengan nama berkas informatif, kemudian tulis interpretasi yang membedakan fakta komputasi dan kesimpulan ilmiah.

Titik kontrol: Jangan melanjutkan ke interpretasi apabila validasi input atau status proses belum memenuhi kriteria.

Kode Dasar Siap Colab

Pada bagian Urutan eksekusi dan kebersihan notebook, perhatian tidak hanya diarahkan pada keberhasilan menjalankan kode, tetapi juga pada ketepatan asumsi, keterlacakan parameter, dan mutu keluaran. Menghindari variabel tersembunyi, sel tidak berurutan, output lama, dan konflik state kernel. Hasil yang dituju berupa sel Markdown, sel kode, output, dan metadata.

Sel kode contoh

```
# Uji notebook bersih
REQUIRED = ["CFG", "run_calculation", "export_results"]
missing = [name for name in REQUIRED if name not in globals()]
if missing:
    raise RuntimeError(f"Sel prasyarat belum dijalankan: {missing}")
print("Urutan eksekusi konsisten.")
```

Cara menjalankan: Tempatkan kode dalam satu atau beberapa sel sesuai komentar. Untuk baris %pip atau perintah shell, gunakan runtime Google Colab.

Penjelasan Baris Kode Penting

Potongan kode	Fungsi
# Uji notebook bersih	Menjalankan operasi utama dan menyimpan hasil pada variabel yang jelas.
REQUIRED = ["CFG", "run_calculation", "export_re	Menambahkan pemeriksaan untuk mencegah hasil tidak valid diteruskan.
missing = [name for name in REQUIRED if name not	Mencetak status agar pengguna tidak hanya bergantung pada keluaran visual.
if missing:	Mencetak status agar pengguna tidak hanya bergantung pada keluaran visual.
raise RuntimeError(f'Sel prasyarat belum dij	Mendefinisikan input atau parameter yang akan digunakan pada langkah berikutnya.
print("Urutan eksekusi konsisten.")	Mendefinisikan input atau parameter yang akan digunakan pada langkah berikutnya.

Nama variabel dipilih agar maknanya terbaca. Hindari singkatan yang tidak dijelaskan, nilai konstan tanpa satuan, serta penggunaan variabel dari sel lama. Komentar harus menerangkan alasan atau asumsi, bukan sekadar mengulang sintaks.

Validasi dan Interpretasi Hasil

Prinsip kerja yang digunakan adalah memisahkan input, proses, validasi, dan ekspor. Pemisahan ini membuat kesalahan lebih mudah dilacak dan mencegah hasil lama dianggap sebagai hasil dari parameter baru. Setiap langkah perlu meninggalkan bukti yang dapat dibaca kembali: nilai parameter, versi pustaka, status proses, nama berkas, serta pesan kesalahan bila terjadi kegagalan.

Aspek validasi	Kriteria lulus
Input	Tipe, bentuk, satuan, muatan, spin, dan jalur berkas dinyatakan eksplisit.
Lingkungan	Versi Python dan pustaka tercatat; backend CPU/GPU terdeteksi.

Proses	Status eksekusi atau konvergensi diperiksa sebelum hasil dipakai.
Numerik	Tidak ada NaN/inf; ukuran array dan jumlah atom konsisten.
Keluaran	Sel markdown, sel kode, output, dan metadata disimpan dan dapat dibuka kembali.
Interpretasi	Klaim tidak melampaui metode, basis set, atau data yang dihitung.
Interpretasi aman: Sebutkan metode, parameter, satuan, status konvergensi, dan keterbatasan saat menuliskan kesimpulan.	

Kesalahan Umum dan Troubleshooting

Risiko utama pada topik ini adalah state kernel tersembunyi dan urutan eksekusi. Karena itu, notebook harus memuat pemeriksaan eksplisit dan bukan hanya mengandalkan keluaran yang tampak masuk akal. Pembaca dianjurkan menjalankan notebook dari keadaan runtime bersih, membandingkan hasil dengan batas kewajaran, dan menyimpan artefak penting di lokasi persisten.

Gejala	Tindakan perbaikan
Sel dijalankan tidak berurutan	Restart session, lalu gunakan Run all dan periksa variabel prasyarat.
Pustaka tidak ditemukan	Jalankan kembali sel %pip, periksa nama paket, lalu restart bila dependensi inti berubah.
Input tampak benar tetapi gagal	Cetak tipe, shape, satuan, muatan, spin, dan beberapa baris pertama data.
State kernel tersembunyi dan urutan eksekusi	Tambahkan validasi sebelum proses utama dan hentikan eksekusi dengan pesan yang spesifik.
Hasil berbeda antar-runtime	Catat versi runtime/paket, random seed, parameter, dan perangkat yang digunakan.

Aturan troubleshooting: Perbaiki satu hal pada satu waktu, simpan pesan kesalahan lengkap, dan catat perubahan versi atau parameter.

Latihan Modifikasi dan Refleksi

171. Ubah satu parameter pada contoh Urutan eksekusi dan kebersihan notebook, jalankan ulang dari runtime bersih, dan jelaskan perubahan keluaran.
172. Tambahkan satu validasi baru yang mencegah state kernel tersembunyi dan urutan eksekusi.
173. Buat berkas ringkasan TXT atau JSON yang memuat input, status proses, waktu, dan hasil utama.
174. Jelaskan bagian mana yang merupakan fakta keluaran program dan bagian mana yang merupakan interpretasi ilmiah.
175. Susun satu kasus uji yang sengaja salah, lalu pastikan notebook menghasilkan pesan kesalahan yang informatif.

Refleksi: Tuliskan apa yang berubah setelah kode dimodifikasi, bukti apa yang menunjukkan keberhasilan, dan bagian mana yang masih memerlukan verifikasi eksternal.

Rubrik	Deskripsi
Skor 4	Kode berjalan dari runtime bersih, validasi lengkap, interpretasi tepat.
Skor 3	Kode berjalan dan hasil tersimpan, tetapi dokumentasi belum lengkap.
Skor 2	Kode hanya berjalan setelah intervensi atau state tertentu.
Skor 1	Kode belum berjalan dan diagnosis kesalahan belum sistematis.

2.2 Instalasi Pustaka dan Pemeriksaan Lingkungan

Instalasi Pustaka dan Pemeriksaan Lingkungan mengorganisasi tiga komponen yang saling bergantung. Pembahasan bergerak dari konsep, menuju prosedur, kemudian menghasilkan artefak yang dapat divalidasi dan disimpan.

Sub-subbab

Fokus

2.2.1 Instalasi dengan %pip	Instalasi pustaka, peningkatan versi, penguncian versi, dan pemasangan paket biner.
2.2.2 Pemeriksaan Python, sistem, RAM, dan versi pustaka	Penggunaan sys, platform, os, dan pemeriksaan kapasitas memori.
2.2.3 Deteksi CPU, GPU, CUDA, dan kompatibilitas	Penggunaan nvidia-smi, deteksi CUDA, jenis GPU, serta keputusan CPU atau GPU.
Produk belajar: Satu notebook terstruktur, satu berkas keluaran, satu catatan validasi, dan satu interpretasi singkat yang menyatakan batas hasil.	

Rencana Praktik dan Produk Belajar

176. Pelajari Instalasi dengan %pip dan jalankan kode dasar yang menyertainya.
177. Pelajari Pemeriksaan Python, sistem, RAM, dan versi pustaka dan jalankan kode dasar yang menyertainya.
178. Pelajari Deteksi CPU, GPU, CUDA, dan kompatibilitas dan jalankan kode dasar yang menyertainya.

Komponen	Kriteria
Waktu	60-120 menit, bergantung pada komputasi.
Input	Notebook baru atau salinan notebook bab.
Produk	Kode, output, berkas hasil, dan refleksi.
Uji	Restart session lalu Run all.
Dokumentasi	Versi, parameter, status, dan waktu eksekusi.
Kriteria selesai: Semua sel dapat dijalankan berurutan, pesan kesalahan ditangani, dan hasil utama tersimpan di lokasi persisten.	

2.2.1 Instalasi dengan %pip

Pada bagian Instalasi dengan %pip, perhatian tidak hanya diarahkan pada keberhasilan menjalankan kode, tetapi juga pada ketepatan asumsi, keterlacakan parameter, dan mutu

keluaran. Instalasi pustaka, peningkatan versi, penguncian versi, dan pemasangan paket biner. Hasil yang dituju berupa hasil yang dapat diperiksa dan disimpan.

Prinsip kerja yang digunakan adalah memisahkan input, proses, validasi, dan ekspor. Pemisahan ini membuat kesalahan lebih mudah dilacak dan mencegah hasil lama dianggap sebagai hasil dari parameter baru. Setiap langkah perlu meninggalkan bukti yang dapat dibaca kembali: nilai parameter, versi pustaka, status proses, nama berkas, serta pesan kesalahan bila terjadi kegagalan.

Tujuan operasional: Pembaca mampu menjelaskan Instalasi dengan %pip, menjalankan contoh minimal, memeriksa keberhasilan, dan menyimpan bukti hasil.

Komponen	Kriteria
Konsep	Instalasi pustaka, peningkatan versi, penguncian versi, dan pemasangan paket biner.
Input	Parameter dan data dinyatakan eksplisit.
Proses	Langkah dapat diulang dari runtime bersih.
Output	Nilai, berkas, atau visualisasi yang dapat diperiksa.
Batas	Interpretasi mengikuti metode dan konteks.

Landasan Teori dan Batas Penerapan

Instalasi dengan %pip merupakan simpul penting dalam alur kerja notebook ilmiah karena menentukan bagaimana prosedur komputasi disiapkan, dijalankan, dan diperiksa. Instalasi pustaka, peningkatan versi, penguncian versi, dan pemasangan paket biner. Hasil yang dituju berupa hasil yang dapat diperiksa dan disimpan.

Prinsip kerja yang digunakan adalah memisahkan input, proses, validasi, dan ekspor. Pemisahan ini membuat kesalahan lebih mudah dilacak dan mencegah hasil lama dianggap sebagai hasil dari parameter baru. Setiap langkah perlu meninggalkan bukti yang dapat dibaca kembali: nilai parameter, versi pustaka, status proses, nama berkas, serta pesan kesalahan bila terjadi kegagalan.

Risiko utama pada topik ini adalah parameter tidak terdokumentasi dan keluaran tidak tervalidasi. Karena itu, notebook harus memuat pemeriksaan eksplisit dan bukan hanya mengandalkan keluaran yang tampak masuk akal. Pembaca dianjurkan menjalankan

notebook dari keadaan runtime bersih, membandingkan hasil dengan batas kewajaran, dan menyimpan artefak penting di lokasi persisten.

Batas penerapan: Contoh pada buku berfungsi sebagai pola kerja. Ukuran molekul, biaya komputasi, versi pustaka, dan perangkat keras dapat mengubah waktu serta strategi yang diperlukan.

Prosedur Bertahap

Pembahasan Instalasi dengan %pip ditempatkan dalam konteks praktik Google Colab agar pembaca dapat menghubungkan konsep, prosedur, kode, dan interpretasi secara utuh. Instalasi pustaka, peningkatan versi, penguncian versi, dan pemasangan paket biner. Hasil yang dituju berupa hasil yang dapat diperiksa dan disimpan.

179. Tuliskan tujuan operasional Instalasi dengan %pip dan tentukan keluaran yang harus tersedia pada akhir sel.
180. Tetapkan parameter pada satu sel konfigurasi agar perubahan dapat ditelusuri dan tidak tersebar di banyak tempat.
181. Siapkan prosedur komputasi dengan input yang eksplisit; hindari nilai tersembunyi yang hanya tersimpan dalam state kernel.
182. Jalankan pemeriksaan prasyarat, termasuk keberadaan pustaka, bentuk data, tipe nilai, serta konsistensi satuan.
183. Eksekusi proses utama, tangkap status konvergensi atau keberhasilan, dan hentikan alur apabila syarat minimum tidak terpenuhi.
184. Simpan hasil yang dapat diperiksa dan disimpan dengan nama berkas informatif, kemudian tulis interpretasi yang membedakan fakta komputasi dan kesimpulan ilmiah.

Titik kontrol: Jangan melanjutkan ke interpretasi apabila validasi input atau status proses belum memenuhi kriteria.

Kode Dasar Siap Colab

Pada bagian Instalasi dengan %pip, perhatian tidak hanya diarahkan pada keberhasilan menjalankan kode, tetapi juga pada ketepatan asumsi, keterlacakan parameter, dan mutu keluaran. Instalasi pustaka, peningkatan versi, penguncian versi, dan pemasangan paket biner. Hasil yang dituju berupa hasil yang dapat diperiksa dan disimpan.

Sel kode contoh

```
%pip -q install --prefer-binary numpy pandas matplotlib scipy
%pip -q install --prefer-binary rdkit py3Dmol
%pip -q install --prefer-binary pyscf geometric nbformat nbclient

# Setelah perubahan paket besar, lakukan Runtime > Restart session.
```

Cara menjalankan: Tempatkan kode dalam satu atau beberapa sel sesuai komentar. Untuk baris %pip atau perintah shell, gunakan runtime Google Colab.

Penjelasan Baris Kode Penting

Potongan kode	Fungsi
%pip -q install --prefer-binary numpy pandas mat	Mencetak status agar pengguna tidak hanya bergantung pada keluaran visual.
%pip -q install --prefer-binary rdkit py3Dmol	Menambahkan pemeriksaan untuk mencegah hasil tidak valid diteruskan.
%pip -q install --prefer-binary pycsf geometric	Mencetak status agar pengguna tidak hanya bergantung pada keluaran visual.
# Setelah perubahan paket besar, lakukan Runtime	Mencetak status agar pengguna tidak hanya bergantung pada keluaran visual.

Nama variabel dipilih agar maknanya terbaca. Hindari singkatan yang tidak dijelaskan, nilai konstan tanpa satuan, serta penggunaan variabel dari sel lama. Komentar harus menerangkan alasan atau asumsi, bukan sekadar mengulang sintaks.

Validasi dan Interpretasi Hasil

Prinsip kerja yang digunakan adalah memisahkan input, proses, validasi, dan ekspor. Pemisahan ini membuat kesalahan lebih mudah dilacak dan mencegah hasil lama dianggap sebagai hasil dari parameter baru. Setiap langkah perlu meninggalkan bukti yang dapat dibaca kembali: nilai parameter, versi pustaka, status proses, nama berkas, serta pesan kesalahan bila terjadi kegagalan.

Aspek validasi	Kriteria lulus
Input	Tipe, bentuk, satuan, muatan, spin, dan jalur berkas dinyatakan eksplisit.
Lingkungan	Versi Python dan pustaka tercatat; backend CPU/GPU terdeteksi.
Proses	Status eksekusi atau konvergensi diperiksa sebelum hasil dipakai.

Numerik	Tidak ada NaN/inf; ukuran array dan jumlah atom konsisten.
Keluaran	Hasil yang dapat diperiksa dan disimpan disimpan dan dapat dibuka kembali.
Interpretasi	Klaim tidak melampaui metode, basis set, atau data yang dihitung.

Interpretasi aman: Sebutkan metode, parameter, satuan, status konvergensi, dan keterbatasan saat menuliskan kesimpulan.

Kesalahan Umum dan Troubleshooting

Risiko utama pada topik ini adalah parameter tidak terdokumentasi dan keluaran tidak tervalidasi. Karena itu, notebook harus memuat pemeriksaan eksplisit dan bukan hanya mengandalkan keluaran yang tampak masuk akal. Pembaca dianjurkan menjalankan notebook dari keadaan runtime bersih, membandingkan hasil dengan batas kewajaran, dan menyimpan artefak penting di lokasi persisten.

Gejala	Tindakan perbaikan
Sel dijalankan tidak berurutan	Restart session, lalu gunakan Run all dan periksa variabel prasyarat.
Pustaka tidak ditemukan	Jalankan kembali sel %pip, periksa nama paket, lalu restart bila dependensi inti berubah.
Input tampak benar tetapi gagal	Cetak tipe, shape, satuan, muatan, spin, dan beberapa baris pertama data.
Parameter tidak terdokumentasi dan keluaran tidak tervalidasi	Tambahkan validasi sebelum proses utama dan hentikan eksekusi dengan pesan yang spesifik.
Hasil berbeda antar-runtime	Catat versi runtime/paket, random seed, parameter, dan perangkat yang digunakan.

Aturan troubleshooting: Perbaiki satu hal pada satu waktu, simpan pesan kesalahan lengkap, dan catat perubahan versi atau parameter.

Latihan Modifikasi dan Refleksi

185. Ubah satu parameter pada contoh Instalasi dengan %pip, jalankan ulang dari runtime bersih, dan jelaskan perubahan keluaran.
186. Tambahkan satu validasi baru yang mencegah parameter tidak terdokumentasi dan keluaran tidak tervalidasi.
187. Buat berkas ringkasan TXT atau JSON yang memuat input, status proses, waktu, dan hasil utama.
188. Jelaskan bagian mana yang merupakan fakta keluaran program dan bagian mana yang merupakan interpretasi ilmiah.
189. Susun satu kasus uji yang sengaja salah, lalu pastikan notebook menghasilkan pesan kesalahan yang informatif.

Refleksi: Tuliskan apa yang berubah setelah kode dimodifikasi, bukti apa yang menunjukkan keberhasilan, dan bagian mana yang masih memerlukan verifikasi eksternal.

Rubrik	Deskripsi
Skor 4	Kode berjalan dari runtime bersih, validasi lengkap, interpretasi tepat.
Skor 3	Kode berjalan dan hasil tersimpan, tetapi dokumentasi belum lengkap.
Skor 2	Kode hanya berjalan setelah intervensi atau state tertentu.
Skor 1	Kode belum berjalan dan diagnosis kesalahan belum sistematis.

2.2.2 Pemeriksaan Python, sistem, RAM, dan versi pustaka

Pemeriksaan Python, sistem, RAM, dan versi pustaka merupakan simpul penting dalam alur kerja notebook ilmiah karena menentukan bagaimana skrip Python disiapkan, dijalankan, dan diperiksa. Penggunaan sys, platform, os, dan pemeriksaan kapasitas memori. Hasil yang dituju berupa nilai, teks, tabel, atau grafik.

Prinsip kerja yang digunakan adalah memisahkan input, proses, validasi, dan ekspor. Pemisahan ini membuat kesalahan lebih mudah dilacak dan mencegah hasil lama dianggap sebagai hasil dari parameter baru. Setiap langkah perlu meninggalkan bukti yang dapat dibaca kembali: nilai parameter, versi pustaka, status proses, nama berkas, serta pesan kesalahan bila terjadi kegagalan.

Tujuan operasional: Pembaca mampu menjelaskan Pemeriksaan Python, sistem, RAM, dan versi pustaka, menjalankan contoh minimal, memeriksa keberhasilan, dan menyimpan bukti hasil.

Komponen	Kriteria
Konsep	Penggunaan sys, platform, os, dan pemeriksaan kapasitas memori.
Input	Parameter dan data dinyatakan eksplisit.
Proses	Langkah dapat diulang dari runtime bersih.
Output	Nilai, berkas, atau visualisasi yang dapat diperiksa.
Batas	Interpretasi mengikuti metode dan konteks.

Landasan Teori dan Batas Penerapan

Pada bagian Pemeriksaan Python, sistem, RAM, dan versi pustaka, perhatian tidak hanya diarahkan pada keberhasilan menjalankan kode, tetapi juga pada ketepatan asumsi, keterlaksanaan parameter, dan mutu keluaran. Penggunaan sys, platform, os, dan pemeriksaan kapasitas memori. Hasil yang dituju berupa nilai, teks, tabel, atau grafik.

Prinsip kerja yang digunakan adalah memisahkan input, proses, validasi, dan ekspor. Pemisahan ini membuat kesalahan lebih mudah dilacak dan mencegah hasil lama dianggap sebagai hasil dari parameter baru. Setiap langkah perlu meninggalkan bukti yang dapat dibaca kembali: nilai parameter, versi pustaka, status proses, nama berkas, serta pesan kesalahan bila terjadi kegagalan.

Risiko utama pada topik ini adalah tipe data dan urutan operasi. Karena itu, notebook harus memuat pemeriksaan eksplisit dan bukan hanya mengandalkan keluaran yang tampak masuk akal. Pembaca dianjurkan menjalankan notebook dari keadaan runtime bersih, membandingkan hasil dengan batas kewajaran, dan menyimpan artefak penting di lokasi persisten.

Batas penerapan: Contoh pada buku berfungsi sebagai pola kerja. Ukuran molekul, biaya komputasi, versi pustaka, dan perangkat keras dapat mengubah waktu serta strategi yang diperlukan.

Prosedur Bertahap

Pada bagian Pemeriksaan Python, sistem, RAM, dan versi pustaka, perhatian tidak hanya diarahkan pada keberhasilan menjalankan kode, tetapi juga pada ketepatan asumsi, keterlacakan parameter, dan mutu keluaran. Penggunaan sys, platform, os, dan pemeriksaan kapasitas memori. Hasil yang dituju berupa nilai, teks, tabel, atau grafik.

190. Tuliskan tujuan operasional Pemeriksaan Python, sistem, RAM, dan versi pustaka dan tentukan keluaran yang harus tersedia pada akhir sel.
191. Tetapkan parameter pada satu sel konfigurasi agar perubahan dapat ditelusuri dan tidak tersebar di banyak tempat.
192. Siapkan skrip Python dengan input yang eksplisit; hindari nilai tersembunyi yang hanya tersimpan dalam state kernel.
193. Jalankan pemeriksaan prasyarat, termasuk keberadaan pustaka, bentuk data, tipe nilai, serta konsistensi satuan.
194. Eksekusi proses utama, tangkap status konvergensi atau keberhasilan, dan hentikan alur apabila syarat minimum tidak terpenuhi.
195. Simpan nilai, teks, tabel, atau grafik dengan nama berkas informatif, kemudian tulis interpretasi yang membedakan fakta komputasi dan kesimpulan ilmiah.

Titik kontrol: Jangan melanjutkan ke interpretasi apabila validasi input atau status proses belum memenuhi kriteria.

Kode Dasar Siap Colab

Pembahasan Pemeriksaan Python, sistem, RAM, dan versi pustaka ditempatkan dalam konteks praktik Google Colab agar pembaca dapat menghubungkan konsep, prosedur, kode, dan interpretasi secara utuh. Penggunaan sys, platform, os, dan pemeriksaan kapasitas memori. Hasil yang dituju berupa nilai, teks, tabel, atau grafik.

Sel kode contoh

```
import importlib.metadata as md
import os, platform, sys

for pkg in ["numpy", "pandas", "rdkit", "pyscf", "nbformat"]:
    try:
        print(f"{pkg:10s}: {md.version(pkg)}")
    except md.PackageNotFoundError:
        print(f"{pkg:10s}: belum terpasang")
print("Python:", sys.version.split()[0])
print("CPU:", os.cpu_count(), platform.machine())
```

Cara menjalankan: Tempatkan kode dalam satu atau beberapa sel sesuai komentar. Untuk baris %pip atau perintah shell, gunakan runtime Google Colab.

Penjelasan Baris Kode Penting

Potongan kode	Fungsi
<code>import importlib.metadata as md</code>	Menambahkan pemeriksaan untuk mencegah hasil tidak valid diteruskan.
<code>import os, platform, sys</code>	Mencetak status agar pengguna tidak hanya bergantung pada keluaran visual.
<code>for pkg in ["numpy", "pandas", "rdkit", "pyscf",</code>	Menjalankan operasi utama dan menyimpan hasil pada variabel yang jelas.
<code>try:</code>	Mencetak status agar pengguna tidak hanya bergantung pada keluaran visual.
<code>print(f'{pkg:10s}: {md.version(pkg)}')</code>	Mencetak status agar pengguna tidak hanya bergantung pada keluaran visual.
<code>except md.PackageNotFoundError:</code>	Mendefinisikan input atau parameter yang akan digunakan pada langkah berikutnya.
<code>print(f'{pkg:10s}: belum terpasang')</code>	Menjalankan operasi utama dan menyimpan hasil pada variabel yang jelas.
<code>print("Python:", sys.version.split()[0])</code>	Mendefinisikan input atau parameter yang akan digunakan pada langkah berikutnya.

Nama variabel dipilih agar maknanya terbaca. Hindari singkatan yang tidak dijelaskan, nilai konstan tanpa satuan, serta penggunaan variabel dari sel lama. Komentar harus menerangkan alasan atau asumsi, bukan sekadar mengulang sintaks.

Validasi dan Interpretasi Hasil

Prinsip kerja yang digunakan adalah memisahkan input, proses, validasi, dan ekspor. Pemisahan ini membuat kesalahan lebih mudah dilacak dan mencegah hasil lama dianggap sebagai hasil dari parameter baru. Setiap langkah perlu meninggalkan bukti yang dapat dibaca kembali: nilai parameter, versi pustaka, status proses, nama berkas, serta pesan kesalahan bila terjadi kegagalan.

Aspek validasi	Kriteria lulus
Input	Tipe, bentuk, satuan, muatan, spin, dan jalur berkas dinyatakan eksplisit.
Lingkungan	Versi Python dan pustaka tercatat; backend CPU/GPU terdeteksi.
Proses	Status eksekusi atau konvergensi diperiksa sebelum hasil dipakai.
Numerik	Tidak ada NaN/inf; ukuran array dan jumlah atom konsisten.
Keluaran	Nilai, teks, tabel, atau grafik disimpan dan dapat dibuka kembali.
Interpretasi	Klaim tidak melampaui metode, basis set, atau data yang dihitung.
Interpretasi aman: Sebutkan metode, parameter, satuan, status konvergensi, dan keterbatasan saat menuliskan kesimpulan.	

Kesalahan Umum dan Troubleshooting

Risiko utama pada topik ini adalah tipe data dan urutan operasi. Karena itu, notebook harus memuat pemeriksaan eksplisit dan bukan hanya mengandalkan keluaran yang tampak masuk akal. Pembaca dianjurkan menjalankan notebook dari keadaan runtime bersih, membandingkan hasil dengan batas kewajaran, dan menyimpan artefak penting di lokasi persisten.

Gejala	Tindakan perbaikan
Sel dijalankan tidak berurutan	Restart session, lalu gunakan Run all dan periksa variabel prasyarat.
Pustaka tidak ditemukan	Jalankan kembali sel %pip, periksa nama paket, lalu restart bila dependensi inti berubah.

Input tampak benar tetapi gagal	Cetak tipe, shape, satuan, muatan, spin, dan beberapa baris pertama data.
Tipe data dan urutan operasi	Tambahkan validasi sebelum proses utama dan hentikan eksekusi dengan pesan yang spesifik.
Hasil berbeda antar-runtime	Catat versi runtime/paket, random seed, parameter, dan perangkat yang digunakan.
Aturan troubleshooting: Perbaiki satu hal pada satu waktu, simpan pesan kesalahan lengkap, dan catat perubahan versi atau parameter.	

Latihan Modifikasi dan Refleksi

196. Ubah satu parameter pada contoh Pemeriksaan Python, sistem, RAM, dan versi pustaka, jalankan ulang dari runtime bersih, dan jelaskan perubahan keluaran.
197. Tambahkan satu validasi baru yang mencegah tipe data dan urutan operasi.
198. Buat berkas ringkasan TXT atau JSON yang memuat input, status proses, waktu, dan hasil utama.
199. Jelaskan bagian mana yang merupakan fakta keluaran program dan bagian mana yang merupakan interpretasi ilmiah.
200. Susun satu kasus uji yang sengaja salah, lalu pastikan notebook menghasilkan pesan kesalahan yang informatif.

Refleksi: Tuliskan apa yang berubah setelah kode dimodifikasi, bukti apa yang menunjukkan keberhasilan, dan bagian mana yang masih memerlukan verifikasi eksternal.

Rubrik	Deskripsi
Skor 4	Kode berjalan dari runtime bersih, validasi lengkap, interpretasi tepat.
Skor 3	Kode berjalan dan hasil tersimpan, tetapi dokumentasi belum lengkap.
Skor 2	Kode hanya berjalan setelah intervensi atau state tertentu.
Skor 1	Kode belum berjalan dan diagnosis kesalahan belum sistematis.

2.2.3 Deteksi CPU, GPU, CUDA, dan kompatibilitas

Pembahasan Deteksi CPU, GPU, CUDA, dan kompatibilitas ditempatkan dalam konteks praktik Google Colab agar pembaca dapat menghubungkan konsep, prosedur, kode, dan interpretasi secara utuh. Penggunaan `nvidia-smi`, deteksi CUDA, jenis GPU, serta keputusan CPU atau GPU. Hasil yang dituju berupa informasi perangkat, waktu eksekusi, dan status penggunaan GPU.

Prinsip kerja yang digunakan adalah memisahkan input, proses, validasi, dan ekspor. Pemisahan ini membuat kesalahan lebih mudah dilacak dan mencegah hasil lama dianggap sebagai hasil dari parameter baru. Setiap langkah perlu meninggalkan bukti yang dapat dibaca kembali: nilai parameter, versi pustaka, status proses, nama berkas, serta pesan kesalahan bila terjadi kegagalan.

Tujuan operasional: Pembaca mampu menjelaskan Deteksi CPU, GPU, CUDA, dan kompatibilitas, menjalankan contoh minimal, memeriksa keberhasilan, dan menyimpan bukti hasil.

Komponen	Kriteria
Konsep	Penggunaan <code>nvidia-smi</code> , deteksi CUDA, jenis GPU, serta keputusan CPU atau GPU.
Input	Parameter dan data dinyatakan eksplisit.
Proses	Langkah dapat diulang dari runtime bersih.
Output	Nilai, berkas, atau visualisasi yang dapat diperiksa.
Batas	Interpretasi mengikuti metode dan konteks.

Landasan Teori dan Batas Penerapan

Pembahasan Deteksi CPU, GPU, CUDA, dan kompatibilitas ditempatkan dalam konteks praktik Google Colab agar pembaca dapat menghubungkan konsep, prosedur, kode, dan interpretasi secara utuh. Penggunaan `nvidia-smi`, deteksi CUDA, jenis GPU, serta keputusan CPU atau GPU. Hasil yang dituju berupa informasi perangkat, waktu eksekusi, dan status penggunaan GPU.

Prinsip kerja yang digunakan adalah memisahkan input, proses, validasi, dan ekspor. Pemisahan ini membuat kesalahan lebih mudah dilacak dan mencegah hasil lama dianggap sebagai hasil dari parameter baru. Setiap langkah perlu meninggalkan bukti

yang dapat dibaca kembali: nilai parameter, versi pustaka, status proses, nama berkas, serta pesan kesalahan bila terjadi kegagalan.

Risiko utama pada topik ini adalah paket CUDA tidak kompatibel atau GPU tidak benar-benar digunakan. Karena itu, notebook harus memuat pemeriksaan eksplisit dan bukan hanya mengandalkan keluaran yang tampak masuk akal. Pembaca dianjurkan menjalankan notebook dari keadaan runtime bersih, membandingkan hasil dengan batas kewajaran, dan menyimpan artefak penting di lokasi persisten.

Batas penerapan: Contoh pada buku berfungsi sebagai pola kerja. Ukuran molekul, biaya komputasi, versi pustaka, dan perangkat keras dapat mengubah waktu serta strategi yang diperlukan.

Prosedur Bertahap

Deteksi CPU, GPU, CUDA, dan kompatibilitas merupakan simpul penting dalam alur kerja notebook ilmiah karena menentukan bagaimana backend komputasi disiapkan, dijalankan, dan diperiksa. Penggunaan `nvidia-smi`, deteksi CUDA, jenis GPU, serta keputusan CPU atau GPU. Hasil yang dituju berupa informasi perangkat, waktu eksekusi, dan status penggunaan GPU.

201. Tuliskan tujuan operasional Deteksi CPU, GPU, CUDA, dan kompatibilitas dan tentukan keluaran yang harus tersedia pada akhir sel.
202. Tetapkan parameter pada satu sel konfigurasi agar perubahan dapat ditelusuri dan tidak tersebar di banyak tempat.
203. Siapkan backend komputasi dengan input yang eksplisit; hindari nilai tersembunyi yang hanya tersimpan dalam state kernel.
204. Jalankan pemeriksaan prasyarat, termasuk keberadaan pustaka, bentuk data, tipe nilai, serta konsistensi satuan.
205. Eksekusi proses utama, tangkap status konvergensi atau keberhasilan, dan hentikan alur apabila syarat minimum tidak terpenuhi.
206. Simpan informasi perangkat, waktu eksekusi, dan status penggunaan GPU dengan nama berkas informatif, kemudian tulis interpretasi yang membedakan fakta komputasi dan kesimpulan ilmiah.

Titik kontrol: Jangan melanjutkan ke interpretasi apabila validasi input atau status proses belum memenuhi kriteria.

Kode Dasar Siap Colab

Pada bagian Deteksi CPU, GPU, CUDA, dan kompatibilitas, perhatian tidak hanya diarahkan pada keberhasilan menjalankan kode, tetapi juga pada ketepatan asumsi, keterlacakan parameter, dan mutu keluaran. Penggunaan `nvidia-smi`, deteksi CUDA, jenis GPU, serta keputusan CPU atau GPU. Hasil yang dituju berupa informasi perangkat, waktu eksekusi, dan status penggunaan GPU.

Sel kode contoh

```
import os, shutil, subprocess

print("CPU logis:", os.cpu_count())
if shutil.which("nvidia-smi"):
    subprocess.run(["nvidia-smi", "--query-
gpu=name,driver_version,memory.total",
                    "--format=csv,noheader"], check=False)
else:
    print("GPU NVIDIA tidak terdeteksi; gunakan mode CPU.")
```

Cara menjalankan: Tempatkan kode dalam satu atau beberapa sel sesuai komentar. Untuk baris %pip atau perintah shell, gunakan runtime Google Colab.

Penjelasan Baris Kode Penting

Potongan kode	Fungsi
import os, shutil, subprocess	Menjalankan operasi utama dan menyimpan hasil pada variabel yang jelas.
print("CPU logis:", os.cpu_count())	Mencetak status agar pengguna tidak hanya bergantung pada keluaran visual.
if shutil.which("nvidia-smi"):	Menambahkan pemeriksaan untuk mencegah hasil tidak valid diteruskan.
subprocess.run(["nvidia-smi", "--query- gpu=n	Mendefinisikan input atau parameter yang akan digunakan pada langkah berikutnya.
"--format=csv,noheader"], ch	Menjalankan operasi utama dan menyimpan hasil pada variabel yang jelas.
else:	Menjalankan operasi utama dan menyimpan hasil pada variabel yang jelas.
print("GPU NVIDIA tidak terdeteksi; gunakan	Mencetak status agar pengguna tidak hanya bergantung pada keluaran visual.

Nama variabel dipilih agar maknanya terbaca. Hindari singkatan yang tidak dijelaskan, nilai konstan tanpa satuan, serta penggunaan variabel dari sel lama. Komentar harus menerangkan alasan atau asumsi, bukan sekadar mengulang sintaks.

Validasi dan Interpretasi Hasil

Prinsip kerja yang digunakan adalah memisahkan input, proses, validasi, dan ekspor. Pemisahan ini membuat kesalahan lebih mudah dilacak dan mencegah hasil lama dianggap sebagai hasil dari parameter baru. Setiap langkah perlu meninggalkan bukti yang dapat dibaca kembali: nilai parameter, versi pustaka, status proses, nama berkas, serta pesan kesalahan bila terjadi kegagalan.

Aspek validasi	Kriteria lulus
Input	Tipe, bentuk, satuan, muatan, spin, dan jalur berkas dinyatakan eksplisit.
Lingkungan	Versi Python dan pustaka tercatat; backend CPU/GPU terdeteksi.
Proses	Status eksekusi atau konvergensi diperiksa sebelum hasil dipakai.
Numerik	Tidak ada NaN/inf; ukuran array dan jumlah atom konsisten.
Keluaran	Informasi perangkat, waktu eksekusi, dan status penggunaan gpu disimpan dan dapat dibuka kembali.
Interpretasi	Klaim tidak melampaui metode, basis set, atau data yang dihitung.
Interpretasi aman: Sebutkan metode, parameter, satuan, status konvergensi, dan keterbatasan saat menuliskan kesimpulan.	

Kesalahan Umum dan Troubleshooting

Risiko utama pada topik ini adalah paket CUDA tidak kompatibel atau GPU tidak benar-benar digunakan. Karena itu, notebook harus memuat pemeriksaan eksplisit dan bukan hanya mengandalkan keluaran yang tampak masuk akal. Pembaca dianjurkan menjalankan notebook dari keadaan runtime bersih, membandingkan hasil dengan batas kewajaran, dan menyimpan artefak penting di lokasi persisten.

Gejala

Tindakan perbaikan

Sel dijalankan tidak berurutan	Restart session, lalu gunakan Run all dan periksa variabel prasyarat.
Pustaka tidak ditemukan	Jalankan kembali sel %pip, periksa nama paket, lalu restart bila dependensi inti berubah.
Input tampak benar tetapi gagal	Cetak tipe, shape, satuan, muatan, spin, dan beberapa baris pertama data.
Paket cuda tidak kompatibel atau gpu tidak benar-benar digunakan	Tambahkan validasi sebelum proses utama dan hentikan eksekusi dengan pesan yang spesifik.
Hasil berbeda antar-runtime	Catat versi runtime/paket, random seed, parameter, dan perangkat yang digunakan.
Aturan troubleshooting: Perbaiki satu hal pada satu waktu, simpan pesan kesalahan lengkap, dan catat perubahan versi atau parameter.	

Latihan Modifikasi dan Refleksi

207. Ubah satu parameter pada contoh Deteksi CPU, GPU, CUDA, dan kompatibilitas, jalankan ulang dari runtime bersih, dan jelaskan perubahan keluaran.
208. Tambahkan satu validasi baru yang mencegah paket CUDA tidak kompatibel atau GPU tidak benar-benar digunakan.
209. Buat berkas ringkasan TXT atau JSON yang memuat input, status proses, waktu, dan hasil utama.
210. Jelaskan bagian mana yang merupakan fakta keluaran program dan bagian mana yang merupakan interpretasi ilmiah.
211. Susun satu kasus uji yang sengaja salah, lalu pastikan notebook menghasilkan pesan kesalahan yang informatif.

Refleksi: Tuliskan apa yang berubah setelah kode dimodifikasi, bukti apa yang menunjukkan keberhasilan, dan bagian mana yang masih memerlukan verifikasi eksternal.

Rubrik	Deskripsi
Skor 4	Kode berjalan dari runtime bersih, validasi lengkap, interpretasi tepat.

Skor 3	Kode berjalan dan hasil tersimpan, tetapi dokumentasi belum lengkap.
Skor 2	Kode hanya berjalan setelah intervensi atau state tertentu.
Skor 1	Kode belum berjalan dan diagnosis kesalahan belum sistematis.

2.3 Input, Output, dan Pengelolaan Berkas

Input, Output, dan Pengelolaan Berkas mengorganisasi tiga komponen yang saling bergantung. Pembahasan bergerak dari konsep, menuju prosedur, kemudian menghasilkan artefak yang dapat divalidasi dan disimpan.

Sub-subbab	Fokus
2.3.1 Membaca dan menulis TXT, CSV, JSON, dan XYZ	Standar encoding, struktur file, validasi isi, dan penamaan berkas.
2.3.2 Google Drive dan penyimpanan hasil	Mount Drive, pembuatan folder proyek, jalur kerja, dan pemulihan hasil setelah runtime terputus.
2.3.3 Membuat laporan dan arsip ZIP	Menggabungkan hasil, log, grafik, struktur molekul, dan ringkasan ke dalam satu arsip.
Produk belajar: Satu notebook terstruktur, satu berkas keluaran, satu catatan validasi, dan satu interpretasi singkat yang menyatakan batas hasil.	

Rencana Praktik dan Produk Belajar

212. Pelajari Membaca dan menulis TXT, CSV, JSON, dan XYZ dan jalankan kode dasar yang menyertainya.
213. Pelajari Google Drive dan penyimpanan hasil dan jalankan kode dasar yang menyertainya.
214. Pelajari Membuat laporan dan arsip ZIP dan jalankan kode dasar yang menyertainya.

Komponen	Kriteria
Waktu	60-120 menit, bergantung pada komputasi.

Input	Notebook baru atau salinan notebook bab.
Produk	Kode, output, berkas hasil, dan refleksi.
Uji	Restart session lalu Run all.
Dokumentasi	Versi, parameter, status, dan waktu eksekusi.
Kriteria selesai: Semua sel dapat dijalankan berurutan, pesan kesalahan ditangani, dan hasil utama tersimpan di lokasi persisten.	

2.3.1 Membaca dan menulis TXT, CSV, JSON, dan XYZ

Membaca dan menulis TXT, CSV, JSON, dan XYZ merupakan simpul penting dalam alur kerja notebook ilmiah karena menentukan bagaimana artefak ilmiah disiapkan, dijalankan, dan diperiksa. Standar encoding, struktur file, validasi isi, dan penamaan berkas. Hasil yang dituju berupa laporan validasi dan jejak versi.

Prinsip kerja yang digunakan adalah memisahkan input, proses, validasi, dan ekspor. Pemisahan ini membuat kesalahan lebih mudah dilacak dan mencegah hasil lama dianggap sebagai hasil dari parameter baru. Setiap langkah perlu meninggalkan bukti yang dapat dibaca kembali: nilai parameter, versi pustaka, status proses, nama berkas, serta pesan kesalahan bila terjadi kegagalan.

Tujuan operasional: Pembaca mampu menjelaskan Membaca dan menulis TXT, CSV, JSON, dan XYZ, menjalankan contoh minimal, memeriksa keberhasilan, dan menyimpan bukti hasil.

Komponen	Kriteria
Konsep	Standar encoding, struktur file, validasi isi, dan penamaan berkas.
Input	Parameter dan data dinyatakan eksplisit.
Proses	Langkah dapat diulang dari runtime bersih.
Output	Nilai, berkas, atau visualisasi yang dapat diperiksa.
Batas	Interpretasi mengikuti metode dan konteks.

Landasan Teori dan Batas Penerapan

Pada bagian Membaca dan menulis TXT, CSV, JSON, dan XYZ, perhatian tidak hanya diarahkan pada keberhasilan menjalankan kode, tetapi juga pada ketepatan asumsi, keterlacakan parameter, dan mutu keluaran. Standar encoding, struktur file, validasi isi, dan penamaan berkas. Hasil yang dituju berupa laporan validasi dan jejak versi.

Prinsip kerja yang digunakan adalah memisahkan input, proses, validasi, dan ekspor. Pemisahan ini membuat kesalahan lebih mudah dilacak dan mencegah hasil lama dianggap sebagai hasil dari parameter baru. Setiap langkah perlu meninggalkan bukti yang dapat dibaca kembali: nilai parameter, versi pustaka, status proses, nama berkas, serta pesan kesalahan bila terjadi kegagalan.

Risiko utama pada topik ini adalah notebook hanya berhasil pada state tertentu atau hasil tidak dapat diulang. Karena itu, notebook harus memuat pemeriksaan eksplisit dan bukan hanya mengandalkan keluaran yang tampak masuk akal. Pembaca dianjurkan menjalankan notebook dari keadaan runtime bersih, membandingkan hasil dengan batas kewajaran, dan menyimpan artefak penting di lokasi persisten.

Batas penerapan: Contoh pada buku berfungsi sebagai pola kerja. Ukuran molekul, biaya komputasi, versi pustaka, dan perangkat keras dapat mengubah waktu serta strategi yang diperlukan.

Prosedur Bertahap

Pembahasan Membaca dan menulis TXT, CSV, JSON, dan XYZ ditempatkan dalam konteks praktik Google Colab agar pembaca dapat menghubungkan konsep, prosedur, kode, dan interpretasi secara utuh. Standar encoding, struktur file, validasi isi, dan penamaan berkas. Hasil yang dituju berupa laporan validasi dan jejak versi.

215. Tuliskan tujuan operasional Membaca dan menulis TXT, CSV, JSON, dan XYZ dan tentukan keluaran yang harus tersedia pada akhir sel.
216. Tetapkan parameter pada satu sel konfigurasi agar perubahan dapat ditelusuri dan tidak tersebar di banyak tempat.
217. Siapkan artefak ilmiah dengan input yang eksplisit; hindari nilai tersembunyi yang hanya tersimpan dalam state kernel.
218. Jalankan pemeriksaan prasyarat, termasuk keberadaan pustaka, bentuk data, tipe nilai, serta konsistensi satuan.
219. Eksekusi proses utama, tangkap status konvergensi atau keberhasilan, dan hentikan alur apabila syarat minimum tidak terpenuhi.
220. Simpan laporan validasi dan jejak versi dengan nama berkas informatif, kemudian tulis interpretasi yang membedakan fakta komputasi dan kesimpulan ilmiah.

Titik kontrol: Jangan melanjutkan ke interpretasi apabila validasi input atau status proses belum memenuhi kriteria.

Kode Dasar Siap Colab

Pada bagian Membaca dan menulis TXT, CSV, JSON, dan XYZ, perhatian tidak hanya diarahkan pada keberhasilan menjalankan kode, tetapi juga pada ketepatan asumsi, keterlacakan parameter, dan mutu keluaran. Standar encoding, struktur file, validasi isi, dan penamaan berkas. Hasil yang dituju berupa laporan validasi dan jejak versi.

Sel kode contoh

```
from pathlib import Path
import json, pandas as pd

Path("catatan.txt").write_text("UTF-8 dan jalur relatif", encoding="utf-8")
pd.DataFrame({"atom":["O","H","H"], "z":[0.0,0.59,0.59]}).to_csv("atom.csv",
index=False)
Path("meta.json").write_text(json.dumps({"charge":0,"spin":0}, indent=2),
encoding="utf-8")
print(Path("catatan.txt").read_text(encoding="utf-8"))
```

Cara menjalankan: Tempatkan kode dalam satu atau beberapa sel sesuai komentar. Untuk baris %pip atau perintah shell, gunakan runtime Google Colab.

Penjelasan Baris Kode Penting

Potongan kode	Fungsi
from pathlib import Path	Mendefinisikan input atau parameter yang akan digunakan pada langkah berikutnya.
import json, pandas as pd	Mencetak status agar pengguna tidak hanya bergantung pada keluaran visual.
Path("catatan.txt").write_text("UTF-8 dan jalur	Mendefinisikan input atau parameter yang akan digunakan pada langkah berikutnya.
pd.DataFrame({"atom":["O","H","H"], "z":[0.0,0.5	Mencetak status agar pengguna tidak hanya bergantung pada keluaran visual.

<code>Path("meta.json").write_text(json.dumps({"charge</code>	Menambahkan pemeriksaan untuk mencegah hasil tidak valid diteruskan.
<code>print(Path("catatan.txt").read_text(encoding="ut</code>	Mendefinisikan input atau parameter yang akan digunakan pada langkah berikutnya.

Nama variabel dipilih agar maknanya terbaca. Hindari singkatan yang tidak dijelaskan, nilai konstan tanpa satuan, serta penggunaan variabel dari sel lama. Komentar harus menerangkan alasan atau asumsi, bukan sekadar mengulang sintaks.

Validasi dan Interpretasi Hasil

Prinsip kerja yang digunakan adalah memisahkan input, proses, validasi, dan ekspor. Pemisahan ini membuat kesalahan lebih mudah dilacak dan mencegah hasil lama dianggap sebagai hasil dari parameter baru. Setiap langkah perlu meninggalkan bukti yang dapat dibaca kembali: nilai parameter, versi pustaka, status proses, nama berkas, serta pesan kesalahan bila terjadi kegagalan.

Aspek validasi	Kriteria lulus
Input	Tipe, bentuk, satuan, muatan, spin, dan jalur berkas dinyatakan eksplisit.
Lingkungan	Versi Python dan pustaka tercatat; backend CPU/GPU terdeteksi.
Proses	Status eksekusi atau konvergensi diperiksa sebelum hasil dipakai.
Numerik	Tidak ada NaN/inf; ukuran array dan jumlah atom konsisten.
Keluaran	Laporan validasi dan jejak versi disimpan dan dapat dibuka kembali.
Interpretasi	Klaim tidak melampaui metode, basis set, atau data yang dihitung.
Interpretasi aman: Sebutkan metode, parameter, satuan, status konvergensi, dan keterbatasan saat menuliskan kesimpulan.	

Kesalahan Umum dan Troubleshooting

Risiko utama pada topik ini adalah notebook hanya berhasil pada state tertentu atau hasil tidak dapat diulang. Karena itu, notebook harus memuat pemeriksaan eksplisit dan bukan hanya mengandalkan keluaran yang tampak masuk akal. Pembaca dianjurkan menjalankan notebook dari keadaan runtime bersih, membandingkan hasil dengan batas kewajaran, dan menyimpan artefak penting di lokasi persisten.

Gejala	Tindakan perbaikan
Sel dijalankan tidak berurutan	Restart session, lalu gunakan Run all dan periksa variabel prasyarat.
Pustaka tidak ditemukan	Jalankan kembali sel %pip, periksa nama paket, lalu restart bila dependensi inti berubah.
Input tampak benar tetapi gagal	Cetak tipe, shape, satuan, muatan, spin, dan beberapa baris pertama data.
Notebook hanya berhasil pada state tertentu atau hasil tidak dapat diulang	Tambahkan validasi sebelum proses utama dan hentikan eksekusi dengan pesan yang spesifik.
Hasil berbeda antar-runtime	Catat versi runtime/paket, random seed, parameter, dan perangkat yang digunakan.
Aturan troubleshooting: Perbaiki satu hal pada satu waktu, simpan pesan kesalahan lengkap, dan catat perubahan versi atau parameter.	

Latihan Modifikasi dan Refleksi

221. Ubah satu parameter pada contoh Membaca dan menulis TXT, CSV, JSON, dan XYZ, jalankan ulang dari runtime bersih, dan jelaskan perubahan keluaran.
222. Tambahkan satu validasi baru yang mencegah notebook hanya berhasil pada state tertentu atau hasil tidak dapat diulang.
223. Buat berkas ringkasan TXT atau JSON yang memuat input, status proses, waktu, dan hasil utama.
224. Jelaskan bagian mana yang merupakan fakta keluaran program dan bagian mana yang merupakan interpretasi ilmiah.
225. Susun satu kasus uji yang sengaja salah, lalu pastikan notebook menghasilkan pesan kesalahan yang informatif.

Refleksi: Tuliskan apa yang berubah setelah kode dimodifikasi, bukti apa yang menunjukkan keberhasilan, dan bagian mana yang masih memerlukan verifikasi eksternal.

Rubrik	Deskripsi
Skor 4	Kode berjalan dari runtime bersih, validasi lengkap, interpretasi tepat.
Skor 3	Kode berjalan dan hasil tersimpan, tetapi dokumentasi belum lengkap.
Skor 2	Kode hanya berjalan setelah intervensi atau state tertentu.
Skor 1	Kode belum berjalan dan diagnosis kesalahan belum sistematis.

2.3.2 Google Drive dan penyimpanan hasil

Pembahasan Google Drive dan penyimpanan hasil ditempatkan dalam konteks praktik Google Colab agar pembaca dapat menghubungkan konsep, prosedur, kode, dan interpretasi secara utuh. Mount Drive, pembuatan folder proyek, jalur kerja, dan pemulihan hasil setelah runtime terputus. Hasil yang dituju berupa informasi lingkungan dan berkas hasil.

Prinsip kerja yang digunakan adalah memisahkan input, proses, validasi, dan ekspor. Pemisahan ini membuat kesalahan lebih mudah dilacak dan mencegah hasil lama dianggap sebagai hasil dari parameter baru. Setiap langkah perlu meninggalkan bukti yang dapat dibaca kembali: nilai parameter, versi pustaka, status proses, nama berkas, serta pesan kesalahan bila terjadi kegagalan.

Tujuan operasional: Pembaca mampu menjelaskan Google Drive dan penyimpanan hasil, menjalankan contoh minimal, memeriksa keberhasilan, dan menyimpan bukti hasil.

Komponen	Kriteria
Konsep	Mount Drive, pembuatan folder proyek, jalur kerja, dan pemulihan hasil setelah runtime terputus.
Input	Parameter dan data dinyatakan eksplisit.

Proses	Langkah dapat diulang dari runtime bersih.
Output	Nilai, berkas, atau visualisasi yang dapat diperiksa.
Batas	Interpretasi mengikuti metode dan konteks.

Landasan Teori dan Batas Penerapan

Pada bagian Google Drive dan penyimpanan hasil, perhatian tidak hanya diarahkan pada keberhasilan menjalankan kode, tetapi juga pada ketepatan asumsi, keterlacakan parameter, dan mutu keluaran. Mount Drive, pembuatan folder proyek, jalur kerja, dan pemulihan hasil setelah runtime terputus. Hasil yang dituju berupa informasi lingkungan dan berkas hasil.

Prinsip kerja yang digunakan adalah memisahkan input, proses, validasi, dan ekspor. Pemisahan ini membuat kesalahan lebih mudah dilacak dan mencegah hasil lama dianggap sebagai hasil dari parameter baru. Setiap langkah perlu meninggalkan bukti yang dapat dibaca kembali: nilai parameter, versi pustaka, status proses, nama berkas, serta pesan kesalahan bila terjadi kegagalan.

Risiko utama pada topik ini adalah runtime terputus, sumber daya berubah, atau berkas sementara hilang. Karena itu, notebook harus memuat pemeriksaan eksplisit dan bukan hanya mengandalkan keluaran yang tampak masuk akal. Pembaca dianjurkan menjalankan notebook dari keadaan runtime bersih, membandingkan hasil dengan batas kewajaran, dan menyimpan artefak penting di lokasi persisten.

Batas penerapan: Contoh pada buku berfungsi sebagai pola kerja. Ukuran molekul, biaya komputasi, versi pustaka, dan perangkat keras dapat mengubah waktu serta strategi yang diperlukan.

Prosedur Bertahap

Pembahasan Google Drive dan penyimpanan hasil ditempatkan dalam konteks praktik Google Colab agar pembaca dapat menghubungkan konsep, prosedur, kode, dan interpretasi secara utuh. Mount Drive, pembuatan folder proyek, jalur kerja, dan pemulihan hasil setelah runtime terputus. Hasil yang dituju berupa informasi lingkungan dan berkas hasil.

226. Tuliskan tujuan operasional Google Drive dan penyimpanan hasil dan tentukan keluaran yang harus tersedia pada akhir sel.
227. Tetapkan parameter pada satu sel konfigurasi agar perubahan dapat ditelusuri dan tidak tersebar di banyak tempat.

228. Siapkan runtime Google Colab dengan input yang eksplisit; hindari nilai tersembunyi yang hanya tersimpan dalam state kernel.
229. Jalankan pemeriksaan prasyarat, termasuk keberadaan pustaka, bentuk data, tipe nilai, serta konsistensi satuan.
230. Eksekusi proses utama, tangkap status konvergensi atau keberhasilan, dan hentikan alur apabila syarat minimum tidak terpenuhi.
231. Simpan informasi lingkungan dan berkas hasil dengan nama berkas informatif, kemudian tulis interpretasi yang membedakan fakta komputasi dan kesimpulan ilmiah.

Titik kontrol: Jangan melanjutkan ke interpretasi apabila validasi input atau status proses belum memenuhi kriteria.

Kode Dasar Siap Colab

Pembahasan Google Drive dan penyimpanan hasil ditempatkan dalam konteks praktik Google Colab agar pembaca dapat menghubungkan konsep, prosedur, kode, dan interpretasi secara utuh. Mount Drive, pembuatan folder proyek, jalur kerja, dan pemulihan hasil setelah runtime terputus. Hasil yang dituju berupa informasi lingkungan dan berkas hasil.

Sel kode contoh

```
from google.colab import drive
from pathlib import Path

drive.mount("/content/drive")
ROOT = Path("/content/drive/MyDrive/Colab_Kimia")
ROOT.mkdir(parents=True, exist_ok=True)
( ROOT / "README.txt" ).write_text("Folder hasil proyek", encoding="utf-8")
print("Hasil persisten:", ROOT)
```

Cara menjalankan: Tempatkan kode dalam satu atau beberapa sel sesuai komentar. Untuk baris %pip atau perintah shell, gunakan runtime Google Colab.

Penjelasan Baris Kode Penting

Potongan kode	Fungsi
<code>from google.colab import drive</code>	Menjalankan operasi utama dan menyimpan hasil pada variabel yang jelas.

<code>from pathlib import Path</code>	Mencetak status agar pengguna tidak hanya bergantung pada keluaran visual.
<code>drive.mount("/content/drive")</code>	Mencetak status agar pengguna tidak hanya bergantung pada keluaran visual.
<code>ROOT = Path("/content/drive/MyDrive/Colab_Kimia")</code>	Mencetak status agar pengguna tidak hanya bergantung pada keluaran visual.
<code>ROOT.mkdir(parents=True, exist_ok=True)</code>	Menjalankan operasi utama dan menyimpan hasil pada variabel yang jelas.
<code>(ROOT / "README.txt").write_text("Folder hasil</code>	Menambahkan pemeriksaan untuk mencegah hasil tidak valid diteruskan.
<code>print("Hasil persisten:", ROOT)</code>	Mendefinisikan input atau parameter yang akan digunakan pada langkah berikutnya.

Nama variabel dipilih agar maknanya terbaca. Hindari singkatan yang tidak dijelaskan, nilai konstan tanpa satuan, serta penggunaan variabel dari sel lama. Komentar harus menerangkan alasan atau asumsi, bukan sekadar mengulang sintaks.

Validasi dan Interpretasi Hasil

Prinsip kerja yang digunakan adalah memisahkan input, proses, validasi, dan ekspor. Pemisahan ini membuat kesalahan lebih mudah dilacak dan mencegah hasil lama dianggap sebagai hasil dari parameter baru. Setiap langkah perlu meninggalkan bukti yang dapat dibaca kembali: nilai parameter, versi pustaka, status proses, nama berkas, serta pesan kesalahan bila terjadi kegagalan.

Aspek validasi	Kriteria lulus
Input	Tipe, bentuk, satuan, muatan, spin, dan jalur berkas dinyatakan eksplisit.
Lingkungan	Versi Python dan pustaka tercatat; backend CPU/GPU terdeteksi.

Proses	Status eksekusi atau konvergensi diperiksa sebelum hasil dipakai.
Numerik	Tidak ada NaN/inf; ukuran array dan jumlah atom konsisten.
Keluaran	Informasi lingkungan dan berkas hasil disimpan dan dapat dibuka kembali.
Interpretasi	Klaim tidak melampaui metode, basis set, atau data yang dihitung.

Interpretasi aman: Sebutkan metode, parameter, satuan, status konvergensi, dan keterbatasan saat menuliskan kesimpulan.

Kesalahan Umum dan Troubleshooting

Risiko utama pada topik ini adalah runtime terputus, sumber daya berubah, atau berkas sementara hilang. Karena itu, notebook harus memuat pemeriksaan eksplisit dan bukan hanya mengandalkan keluaran yang tampak masuk akal. Pembaca dianjurkan menjalankan notebook dari keadaan runtime bersih, membandingkan hasil dengan batas kewajaran, dan menyimpan artefak penting di lokasi persisten.

Gejala	Tindakan perbaikan
Sel dijalankan tidak berurutan	Restart session, lalu gunakan Run all dan periksa variabel prasyarat.
Pustaka tidak ditemukan	Jalankan kembali sel %pip, periksa nama paket, lalu restart bila dependensi inti berubah.
Input tampak benar tetapi gagal	Cetak tipe, shape, satuan, muatan, spin, dan beberapa baris pertama data.
Runtime terputus, sumber daya berubah, atau berkas sementara hilang	Tambahkan validasi sebelum proses utama dan hentikan eksekusi dengan pesan yang spesifik.
Hasil berbeda antar-runtime	Catat versi runtime/paket, random seed, parameter, dan perangkat yang digunakan.

Aturan troubleshooting: Perbaiki satu hal pada satu waktu, simpan pesan kesalahan lengkap, dan catat perubahan versi atau parameter.

Latihan Modifikasi dan Refleksi

232. Ubah satu parameter pada contoh Google Drive dan penyimpanan hasil, jalankan ulang dari runtime bersih, dan jelaskan perubahan keluaran.
233. Tambahkan satu validasi baru yang mencegah runtime terputus, sumber daya berubah, atau berkas sementara hilang.
234. Buat berkas ringkasan TXT atau JSON yang memuat input, status proses, waktu, dan hasil utama.
235. Jelaskan bagian mana yang merupakan fakta keluaran program dan bagian mana yang merupakan interpretasi ilmiah.
236. Susun satu kasus uji yang sengaja salah, lalu pastikan notebook menghasilkan pesan kesalahan yang informatif.

Refleksi: Tuliskan apa yang berubah setelah kode dimodifikasi, bukti apa yang menunjukkan keberhasilan, dan bagian mana yang masih memerlukan verifikasi eksternal.

Rubrik	Deskripsi
Skor 4	Kode berjalan dari runtime bersih, validasi lengkap, interpretasi tepat.
Skor 3	Kode berjalan dan hasil tersimpan, tetapi dokumentasi belum lengkap.
Skor 2	Kode hanya berjalan setelah intervensi atau state tertentu.
Skor 1	Kode belum berjalan dan diagnosis kesalahan belum sistematis.

2.3.3 Membuat laporan dan arsip ZIP

Pembahasan Membuat laporan dan arsip ZIP ditempatkan dalam konteks praktik Google Colab agar pembaca dapat menghubungkan konsep, prosedur, kode, dan interpretasi secara utuh. Menggabungkan hasil, log, grafik, struktur molekul, dan ringkasan ke dalam satu arsip. Hasil yang dituju berupa konformer tiga dimensi dan koordinat XYZ.

Prinsip kerja yang digunakan adalah memisahkan input, proses, validasi, dan ekspor. Pemisahan ini membuat kesalahan lebih mudah dilacak dan mencegah hasil lama dianggap sebagai hasil dari parameter baru. Setiap langkah perlu meninggalkan bukti

yang dapat dibaca kembali: nilai parameter, versi pustaka, status proses, nama berkas, serta pesan kesalahan bila terjadi kegagalan.

Tujuan operasional: Pembaca mampu menjelaskan Membuat laporan dan arsip ZIP, menjalankan contoh minimal, memeriksa keberhasilan, dan menyimpan bukti hasil.

Komponen	Kriteria
Konsep	Menggabungkan hasil, log, grafik, struktur molekul, dan ringkasan ke dalam satu arsip.
Input	Parameter dan data dinyatakan eksplisit.
Proses	Langkah dapat diulang dari runtime bersih.
Output	Nilai, berkas, atau visualisasi yang dapat diperiksa.
Batas	Interpretasi mengikuti metode dan konteks.

Landasan Teori dan Batas Penerapan

Membuat laporan dan arsip ZIP merupakan simpul penting dalam alur kerja notebook ilmiah karena menentukan bagaimana struktur molekul disiapkan, dijalankan, dan diperiksa. Menggabungkan hasil, log, grafik, struktur molekul, dan ringkasan ke dalam satu arsip. Hasil yang dituju berupa konformer tiga dimensi dan koordinat XYZ.

Prinsip kerja yang digunakan adalah memisahkan input, proses, validasi, dan ekspor. Pemisahan ini membuat kesalahan lebih mudah dilacak dan mencegah hasil lama dianggap sebagai hasil dari parameter baru. Setiap langkah perlu meninggalkan bukti yang dapat dibaca kembali: nilai parameter, versi pustaka, status proses, nama berkas, serta pesan kesalahan bila terjadi kegagalan.

Risiko utama pada topik ini adalah struktur 2D, stereokimia, muatan, dan koordinat tidak valid. Karena itu, notebook harus memuat pemeriksaan eksplisit dan bukan hanya mengandalkan keluaran yang tampak masuk akal. Pembaca dianjurkan menjalankan notebook dari keadaan runtime bersih, membandingkan hasil dengan batas kewajaran, dan menyimpan artefak penting di lokasi persisten.

Batas penerapan: Contoh pada buku berfungsi sebagai pola kerja. Ukuran molekul, biaya komputasi, versi pustaka, dan perangkat keras dapat mengubah waktu serta strategi yang diperlukan.

Prosedur Bertahap

Pembahasan Membuat laporan dan arsip ZIP ditempatkan dalam konteks praktik Google Colab agar pembaca dapat menghubungkan konsep, prosedur, kode, dan interpretasi secara utuh. Menggabungkan hasil, log, grafik, struktur molekul, dan ringkasan ke dalam satu arsip. Hasil yang dituju berupa konformer tiga dimensi dan koordinat XYZ.

237. Tuliskan tujuan operasional Membuat laporan dan arsip ZIP dan tentukan keluaran yang harus tersedia pada akhir sel.
238. Tetapkan parameter pada satu sel konfigurasi agar perubahan dapat ditelusuri dan tidak tersebar di banyak tempat.
239. Siapkan struktur molekul dengan input yang eksplisit; hindari nilai tersembunyi yang hanya tersimpan dalam state kernel.
240. Jalankan pemeriksaan prasyarat, termasuk keberadaan pustaka, bentuk data, tipe nilai, serta konsistensi satuan.
241. Eksekusi proses utama, tangkap status konvergensi atau keberhasilan, dan hentikan alur apabila syarat minimum tidak terpenuhi.
242. Simpan konformer tiga dimensi dan koordinat XYZ dengan nama berkas informatif, kemudian tulis interpretasi yang membedakan fakta komputasi dan kesimpulan ilmiah.

Titik kontrol: Jangan melanjutkan ke interpretasi apabila validasi input atau status proses belum memenuhi kriteria.

Kode Dasar Siap Colab

Membuat laporan dan arsip ZIP merupakan simpul penting dalam alur kerja notebook ilmiah karena menentukan bagaimana struktur molekul disiapkan, dijalankan, dan diperiksa. Menggabungkan hasil, log, grafik, struktur molekul, dan ringkasan ke dalam satu arsip. Hasil yang dituju berupa konformer tiga dimensi dan koordinat XYZ.

Sel kode contoh

```
from pathlib import Path
import shutil

folder = Path("hasil_proyek")
folder.mkdir(exist_ok=True)
(folder/"ringkasan.txt").write_text("Status: selesai", encoding="utf-8")
(folder/"parameter.json").write_text('{"method": "B3LYP"}', encoding="utf-8")
shutil.make_archive("hasil_proyek", "zip", root_dir=folder)
print("Arsip:", Path("hasil_proyek.zip").resolve())
```

Cara menjalankan: Tempatkan kode dalam satu atau beberapa sel sesuai komentar. Untuk baris %pip atau perintah shell, gunakan runtime Google Colab.

Penjelasan Baris Kode Penting

Potongan kode	Fungsi
<code>from pathlib import Path</code>	Mencetak status agar pengguna tidak hanya bergantung pada keluaran visual.
<code>import shutil</code>	Mencetak status agar pengguna tidak hanya bergantung pada keluaran visual.
<code>folder = Path("hasil_proyek")</code>	Menambahkan pemeriksaan untuk mencegah hasil tidak valid diteruskan.
<code>folder.mkdir(exist_ok=True)</code>	Mencetak status agar pengguna tidak hanya bergantung pada keluaran visual.
<code>(folder/"ringkasan.txt").write_text("Status: sel</code>	Menambahkan pemeriksaan untuk mencegah hasil tidak valid diteruskan.
<code>(folder/"parameter.json").write_text '{"method":</code>	Mendefinisikan input atau parameter yang akan digunakan pada langkah berikutnya.
<code>shutil.make_archive("hasil_proyek", "zip", root_</code>	Mendefinisikan input atau parameter yang akan digunakan pada langkah berikutnya.
<code>print("Arsip:", Path("hasil_proyek.zip").resolve</code>	Mendefinisikan input atau parameter yang akan digunakan pada langkah berikutnya.

Nama variabel dipilih agar maknanya terbaca. Hindari singkatan yang tidak dijelaskan, nilai konstan tanpa satuan, serta penggunaan variabel dari sel lama. Komentar harus menerangkan alasan atau asumsi, bukan sekadar mengulang sintaks.

Validasi dan Interpretasi Hasil

Prinsip kerja yang digunakan adalah memisahkan input, proses, validasi, dan ekspor. Pemisahan ini membuat kesalahan lebih mudah dilacak dan mencegah hasil lama

dianggap sebagai hasil dari parameter baru. Setiap langkah perlu meninggalkan bukti yang dapat dibaca kembali: nilai parameter, versi pustaka, status proses, nama berkas, serta pesan kesalahan bila terjadi kegagalan.

Aspek validasi	Kriteria lulus
Input	Tipe, bentuk, satuan, muatan, spin, dan jalur berkas dinyatakan eksplisit.
Lingkungan	Versi Python dan pustaka tercatat; backend CPU/GPU terdeteksi.
Proses	Status eksekusi atau konvergensi diperiksa sebelum hasil dipakai.
Numerik	Tidak ada NaN/inf; ukuran array dan jumlah atom konsisten.
Keluaran	Konformer tiga dimensi dan koordinat xyz disimpan dan dapat dibuka kembali.
Interpretasi	Klaim tidak melampaui metode, basis set, atau data yang dihitung.
Interpretasi aman: Sebutkan metode, parameter, satuan, status konvergensi, dan keterbatasan saat menuliskan kesimpulan.	

Kesalahan Umum dan Troubleshooting

Risiko utama pada topik ini adalah struktur 2D, stereokimia, muatan, dan koordinat tidak valid. Karena itu, notebook harus memuat pemeriksaan eksplisit dan bukan hanya mengandalkan keluaran yang tampak masuk akal. Pembaca dianjurkan menjalankan notebook dari keadaan runtime bersih, membandingkan hasil dengan batas kewajaran, dan menyimpan artefak penting di lokasi persisten.

Gejala	Tindakan perbaikan
Sel dijalankan tidak berurutan	Restart session, lalu gunakan Run all dan periksa variabel prasyarat.
Pustaka tidak ditemukan	Jalankan kembali sel %pip, periksa nama paket, lalu restart bila dependensi inti berubah.

Input tampak benar tetapi gagal	Cetak tipe, shape, satuan, muatan, spin, dan beberapa baris pertama data.
Struktur 2d, stereokimia, muatan, dan koordinat tidak valid	Tambahkan validasi sebelum proses utama dan hentikan eksekusi dengan pesan yang spesifik.
Hasil berbeda antar-runtime	Catat versi runtime/paket, random seed, parameter, dan perangkat yang digunakan.
Aturan troubleshooting: Perbaiki satu hal pada satu waktu, simpan pesan kesalahan lengkap, dan catat perubahan versi atau parameter.	

Latihan Modifikasi dan Refleksi

243. Ubah satu parameter pada contoh Membuat laporan dan arsip ZIP, jalankan ulang dari runtime bersih, dan jelaskan perubahan keluaran.
244. Tambahkan satu validasi baru yang mencegah struktur 2D, stereokimia, muatan, dan koordinat tidak valid.
245. Buat berkas ringkasan TXT atau JSON yang memuat input, status proses, waktu, dan hasil utama.
246. Jelaskan bagian mana yang merupakan fakta keluaran program dan bagian mana yang merupakan interpretasi ilmiah.
247. Susun satu kasus uji yang sengaja salah, lalu pastikan notebook menghasilkan pesan kesalahan yang informatif.

Refleksi: Tuliskan apa yang berubah setelah kode dimodifikasi, bukti apa yang menunjukkan keberhasilan, dan bagian mana yang masih memerlukan verifikasi eksternal.

Rubrik	Deskripsi
Skor 4	Kode berjalan dari runtime bersih, validasi lengkap, interpretasi tepat.
Skor 3	Kode berjalan dan hasil tersimpan, tetapi dokumentasi belum lengkap.
Skor 2	Kode hanya berjalan setelah intervensi atau state tertentu.
Skor 1	Kode belum berjalan dan diagnosis kesalahan belum sistematis.

Rangkuman Bab

Bab ini menempatkan reproduksibilitas sebagai prinsip utama. Struktur notebook, instalasi paket, pemeriksaan lingkungan, pencatatan versi, penyimpanan persisten, dan arsip hasil dibahas sebagai satu alur.

- Judul, identitas, tujuan, dan teori singkat
- Konfigurasi parameter dan fungsi modular
- Urutan eksekusi dan kebersihan notebook
- Instalasi dengan %pip
- Pemeriksaan Python, sistem, RAM, dan versi pustaka
- Deteksi CPU, GPU, CUDA, dan kompatibilitas
- Membaca dan menulis TXT, CSV, JSON, dan XYZ
- Google Drive dan penyimpanan hasil
- Membuat laporan dan arsip ZIP

Inti bab: Keberhasilan teknis harus selalu disertai validasi, dokumentasi, dan interpretasi yang proporsional.

Proyek Praktik Terpadu

Proyek terpadu menggabungkan seluruh keterampilan bab dalam satu notebook. Gunakan satu sel konfigurasi, fungsi modular, validasi, logging sederhana, penyimpanan hasil, dan bagian pembahasan.

248. Tentukan pertanyaan atau tujuan komputasi.
249. Siapkan input dan parameter.
250. Jalankan proses utama.
251. Validasi status dan nilai.
252. Buat satu visualisasi atau tabel.
253. Ekspor hasil dan tulis pembahasan.
254. Restart session dan Run all.

Produk akhir: Notebook .ipynb, satu folder hasil, arsip ZIP, dan ringkasan pengujian.

Pertanyaan Analitis

255. Mengapa kode yang menghasilkan angka belum tentu menghasilkan bukti ilmiah?
256. Bagaimana versi runtime dapat memengaruhi reproduksibilitas?
257. Apa perbedaan pemeriksaan sintaks, pemeriksaan numerik, dan validasi kimia?
258. Kapan fallback dapat diterima, dan bagaimana harus dilaporkan?
259. Apa batas interpretasi dari metode atau contoh pada bab ini?
260. Bagaimana merancang notebook agar mudah direview oleh orang lain?

Panduan Jawaban dan Rubrik

Komponen	Bobot dan indikator
Ketepatan konsep	25% - definisi, asumsi, dan satuan benar.
Kualitas kode	25% - modular, terbaca, dan menangani kesalahan.
Validasi	25% - input, proses, dan keluaran diperiksa.
Interpretasi	15% - klaim sesuai bukti dan keterbatasan.
Reproduksibilitas	10% - dapat dijalankan dari runtime bersih.

Panduan jawaban menilai alur penalaran, bukan hanya keluaran akhir. Perbedaan nilai kecil dapat diterima bila versi atau perangkat berbeda dan semua parameter dilaporkan.

Daftar Periksa Penguasaan Bab

Pernyataan	Cek
Saya dapat menjelaskan konsep utama	☐
Saya dapat menjalankan kode dari runtime bersih	☐
Saya memeriksa status dan nilai numerik	☐
Saya menyimpan hasil di lokasi persisten	☐
Saya dapat menjelaskan kesalahan umum	☐
Saya menuliskan batas interpretasi	☐
Saya mencatat versi dan waktu pengujian	☐
Tindak lanjut: Ulangi halaman yang belum dikuasai, lalu kerjakan proyek bab tanpa melihat kode contoh secara langsung.	

3

Membangun dan Menampilkan Struktur Molekul

**BAB 3 - MEMBANGUN DAN MENAMPILKAN
STRUKTUR MOLEKUL TIGA DIMENSI**

Bab ini mengubah identitas kimia menjadi struktur molekul tiga dimensi. Pembaca mempelajari SMILES, muatan, spin, ETKDGv3, MMFF94, UFF, ekspor XYZ, dan visualisasi py3Dmol.

Arah Bab: Setiap konsep diikuti prosedur, kode, penjelasan, validasi, troubleshooting, latihan, dan refleksi. Keluaran numerik tidak boleh dipisahkan dari konteks metode dan batas penerapannya.

Komponen	Kriteria
Orientasi	Memahami konsep dan istilah.
Praktik	Menjalankan kode secara bertahap.
Validasi	Memastikan input, proses, dan keluaran sah.
Interpretasi	Menjelaskan hasil tanpa klaim berlebihan.

Capaian Pembelajaran dan Prasyarat

- Menjelaskan konsep utama bab dengan istilah yang tepat.
- Menjalankan notebook dari runtime bersih dan mendokumentasikan versi.
- Memodifikasi parameter tanpa merusak alur validasi.
- Menyimpan hasil dalam format yang dapat dibuka kembali.
- Membedakan keluaran program, interpretasi, dan keterbatasan.

Prasyarat: Kemampuan menggunakan browser, akun Google, operasi berkas dasar, dan kemauan membaca pesan kesalahan secara teliti.

Komponen	Kriteria
Pengetahuan	Konsep sebelumnya dan satuan dasar.
Keterampilan	Menyalin kode, menjalankan sel, dan membaca output.
Sikap ilmiah	Jujur terhadap status pengujian dan ketidakpastian.
Perangkat	Browser modern dan koneksi internet stabil.

Peta Konsep dan Alur Kerja

Subbab	Alur konsep
3.1 Representasi dan Validasi Molekul	Representasi SMILES, SDF, MOL, dan XYZ → Muatan, jumlah elektron, multiplicity, dan spin → Sumber struktur molekul
3.2 Pembuatan Konformer dengan RDKit	Mengubah SMILES menjadi molekul tiga dimensi → ETKDGv3, MMFF94, dan UFF → Validasi dan ekspor koordinat
3.3 Visualisasi Molekul dengan py3Dmol	Menampilkan struktur awal → Memberikan label dan informasi geometri → Membandingkan struktur awal dan akhir

Peta konsep dibaca dari kiri ke kanan dan dari input menuju keluaran. Hubungan antarkomponen tidak selalu linear: validasi dapat mengembalikan pembaca ke tahap input, sedangkan troubleshooting dapat memerlukan instalasi ulang atau penggantian metode.

Prinsip alur: Notebook yang baik memperlihatkan jalur keputusan: apa yang diperiksa, kapan proses dihentikan, kapan fallback digunakan, dan berkas apa yang disimpan.

Istilah Kunci dan Konvensi

Istilah	Konvensi penggunaan

notebook	Istilah digunakan secara konsisten dan dijelaskan saat pertama muncul.
runtime	Istilah digunakan secara konsisten dan dijelaskan saat pertama muncul.
kernel	Nilai atau statusnya dicatat dalam notebook dan laporan.
sel Markdown	Istilah digunakan secara konsisten dan dijelaskan saat pertama muncul.
sel kode	Nilai atau statusnya dicatat dalam notebook dan laporan.
parameter	Maknanya dibedakan dari istilah sehari-hari yang mirip.
validasi	Nilai atau statusnya dicatat dalam notebook dan laporan.
konvergensi	Maknanya dibedakan dari istilah sehari-hari yang mirip.

Konvensi satuan: Panjang molekul dinyatakan dalam Angstrom, energi elektronik dalam Hartree, energi orbital dapat dikonversi ke eV, dan perubahan energi dapat dilaporkan dalam kJ mol⁻¹.

Asesmen Diagnostik Awal

261. Apa perbedaan notebook, runtime, dan berkas .ipynb?
262. Mengapa menjalankan sel tidak berurutan dapat menghasilkan keluaran yang menyesatkan?
263. Kapan sebuah nilai numerik boleh ditafsirkan sebagai hasil yang sah?
264. Informasi lingkungan apa yang perlu dicatat agar pekerjaan dapat diulang?
265. Bagaimana membedakan kegagalan instalasi, kegagalan input, dan kegagalan konvergensi?
266. Berkas apa yang harus disimpan sebelum runtime Colab terputus?

Cara menggunakan asesmen: Jawab sebelum memulai bab, tandai tingkat keyakinan, lalu ulangi setelah menyelesaikan proyek bab. Perubahan alasan lebih penting daripada perubahan skor saja.

3.1 Representasi dan Validasi Molekul

Representasi dan Validasi Molekul mengorganisasi tiga komponen yang saling bergantung. Pembahasan bergerak dari konsep, menuju prosedur, kemudian menghasilkan artefak yang dapat divalidasi dan disimpan.

Sub-subbab	Fokus
3.1.1 Representasi SMILES, SDF, MOL, dan XYZ	Fungsi setiap format, informasi yang disimpan, serta keterbatasan koordinat dua dimensi.
3.1.2 Muatan, jumlah elektron, multiplicity, dan spin	Perbedaan sistem closed-shell dan open-shell serta pemilihan RKS, UKS, RHF, atau UHF.
3.1.3 Sumber struktur molekul	Struktur eksperimental, basis data kimia, koordinat literatur, dan konformer komputasional.
Produk belajar: Satu notebook terstruktur, satu berkas keluaran, satu catatan validasi, dan satu interpretasi singkat yang menyatakan batas hasil.	

Rencana Praktik dan Produk Belajar

267. Pelajari Representasi SMILES, SDF, MOL, dan XYZ dan jalankan kode dasar yang menyertainya.
268. Pelajari Muatan, jumlah elektron, multiplicity, dan spin dan jalankan kode dasar yang menyertainya.
269. Pelajari Sumber struktur molekul dan jalankan kode dasar yang menyertainya.

Komponen	Kriteria
Waktu	60-120 menit, bergantung pada komputasi.
Input	Notebook baru atau salinan notebook bab.
Produk	Kode, output, berkas hasil, dan refleksi.
Uji	Restart session lalu Run all.
Dokumentasi	Versi, parameter, status, dan waktu eksekusi.

Kriteria selesai: Semua sel dapat dijalankan berurutan, pesan kesalahan ditangani, dan hasil utama tersimpan di lokasi persisten.

3.1.1 Representasi SMILES, SDF, MOL, dan XYZ

Pembahasan Representasi SMILES, SDF, MOL, dan XYZ ditempatkan dalam konteks praktik Google Colab agar pembaca dapat menghubungkan konsep, prosedur, kode, dan interpretasi secara utuh. Fungsi setiap format, informasi yang disimpan, serta keterbatasan koordinat dua dimensi. Hasil yang dituju berupa konformer tiga dimensi dan koordinat XYZ.

Prinsip kerja yang digunakan adalah memisahkan input, proses, validasi, dan ekspor. Pemisahan ini membuat kesalahan lebih mudah dilacak dan mencegah hasil lama dianggap sebagai hasil dari parameter baru. Setiap langkah perlu meninggalkan bukti yang dapat dibaca kembali: nilai parameter, versi pustaka, status proses, nama berkas, serta pesan kesalahan bila terjadi kegagalan.

Tujuan operasional: Pembaca mampu menjelaskan Representasi SMILES, SDF, MOL, dan XYZ, menjalankan contoh minimal, memeriksa keberhasilan, dan menyimpan bukti hasil.

Komponen	Kriteria
Konsep	Fungsi setiap format, informasi yang disimpan, serta keterbatasan koordinat dua dimensi.
Input	Parameter dan data dinyatakan eksplisit.
Proses	Langkah dapat diulang dari runtime bersih.
Output	Nilai, berkas, atau visualisasi yang dapat diperiksa.
Batas	Interpretasi mengikuti metode dan konteks.

Landasan Teori dan Batas Penerapan

Representasi SMILES, SDF, MOL, dan XYZ merupakan simpul penting dalam alur kerja notebook ilmiah karena menentukan bagaimana struktur molekul disiapkan, dijalankan, dan diperiksa. Fungsi setiap format, informasi yang disimpan, serta keterbatasan koordinat dua dimensi. Hasil yang dituju berupa konformer tiga dimensi dan koordinat XYZ.

Prinsip kerja yang digunakan adalah memisahkan input, proses, validasi, dan ekspor. Pemisahan ini membuat kesalahan lebih mudah dilacak dan mencegah hasil lama dianggap sebagai hasil dari parameter baru. Setiap langkah perlu meninggalkan bukti yang dapat dibaca kembali: nilai parameter, versi pustaka, status proses, nama berkas, serta pesan kesalahan bila terjadi kegagalan.

Risiko utama pada topik ini adalah struktur 2D, stereokimia, muatan, dan koordinat tidak valid. Karena itu, notebook harus memuat pemeriksaan eksplisit dan bukan hanya mengandalkan keluaran yang tampak masuk akal. Pembaca dianjurkan menjalankan notebook dari keadaan runtime bersih, membandingkan hasil dengan batas kewajaran, dan menyimpan artefak penting di lokasi persisten.

Batas penerapan: Contoh pada buku berfungsi sebagai pola kerja. Ukuran molekul, biaya komputasi, versi pustaka, dan perangkat keras dapat mengubah waktu serta strategi yang diperlukan.

Prosedur Bertahap

Pada bagian Representasi SMILES, SDF, MOL, dan XYZ, perhatian tidak hanya diarahkan pada keberhasilan menjalankan kode, tetapi juga pada ketepatan asumsi, keterlacakan parameter, dan mutu keluaran. Fungsi setiap format, informasi yang disimpan, serta keterbatasan koordinat dua dimensi. Hasil yang dituju berupa konformer tiga dimensi dan koordinat XYZ.

270. Tuliskan tujuan operasional Representasi SMILES, SDF, MOL, dan XYZ dan tentukan keluaran yang harus tersedia pada akhir sel.
271. Tetapkan parameter pada satu sel konfigurasi agar perubahan dapat ditelusuri dan tidak tersebar di banyak tempat.
272. Siapkan struktur molekul dengan input yang eksplisit; hindari nilai tersembunyi yang hanya tersimpan dalam state kernel.
273. Jalankan pemeriksaan prasyarat, termasuk keberadaan pustaka, bentuk data, tipe nilai, serta konsistensi satuan.
274. Eksekusi proses utama, tangkap status konvergensi atau keberhasilan, dan hentikan alur apabila syarat minimum tidak terpenuhi.
275. Simpan konformer tiga dimensi dan koordinat XYZ dengan nama berkas informatif, kemudian tulis interpretasi yang membedakan fakta komputasi dan kesimpulan ilmiah.

Titik kontrol: Jangan melanjutkan ke interpretasi apabila validasi input atau status proses belum memenuhi kriteria.

Kode Dasar Siap Colab

Representasi SMILES, SDF, MOL, dan XYZ merupakan simpul penting dalam alur kerja notebook ilmiah karena menentukan bagaimana struktur molekul disiapkan, dijalankan,

dan diperiksa. Fungsi setiap format, informasi yang disimpan, serta keterbatasan koordinat dua dimensi. Hasil yang dituju berupa konformer tiga dimensi dan koordinat XYZ.

Sel kode contoh

```
from rdkit import Chem

smiles = "CCO"
mol = Chem.MolFromSmiles(smiles)
if mol is None:
    raise ValueError("SMILES tidak valid")
print("Atom berat:", mol.GetNumHeavyAtoms())
print("SMILES kanonik:", Chem.MolToSmiles(mol))

# XYZ memerlukan koordinat 3D; SMILES sendiri tidak menyimpannya.
```

Cara menjalankan: Tempatkan kode dalam satu atau beberapa sel sesuai komentar. Untuk baris %pip atau perintah shell, gunakan runtime Google Colab.

Penjelasan Baris Kode Penting

Potongan kode	Fungsi
from rdkit import Chem	Mendefinisikan input atau parameter yang akan digunakan pada langkah berikutnya.
smiles = "CCO"	Menambahkan pemeriksaan untuk mencegah hasil tidak valid diteruskan.
mol = Chem.MolFromSmiles(smiles)	Menambahkan pemeriksaan untuk mencegah hasil tidak valid diteruskan.
if mol is None:	Menambahkan pemeriksaan untuk mencegah hasil tidak valid diteruskan.
raise ValueError("SMILES tidak valid")	Mencetak status agar pengguna tidak hanya bergantung pada keluaran visual.
print("Atom berat:", mol.GetNumHeavyAtoms())	Mencetak status agar pengguna tidak hanya bergantung pada keluaran visual.

<code>print("SMILES kanonik:", Chem.MolToSmiles(mol))</code>	Menjalankan operasi utama dan menyimpan hasil pada variabel yang jelas.
<code># XYZ memerlukan koordinat 3D; SMILES sendiri ti</code>	Mencetak status agar pengguna tidak hanya bergantung pada keluaran visual.

Nama variabel dipilih agar maknanya terbaca. Hindari singkatan yang tidak dijelaskan, nilai konstan tanpa satuan, serta penggunaan variabel dari sel lama. Komentar harus menerangkan alasan atau asumsi, bukan sekadar mengulang sintaks.

Validasi dan Interpretasi Hasil

Prinsip kerja yang digunakan adalah memisahkan input, proses, validasi, dan ekspor. Pemisahan ini membuat kesalahan lebih mudah dilacak dan mencegah hasil lama dianggap sebagai hasil dari parameter baru. Setiap langkah perlu meninggalkan bukti yang dapat dibaca kembali: nilai parameter, versi pustaka, status proses, nama berkas, serta pesan kesalahan bila terjadi kegagalan.

Aspek validasi	Kriteria lulus
Input	Tipe, bentuk, satuan, muatan, spin, dan jalur berkas dinyatakan eksplisit.
Lingkungan	Versi Python dan pustaka tercatat; backend CPU/GPU terdeteksi.
Proses	Status eksekusi atau konvergensi diperiksa sebelum hasil dipakai.
Numerik	Tidak ada NaN/inf; ukuran array dan jumlah atom konsisten.
Keluaran	Konformer tiga dimensi dan koordinat xyz disimpan dan dapat dibuka kembali.
Interpretasi	Klaim tidak melampaui metode, basis set, atau data yang dihitung.
Interpretasi aman: Sebutkan metode, parameter, satuan, status konvergensi, dan keterbatasan saat menuliskan kesimpulan.	

Kesalahan Umum dan Troubleshooting

Risiko utama pada topik ini adalah struktur 2D, stereokimia, muatan, dan koordinat tidak valid. Karena itu, notebook harus memuat pemeriksaan eksplisit dan bukan hanya mengandalkan keluaran yang tampak masuk akal. Pembaca dianjurkan menjalankan notebook dari keadaan runtime bersih, membandingkan hasil dengan batas kewajaran, dan menyimpan artefak penting di lokasi persisten.

Gejala	Tindakan perbaikan
Sel dijalankan tidak berurutan	Restart session, lalu gunakan Run all dan periksa variabel prasyarat.
Pustaka tidak ditemukan	Jalankan kembali sel %pip, periksa nama paket, lalu restart bila dependensi inti berubah.
Input tampak benar tetapi gagal	Cetak tipe, shape, satuan, muatan, spin, dan beberapa baris pertama data.
Struktur 2d, stereokimia, muatan, dan koordinat tidak valid	Tambahkan validasi sebelum proses utama dan hentikan eksekusi dengan pesan yang spesifik.
Hasil berbeda antar-runtime	Catat versi runtime/paket, random seed, parameter, dan perangkat yang digunakan.
Aturan troubleshooting: Perbaiki satu hal pada satu waktu, simpan pesan kesalahan lengkap, dan catat perubahan versi atau parameter.	

Latihan Modifikasi dan Refleksi

276. Ubah satu parameter pada contoh Representasi SMILES, SDF, MOL, dan XYZ, jalankan ulang dari runtime bersih, dan jelaskan perubahan keluaran.
277. Tambahkan satu validasi baru yang mencegah struktur 2D, stereokimia, muatan, dan koordinat tidak valid.
278. Buat berkas ringkasan TXT atau JSON yang memuat input, status proses, waktu, dan hasil utama.
279. Jelaskan bagian mana yang merupakan fakta keluaran program dan bagian mana yang merupakan interpretasi ilmiah.
280. Susun satu kasus uji yang sengaja salah, lalu pastikan notebook menghasilkan pesan kesalahan yang informatif.

Refleksi: Tuliskan apa yang berubah setelah kode dimodifikasi, bukti apa yang menunjukkan keberhasilan, dan bagian mana yang masih memerlukan verifikasi eksternal.

Rubrik	Deskripsi
Skor 4	Kode berjalan dari runtime bersih, validasi lengkap, interpretasi tepat.
Skor 3	Kode berjalan dan hasil tersimpan, tetapi dokumentasi belum lengkap.
Skor 2	Kode hanya berjalan setelah intervensi atau state tertentu.
Skor 1	Kode belum berjalan dan diagnosis kesalahan belum sistematis.

3.1.2 Muatan, jumlah elektron, multiplicity, dan spin

Pada bagian Muatan, jumlah elektron, multiplicity, dan spin, perhatian tidak hanya diarahkan pada keberhasilan menjalankan kode, tetapi juga pada ketepatan asumsi, keterlacakan parameter, dan mutu keluaran. Perbedaan sistem closed-shell dan open-shell serta pemilihan RKS, UKS, RHF, atau UHF. Hasil yang dituju berupa hasil yang dapat diperiksa dan disimpan.

Prinsip kerja yang digunakan adalah memisahkan input, proses, validasi, dan ekspor. Pemisahan ini membuat kesalahan lebih mudah dilacak dan mencegah hasil lama dianggap sebagai hasil dari parameter baru. Setiap langkah perlu meninggalkan bukti yang dapat dibaca kembali: nilai parameter, versi pustaka, status proses, nama berkas, serta pesan kesalahan bila terjadi kegagalan.

Tujuan operasional: Pembaca mampu menjelaskan Muatan, jumlah elektron, multiplicity, dan spin, menjalankan contoh minimal, memeriksa keberhasilan, dan menyimpan bukti hasil.

Komponen	Kriteria
Konsep	Perbedaan sistem closed-shell dan open-shell serta pemilihan RKS, UKS, RHF, atau UHF.
Input	Parameter dan data dinyatakan eksplisit.

Proses	Langkah dapat diulang dari runtime bersih.
Output	Nilai, berkas, atau visualisasi yang dapat diperiksa.
Batas	Interpretasi mengikuti metode dan konteks.

Landasan Teori dan Batas Penerapan

Muatan, jumlah elektron, multiplicity, dan spin merupakan simpul penting dalam alur kerja notebook ilmiah karena menentukan bagaimana prosedur komputasi disiapkan, dijalankan, dan diperiksa. Perbedaan sistem closed-shell dan open-shell serta pemilihan RKS, UKS, RHF, atau UHF. Hasil yang dituju berupa hasil yang dapat diperiksa dan disimpan.

Prinsip kerja yang digunakan adalah memisahkan input, proses, validasi, dan ekspor. Pemisahan ini membuat kesalahan lebih mudah dilacak dan mencegah hasil lama dianggap sebagai hasil dari parameter baru. Setiap langkah perlu meninggalkan bukti yang dapat dibaca kembali: nilai parameter, versi pustaka, status proses, nama berkas, serta pesan kesalahan bila terjadi kegagalan.

Risiko utama pada topik ini adalah parameter tidak terdokumentasi dan keluaran tidak tervalidasi. Karena itu, notebook harus memuat pemeriksaan eksplisit dan bukan hanya mengandalkan keluaran yang tampak masuk akal. Pembaca dianjurkan menjalankan notebook dari keadaan runtime bersih, membandingkan hasil dengan batas kewajaran, dan menyimpan artefak penting di lokasi persisten.

Batas penerapan: Contoh pada buku berfungsi sebagai pola kerja. Ukuran molekul, biaya komputasi, versi pustaka, dan perangkat keras dapat mengubah waktu serta strategi yang diperlukan.

Prosedur Bertahap

Pada bagian Muatan, jumlah elektron, multiplicity, dan spin, perhatian tidak hanya diarahkan pada keberhasilan menjalankan kode, tetapi juga pada ketepatan asumsi, keterlacakan parameter, dan mutu keluaran. Perbedaan sistem closed-shell dan open-shell serta pemilihan RKS, UKS, RHF, atau UHF. Hasil yang dituju berupa hasil yang dapat diperiksa dan disimpan.

281. Tuliskan tujuan operasional Muatan, jumlah elektron, multiplicity, dan spin dan tentukan keluaran yang harus tersedia pada akhir sel.
282. Tetapkan parameter pada satu sel konfigurasi agar perubahan dapat ditelusuri dan tidak tersebar di banyak tempat.

283. Siapkan prosedur komputasi dengan input yang eksplisit; hindari nilai tersembunyi yang hanya tersimpan dalam state kernel.
284. Jalankan pemeriksaan prasyarat, termasuk keberadaan pustaka, bentuk data, tipe nilai, serta konsistensi satuan.
285. Eksekusi proses utama, tangkap status konvergensi atau keberhasilan, dan hentikan alur apabila syarat minimum tidak terpenuhi.
286. Simpan hasil yang dapat diperiksa dan disimpan dengan nama berkas informatif, kemudian tulis interpretasi yang membedakan fakta komputasi dan kesimpulan ilmiah.

Titik kontrol: Jangan melanjutkan ke interpretasi apabila validasi input atau status proses belum memenuhi kriteria.

Kode Dasar Siap Colab

Pembahasan Muatan, jumlah elektron, multiplicity, dan spin ditempatkan dalam konteks praktik Google Colab agar pembaca dapat menghubungkan konsep, prosedur, kode, dan interpretasi secara utuh. Perbedaan sistem closed-shell dan open-shell serta pemilihan RKS, UKS, RHF, atau UHF. Hasil yang dituju berupa hasil yang dapat diperiksa dan disimpan.

Sel kode contoh

```
from rdkit import Chem

Z = {a.GetIdx(): a.GetAtomicNum() for a in Chem.MolFromSmiles("O").GetAtoms()}
jumlah_elektron = sum(Z.values()) - 0
multiplicity = 1
spin_pyscf = multiplicity - 1
if (jumlah_elektron - spin_pyscf) % 2:
    raise ValueError("Kombinasi elektron dan spin tidak konsisten")
print(jumlah_elektron, spin_pyscf)
```

Cara menjalankan: Tempatkan kode dalam satu atau beberapa sel sesuai komentar. Untuk baris %pip atau perintah shell, gunakan runtime Google Colab.

Penjelasan Baris Kode Penting

Potongan kode	Fungsi
from rdkit import Chem	Menambahkan pemeriksaan untuk mencegah hasil tidak valid diteruskan.
Z = {a.GetIdx(): a.GetAtomicNum() for a in Chem.	Mencetak status agar pengguna tidak hanya bergantung pada keluaran visual.

<code>jumlah_elektron = sum(Z.values()) - 0</code>	Mendefinisikan input atau parameter yang akan digunakan pada langkah berikutnya.
<code>multiplicity = 1</code>	Mencetak status agar pengguna tidak hanya bergantung pada keluaran visual.
<code>spin_pyscf = multiplicity - 1</code>	Mencetak status agar pengguna tidak hanya bergantung pada keluaran visual.
<code>if (jumlah_elektron - spin_pyscf) % 2:</code>	Mencetak status agar pengguna tidak hanya bergantung pada keluaran visual.
<code>raise ValueError("Kombinasi elektron dan spi</code>	Mencetak status agar pengguna tidak hanya bergantung pada keluaran visual.
<code>print(jumlah_elektron, spin_pyscf)</code>	Mendefinisikan input atau parameter yang akan digunakan pada langkah berikutnya.

Nama variabel dipilih agar maknanya terbaca. Hindari singkatan yang tidak dijelaskan, nilai konstan tanpa satuan, serta penggunaan variabel dari sel lama. Komentar harus menerangkan alasan atau asumsi, bukan sekadar mengulang sintaks.

Validasi dan Interpretasi Hasil

Prinsip kerja yang digunakan adalah memisahkan input, proses, validasi, dan ekspor. Pemisahan ini membuat kesalahan lebih mudah dilacak dan mencegah hasil lama dianggap sebagai hasil dari parameter baru. Setiap langkah perlu meninggalkan bukti yang dapat dibaca kembali: nilai parameter, versi pustaka, status proses, nama berkas, serta pesan kesalahan bila terjadi kegagalan.

Aspek validasi	Kriteria lulus
Input	Tipe, bentuk, satuan, muatan, spin, dan jalur berkas dinyatakan eksplisit.
Lingkungan	Versi Python dan pustaka tercatat; backend CPU/GPU terdeteksi.
Proses	Status eksekusi atau konvergensi diperiksa sebelum hasil dipakai.

Numerik	Tidak ada NaN/inf; ukuran array dan jumlah atom konsisten.
Keluaran	Hasil yang dapat diperiksa dan disimpan disimpan dan dapat dibuka kembali.
Interpretasi	Klaim tidak melampaui metode, basis set, atau data yang dihitung.

Interpretasi aman: Sebutkan metode, parameter, satuan, status konvergensi, dan keterbatasan saat menuliskan kesimpulan.

Kesalahan Umum dan Troubleshooting

Risiko utama pada topik ini adalah parameter tidak terdokumentasi dan keluaran tidak tervalidasi. Karena itu, notebook harus memuat pemeriksaan eksplisit dan bukan hanya mengandalkan keluaran yang tampak masuk akal. Pembaca dianjurkan menjalankan notebook dari keadaan runtime bersih, membandingkan hasil dengan batas kewajaran, dan menyimpan artefak penting di lokasi persisten.

Gejala	Tindakan perbaikan
Sel dijalankan tidak berurutan	Restart session, lalu gunakan Run all dan periksa variabel prasyarat.
Pustaka tidak ditemukan	Jalankan kembali sel %pip, periksa nama paket, lalu restart bila dependensi inti berubah.
Input tampak benar tetapi gagal	Cetak tipe, shape, satuan, muatan, spin, dan beberapa baris pertama data.
Parameter tidak terdokumentasi dan keluaran tidak tervalidasi	Tambahkan validasi sebelum proses utama dan hentikan eksekusi dengan pesan yang spesifik.
Hasil berbeda antar-runtime	Catat versi runtime/paket, random seed, parameter, dan perangkat yang digunakan.

Aturan troubleshooting: Perbaiki satu hal pada satu waktu, simpan pesan kesalahan lengkap, dan catat perubahan versi atau parameter.

Latihan Modifikasi dan Refleksi

287. Ubah satu parameter pada contoh Muatan, jumlah elektron, multiplicity, dan spin, jalankan ulang dari runtime bersih, dan jelaskan perubahan keluaran.
288. Tambahkan satu validasi baru yang mencegah parameter tidak terdokumentasi dan keluaran tidak tervalidasi.
289. Buat berkas ringkasan TXT atau JSON yang memuat input, status proses, waktu, dan hasil utama.
290. Jelaskan bagian mana yang merupakan fakta keluaran program dan bagian mana yang merupakan interpretasi ilmiah.
291. Susun satu kasus uji yang sengaja salah, lalu pastikan notebook menghasilkan pesan kesalahan yang informatif.

Refleksi: Tuliskan apa yang berubah setelah kode dimodifikasi, bukti apa yang menunjukkan keberhasilan, dan bagian mana yang masih memerlukan verifikasi eksternal.

Rubrik	Deskripsi
Skor 4	Kode berjalan dari runtime bersih, validasi lengkap, interpretasi tepat.
Skor 3	Kode berjalan dan hasil tersimpan, tetapi dokumentasi belum lengkap.
Skor 2	Kode hanya berjalan setelah intervensi atau state tertentu.
Skor 1	Kode belum berjalan dan diagnosis kesalahan belum sistematis.

3.1.3 Sumber struktur molekul

Sumber struktur molekul merupakan simpul penting dalam alur kerja notebook ilmiah karena menentukan bagaimana struktur molekul disiapkan, dijalankan, dan diperiksa. Struktur eksperimental, basis data kimia, koordinat literatur, dan konformer komputasional. Hasil yang dituju berupa konformer tiga dimensi dan koordinat XYZ.

Prinsip kerja yang digunakan adalah memisahkan input, proses, validasi, dan ekspor. Pemisahan ini membuat kesalahan lebih mudah dilacak dan mencegah hasil lama dianggap sebagai hasil dari parameter baru. Setiap langkah perlu meninggalkan bukti yang dapat dibaca kembali: nilai parameter, versi pustaka, status proses, nama berkas, serta pesan kesalahan bila terjadi kegagalan.

Tujuan operasional: Pembaca mampu menjelaskan Sumber struktur molekul, menjalankan contoh minimal, memeriksa keberhasilan, dan menyimpan bukti hasil.

Komponen	Kriteria
Konsep	Struktur eksperimental, basis data kimia, koordinat literatur, dan konformer komputasional.
Input	Parameter dan data dinyatakan eksplisit.
Proses	Langkah dapat diulang dari runtime bersih.
Output	Nilai, berkas, atau visualisasi yang dapat diperiksa.
Batas	Interpretasi mengikuti metode dan konteks.

Landasan Teori dan Batas Penerapan

Pada bagian Sumber struktur molekul, perhatian tidak hanya diarahkan pada keberhasilan menjalankan kode, tetapi juga pada ketepatan asumsi, keterlacakan parameter, dan mutu keluaran. Struktur eksperimental, basis data kimia, koordinat literatur, dan konformer komputasional. Hasil yang dituju berupa konformer tiga dimensi dan koordinat XYZ.

Prinsip kerja yang digunakan adalah memisahkan input, proses, validasi, dan ekspor. Pemisahan ini membuat kesalahan lebih mudah dilacak dan mencegah hasil lama dianggap sebagai hasil dari parameter baru. Setiap langkah perlu meninggalkan bukti yang dapat dibaca kembali: nilai parameter, versi pustaka, status proses, nama berkas, serta pesan kesalahan bila terjadi kegagalan.

Risiko utama pada topik ini adalah struktur 2D, stereokimia, muatan, dan koordinat tidak valid. Karena itu, notebook harus memuat pemeriksaan eksplisit dan bukan hanya mengandalkan keluaran yang tampak masuk akal. Pembaca dianjurkan menjalankan notebook dari keadaan runtime bersih, membandingkan hasil dengan batas kewajaran, dan menyimpan artefak penting di lokasi persisten.

Batas penerapan: Contoh pada buku berfungsi sebagai pola kerja. Ukuran molekul, biaya komputasi, versi pustaka, dan perangkat keras dapat mengubah waktu serta strategi yang diperlukan.

Prosedur Bertahap

Pada bagian Sumber struktur molekul, perhatian tidak hanya diarahkan pada keberhasilan menjalankan kode, tetapi juga pada ketepatan asumsi, keterlacakan parameter, dan mutu keluaran. Struktur eksperimental, basis data kimia, koordinat literatur, dan konformer komputasional. Hasil yang dituju berupa konformer tiga dimensi dan koordinat XYZ.

292. Tuliskan tujuan operasional Sumber struktur molekul dan tentukan keluaran yang harus tersedia pada akhir sel.
293. Tetapkan parameter pada satu sel konfigurasi agar perubahan dapat ditelusuri dan tidak tersebar di banyak tempat.
294. Siapkan struktur molekul dengan input yang eksplisit; hindari nilai tersembunyi yang hanya tersimpan dalam state kernel.
295. Jalankan pemeriksaan prasyarat, termasuk keberadaan pustaka, bentuk data, tipe nilai, serta konsistensi satuan.
296. Eksekusi proses utama, tangkap status konvergensi atau keberhasilan, dan hentikan alur apabila syarat minimum tidak terpenuhi.
297. Simpan konformer tiga dimensi dan koordinat XYZ dengan nama berkas informatif, kemudian tulis interpretasi yang membedakan fakta komputasi dan kesimpulan ilmiah.

Titik kontrol: Jangan melanjutkan ke interpretasi apabila validasi input atau status proses belum memenuhi kriteria.

Kode Dasar Siap Colab

Pembahasan Sumber struktur molekul ditempatkan dalam konteks praktik Google Colab agar pembaca dapat menghubungkan konsep, prosedur, kode, dan interpretasi secara utuh. Struktur eksperimental, basis data kimia, koordinat literatur, dan konformer komputasional. Hasil yang dituju berupa konformer tiga dimensi dan koordinat XYZ.

Sel kode contoh

```
from pathlib import Path

SOURCES = {
    "jenis": "konformer komputasional",
    "asal": "SMILES tervalidasi",
    "metode_awal": "ETKDGv3 + MMFF94",
    "tanggal": "2026-07-27",
}

Path("provenance.txt").write_text(
    "\n".join(f"{k}: {v}" for k,v in SOURCES.items()),
    encoding="utf-8"
)
```


Cara menjalankan: Tempatkan kode dalam satu atau beberapa sel sesuai komentar. Untuk baris %pip atau perintah shell, gunakan runtime Google Colab.

Penjelasan Baris Kode Penting

Potongan kode	Fungsi
<code>from pathlib import Path</code>	Mendefinisikan input atau parameter yang akan digunakan pada langkah berikutnya.
<code>SOURCES = {</code>	Menjalankan operasi utama dan menyimpan hasil pada variabel yang jelas.
<code>"jenis": "konformer komputasional",</code>	Menjalankan operasi utama dan menyimpan hasil pada variabel yang jelas.
<code>"asal": "SMILES tervalidasi",</code>	Mendefinisikan input atau parameter yang akan digunakan pada langkah berikutnya.
<code>"metode_awal": "ETKDGv3 + MMFF94",</code>	Menjalankan operasi utama dan menyimpan hasil pada variabel yang jelas.
<code>"tanggal": "2026-07-27",</code>	Mendefinisikan input atau parameter yang akan digunakan pada langkah berikutnya.
<code>}</code>	Mencetak status agar pengguna tidak hanya bergantung pada keluaran visual.
<code>Path("provenance.txt").write_text(</code>	Mencetak status agar pengguna tidak hanya bergantung pada keluaran visual.

Nama variabel dipilih agar maknanya terbaca. Hindari singkatan yang tidak dijelaskan, nilai konstan tanpa satuan, serta penggunaan variabel dari sel lama. Komentar harus menerangkan alasan atau asumsi, bukan sekadar mengulang sintaks.

Validasi dan Interpretasi Hasil

Prinsip kerja yang digunakan adalah memisahkan input, proses, validasi, dan ekspor. Pemisahan ini membuat kesalahan lebih mudah dilacak dan mencegah hasil lama dianggap sebagai hasil dari parameter baru. Setiap langkah perlu meninggalkan bukti yang dapat dibaca kembali: nilai parameter, versi pustaka, status proses, nama berkas, serta pesan kesalahan bila terjadi kegagalan.

Aspek validasi	Kriteria lulus
Input	Tipe, bentuk, satuan, muatan, spin, dan jalur berkas dinyatakan eksplisit.
Lingkungan	Versi Python dan pustaka tercatat; backend CPU/GPU terdeteksi.
Proses	Status eksekusi atau konvergensi diperiksa sebelum hasil dipakai.
Numerik	Tidak ada NaN/inf; ukuran array dan jumlah atom konsisten.
Keluaran	Konformer tiga dimensi dan koordinat xyz disimpan dan dapat dibuka kembali.
Interpretasi	Klaim tidak melampaui metode, basis set, atau data yang dihitung.
Interpretasi aman: Sebutkan metode, parameter, satuan, status konvergensi, dan keterbatasan saat menuliskan kesimpulan.	

Kesalahan Umum dan Troubleshooting

Risiko utama pada topik ini adalah struktur 2D, stereokimia, muatan, dan koordinat tidak valid. Karena itu, notebook harus memuat pemeriksaan eksplisit dan bukan hanya mengandalkan keluaran yang tampak masuk akal. Pembaca dianjurkan menjalankan notebook dari keadaan runtime bersih, membandingkan hasil dengan batas kewajaran, dan menyimpan artefak penting di lokasi persisten.

Gejala	Tindakan perbaikan
Sel dijalankan tidak berurutan	Restart session, lalu gunakan Run all dan periksa variabel prasyarat.

Pustaka tidak ditemukan	Jalankan kembali sel %pip, periksa nama paket, lalu restart bila dependensi inti berubah.
Input tampak benar tetapi gagal	Cetak tipe, shape, satuan, muatan, spin, dan beberapa baris pertama data.
Struktur 2d, stereokimia, muatan, dan koordinat tidak valid	Tambahkan validasi sebelum proses utama dan hentikan eksekusi dengan pesan yang spesifik.
Hasil berbeda antar-runtime	Catat versi runtime/paket, random seed, parameter, dan perangkat yang digunakan.
Aturan troubleshooting: Perbaiki satu hal pada satu waktu, simpan pesan kesalahan lengkap, dan catat perubahan versi atau parameter.	

Latihan Modifikasi dan Refleksi

298. Ubah satu parameter pada contoh Sumber struktur molekul, jalankan ulang dari runtime bersih, dan jelaskan perubahan keluaran.
299. Tambahkan satu validasi baru yang mencegah struktur 2D, stereokimia, muatan, dan koordinat tidak valid.
300. Buat berkas ringkasan TXT atau JSON yang memuat input, status proses, waktu, dan hasil utama.
301. Jelaskan bagian mana yang merupakan fakta keluaran program dan bagian mana yang merupakan interpretasi ilmiah.
302. Susun satu kasus uji yang sengaja salah, lalu pastikan notebook menghasilkan pesan kesalahan yang informatif.

Refleksi: Tuliskan apa yang berubah setelah kode dimodifikasi, bukti apa yang menunjukkan keberhasilan, dan bagian mana yang masih memerlukan verifikasi eksternal.

Rubrik	Deskripsi
Skor 4	Kode berjalan dari runtime bersih, validasi lengkap, interpretasi tepat.
Skor 3	Kode berjalan dan hasil tersimpan, tetapi dokumentasi belum lengkap.

Skor 2	Kode hanya berjalan setelah intervensi atau state tertentu.
Skor 1	Kode belum berjalan dan diagnosis kesalahan belum sistematis.

3.2 Pembuatan Konformer dengan RDKit

Pembuatan Konformer dengan RDKit mengorganisasi tiga komponen yang saling bergantung. Pembahasan bergerak dari konsep, menuju prosedur, kemudian menghasilkan artefak yang dapat divalidasi dan disimpan.

Sub-subbab	Fokus
3.2.1 Mengubah SMILES menjadi molekul tiga dimensi	Penambahan atom hidrogen eksplisit dan pembentukan objek molekul.
3.2.2 ETKDGv3, MMFF94, dan UFF	Pembuatan konformer, optimasi mekanika molekul, pemilihan force field, dan fallback.
3.2.3 Validasi dan ekspor koordinat	Pemeriksaan jumlah atom, koordinat finite, stereokimia, energi force field, dan ekspor XYZ.
Produk belajar: Satu notebook terstruktur, satu berkas keluaran, satu catatan validasi, dan satu interpretasi singkat yang menyatakan batas hasil.	

Rencana Praktik dan Produk Belajar

303. Pelajari Mengubah SMILES menjadi molekul tiga dimensi dan jalankan kode dasar yang menyertainya.
304. Pelajari ETKDGv3, MMFF94, dan UFF dan jalankan kode dasar yang menyertainya.
305. Pelajari Validasi dan ekspor koordinat dan jalankan kode dasar yang menyertainya.

Komponen	Kriteria
Waktu	60-120 menit, bergantung pada komputasi.
Input	Notebook baru atau salinan notebook bab.

Produk	Kode, output, berkas hasil, dan refleksi.
Uji	Restart session lalu Run all.
Dokumentasi	Versi, parameter, status, dan waktu eksekusi.
Kriteria selesai: Semua sel dapat dijalankan berurutan, pesan kesalahan ditangani, dan hasil utama tersimpan di lokasi persisten.	

3.2.1 Mengubah SMILES menjadi molekul tiga dimensi

Mengubah SMILES menjadi molekul tiga dimensi merupakan simpul penting dalam alur kerja notebook ilmiah karena menentukan bagaimana struktur molekul disiapkan, dijalankan, dan diperiksa. Penambahan atom hidrogen eksplisit dan pembentukan objek molekul. Hasil yang dituju berupa konformer tiga dimensi dan koordinat XYZ.

Prinsip kerja yang digunakan adalah memisahkan input, proses, validasi, dan ekspor. Pemisahan ini membuat kesalahan lebih mudah dilacak dan mencegah hasil lama dianggap sebagai hasil dari parameter baru. Setiap langkah perlu meninggalkan bukti yang dapat dibaca kembali: nilai parameter, versi pustaka, status proses, nama berkas, serta pesan kesalahan bila terjadi kegagalan.

Tujuan operasional: Pembaca mampu menjelaskan Mengubah SMILES menjadi molekul tiga dimensi, menjalankan contoh minimal, memeriksa keberhasilan, dan menyimpan bukti hasil.

Komponen	Kriteria
Konsep	Penambahan atom hidrogen eksplisit dan pembentukan objek molekul.
Input	Parameter dan data dinyatakan eksplisit.
Proses	Langkah dapat diulang dari runtime bersih.
Output	Nilai, berkas, atau visualisasi yang dapat diperiksa.
Batas	Interpretasi mengikuti metode dan konteks.

Landasan Teori dan Batas Penerapan

Pada bagian Mengubah SMILES menjadi molekul tiga dimensi, perhatian tidak hanya diarahkan pada keberhasilan menjalankan kode, tetapi juga pada ketepatan asumsi, keterlacakan parameter, dan mutu keluaran. Penambahan atom hidrogen eksplisit dan pembentukan objek molekul. Hasil yang dituju berupa konformer tiga dimensi dan koordinat XYZ.

Prinsip kerja yang digunakan adalah memisahkan input, proses, validasi, dan ekspor. Pemisahan ini membuat kesalahan lebih mudah dilacak dan mencegah hasil lama dianggap sebagai hasil dari parameter baru. Setiap langkah perlu meninggalkan bukti yang dapat dibaca kembali: nilai parameter, versi pustaka, status proses, nama berkas, serta pesan kesalahan bila terjadi kegagalan.

Risiko utama pada topik ini adalah struktur 2D, stereokimia, muatan, dan koordinat tidak valid. Karena itu, notebook harus memuat pemeriksaan eksplisit dan bukan hanya mengandalkan keluaran yang tampak masuk akal. Pembaca dianjurkan menjalankan notebook dari keadaan runtime bersih, membandingkan hasil dengan batas kewajaran, dan menyimpan artefak penting di lokasi persisten.

Batas penerapan: Contoh pada buku berfungsi sebagai pola kerja. Ukuran molekul, biaya komputasi, versi pustaka, dan perangkat keras dapat mengubah waktu serta strategi yang diperlukan.

Prosedur Bertahap

Pada bagian Mengubah SMILES menjadi molekul tiga dimensi, perhatian tidak hanya diarahkan pada keberhasilan menjalankan kode, tetapi juga pada ketepatan asumsi, keterlacakan parameter, dan mutu keluaran. Penambahan atom hidrogen eksplisit dan pembentukan objek molekul. Hasil yang dituju berupa konformer tiga dimensi dan koordinat XYZ.

306. Tuliskan tujuan operasional Mengubah SMILES menjadi molekul tiga dimensi dan tentukan keluaran yang harus tersedia pada akhir sel.
307. Tetapkan parameter pada satu sel konfigurasi agar perubahan dapat ditelusuri dan tidak tersebar di banyak tempat.
308. Siapkan struktur molekul dengan input yang eksplisit; hindari nilai tersembunyi yang hanya tersimpan dalam state kernel.
309. Jalankan pemeriksaan prasyarat, termasuk keberadaan pustaka, bentuk data, tipe nilai, serta konsistensi satuan.
310. Eksekusi proses utama, tangkap status konvergensi atau keberhasilan, dan hentikan alur apabila syarat minimum tidak terpenuhi.
311. Simpan konformer tiga dimensi dan koordinat XYZ dengan nama berkas informatif, kemudian tulis interpretasi yang membedakan fakta komputasi dan kesimpulan ilmiah.

Titik kontrol: Jangan melanjutkan ke interpretasi apabila validasi input atau status proses belum memenuhi kriteria.

Kode Dasar Siap Colab

Mengubah SMILES menjadi molekul tiga dimensi merupakan simpul penting dalam alur kerja notebook ilmiah karena menentukan bagaimana struktur molekul disiapkan, dijalankan, dan diperiksa. Penambahan atom hidrogen eksplisit dan pembentukan objek molekul. Hasil yang dituju berupa konformer tiga dimensi dan koordinat XYZ.

Sel kode contoh

```
from rdkit import Chem
from rdkit.Chem import AllChem

mol = Chem.AddHs(Chem.MolFromSmiles("CCO"))
params = AllChem.ETKDGv3()
params.randomSeed = 2026
status = AllChem.EmbedMolecule(mol, params)
if status != 0:
    raise RuntimeError("Embedding 3D gagal")
print("Konformer 3D tersedia:", mol.GetNumConformers())
```

Cara menjalankan: Tempatkan kode dalam satu atau beberapa sel sesuai komentar. Untuk baris %pip atau perintah shell, gunakan runtime Google Colab.

Penjelasan Baris Kode Penting

Potongan kode	Fungsi
<code>from rdkit import Chem</code>	Menambahkan pemeriksaan untuk mencegah hasil tidak valid diteruskan.
<code>from rdkit.Chem import AllChem</code>	Menjalankan operasi utama dan menyimpan hasil pada variabel yang jelas.
<code>mol = Chem.AddHs(Chem.MolFromSmiles("CCO"))</code>	Mencetak status agar pengguna tidak hanya bergantung pada keluaran visual.

<code>params = AllChem.ETKDGv3()</code>	Mencetak status agar pengguna tidak hanya bergantung pada keluaran visual.
<code>params.randomSeed = 2026</code>	Mendefinisikan input atau parameter yang akan digunakan pada langkah berikutnya.
<code>status = AllChem.EmbedMolecule(mol, params)</code>	Menjalankan operasi utama dan menyimpan hasil pada variabel yang jelas.
<code>if status != 0:</code>	Menambahkan pemeriksaan untuk mencegah hasil tidak valid diteruskan.
<code>raise RuntimeError("Embedding 3D gagal")</code>	Menjalankan operasi utama dan menyimpan hasil pada variabel yang jelas.

Nama variabel dipilih agar maknanya terbaca. Hindari singkatan yang tidak dijelaskan, nilai konstan tanpa satuan, serta penggunaan variabel dari sel lama. Komentar harus menerangkan alasan atau asumsi, bukan sekadar mengulang sintaks.

Validasi dan Interpretasi Hasil

Prinsip kerja yang digunakan adalah memisahkan input, proses, validasi, dan ekspor. Pemisahan ini membuat kesalahan lebih mudah dilacak dan mencegah hasil lama dianggap sebagai hasil dari parameter baru. Setiap langkah perlu meninggalkan bukti yang dapat dibaca kembali: nilai parameter, versi pustaka, status proses, nama berkas, serta pesan kesalahan bila terjadi kegagalan.

Aspek validasi	Kriteria lulus
Input	Tipe, bentuk, satuan, muatan, spin, dan jalur berkas dinyatakan eksplisit.
Lingkungan	Versi Python dan pustaka tercatat; backend CPU/GPU terdeteksi.
Proses	Status eksekusi atau konvergensi diperiksa sebelum hasil dipakai.

Numerik	Tidak ada NaN/inf; ukuran array dan jumlah atom konsisten.
Keluaran	Konformer tiga dimensi dan koordinat xyz disimpan dan dapat dibuka kembali.
Interpretasi	Klaim tidak melampaui metode, basis set, atau data yang dihitung.

Interpretasi aman: Sebutkan metode, parameter, satuan, status konvergensi, dan keterbatasan saat menuliskan kesimpulan.

Kesalahan Umum dan Troubleshooting

Risiko utama pada topik ini adalah struktur 2D, stereokimia, muatan, dan koordinat tidak valid. Karena itu, notebook harus memuat pemeriksaan eksplisit dan bukan hanya mengandalkan keluaran yang tampak masuk akal. Pembaca dianjurkan menjalankan notebook dari keadaan runtime bersih, membandingkan hasil dengan batas kewajaran, dan menyimpan artefak penting di lokasi persisten.

Gejala	Tindakan perbaikan
Sel dijalankan tidak berurutan	Restart session, lalu gunakan Run all dan periksa variabel prasyarat.
Pustaka tidak ditemukan	Jalankan kembali sel %pip, periksa nama paket, lalu restart bila dependensi inti berubah.
Input tampak benar tetapi gagal	Cetak tipe, shape, satuan, muatan, spin, dan beberapa baris pertama data.
Struktur 2d, stereokimia, muatan, dan koordinat tidak valid	Tambahkan validasi sebelum proses utama dan hentikan eksekusi dengan pesan yang spesifik.
Hasil berbeda antar-runtime	Catat versi runtime/paket, random seed, parameter, dan perangkat yang digunakan.

Aturan troubleshooting: Perbaiki satu hal pada satu waktu, simpan pesan kesalahan lengkap, dan catat perubahan versi atau parameter.

Latihan Modifikasi dan Refleksi

312. Ubah satu parameter pada contoh Mengubah SMILES menjadi molekul tiga dimensi, jalankan ulang dari runtime bersih, dan jelaskan perubahan keluaran.
313. Tambahkan satu validasi baru yang mencegah struktur 2D, stereokimia, muatan, dan koordinat tidak valid.
314. Buat berkas ringkasan TXT atau JSON yang memuat input, status proses, waktu, dan hasil utama.
315. Jelaskan bagian mana yang merupakan fakta keluaran program dan bagian mana yang merupakan interpretasi ilmiah.
316. Susun satu kasus uji yang sengaja salah, lalu pastikan notebook menghasilkan pesan kesalahan yang informatif.

Refleksi: Tuliskan apa yang berubah setelah kode dimodifikasi, bukti apa yang menunjukkan keberhasilan, dan bagian mana yang masih memerlukan verifikasi eksternal.

Rubrik	Deskripsi
Skor 4	Kode berjalan dari runtime bersih, validasi lengkap, interpretasi tepat.
Skor 3	Kode berjalan dan hasil tersimpan, tetapi dokumentasi belum lengkap.
Skor 2	Kode hanya berjalan setelah intervensi atau state tertentu.
Skor 1	Kode belum berjalan dan diagnosis kesalahan belum sistematis.

3.2.2 ETKDGv3, MMFF94, dan UFF

Pada bagian ETKDGv3, MMFF94, dan UFF, perhatian tidak hanya diarahkan pada keberhasilan menjalankan kode, tetapi juga pada ketepatan asumsi, keterlacakan parameter, dan mutu keluaran. Pembuatan konformer, optimasi mekanika molekul, pemilihan force field, dan fallback. Hasil yang dituju berupa konformer tiga dimensi dan koordinat XYZ.

Prinsip kerja yang digunakan adalah memisahkan input, proses, validasi, dan ekspor. Pemisahan ini membuat kesalahan lebih mudah dilacak dan mencegah hasil lama dianggap sebagai hasil dari parameter baru. Setiap langkah perlu meninggalkan bukti yang dapat dibaca kembali: nilai parameter, versi pustaka, status proses, nama berkas, serta pesan kesalahan bila terjadi kegagalan.

Tujuan operasional: Pembaca mampu menjelaskan ETKDGv3, MMFF94, dan UFF, menjalankan contoh minimal, memeriksa keberhasilan, dan menyimpan bukti hasil.

Komponen	Kriteria
Konsep	Pembuatan konformer, optimasi mekanika molekul, pemilihan force field, dan fallback.
Input	Parameter dan data dinyatakan eksplisit.
Proses	Langkah dapat diulang dari runtime bersih.
Output	Nilai, berkas, atau visualisasi yang dapat diperiksa.
Batas	Interpretasi mengikuti metode dan konteks.

Landasan Teori dan Batas Penerapan

ETKDGv3, MMFF94, dan UFF merupakan simpul penting dalam alur kerja notebook ilmiah karena menentukan bagaimana struktur molekul disiapkan, dijalankan, dan diperiksa. Pembuatan konformer, optimasi mekanika molekul, pemilihan force field, dan fallback. Hasil yang dituju berupa konformer tiga dimensi dan koordinat XYZ.

Prinsip kerja yang digunakan adalah memisahkan input, proses, validasi, dan ekspor. Pemisahan ini membuat kesalahan lebih mudah dilacak dan mencegah hasil lama dianggap sebagai hasil dari parameter baru. Setiap langkah perlu meninggalkan bukti yang dapat dibaca kembali: nilai parameter, versi pustaka, status proses, nama berkas, serta pesan kesalahan bila terjadi kegagalan.

Risiko utama pada topik ini adalah struktur 2D, stereokimia, muatan, dan koordinat tidak valid. Karena itu, notebook harus memuat pemeriksaan eksplisit dan bukan hanya mengandalkan keluaran yang tampak masuk akal. Pembaca dianjurkan menjalankan notebook dari keadaan runtime bersih, membandingkan hasil dengan batas kewajaran, dan menyimpan artefak penting di lokasi persisten.

Batas penerapan: Contoh pada buku berfungsi sebagai pola kerja. Ukuran molekul, biaya komputasi, versi pustaka, dan perangkat keras dapat mengubah waktu serta strategi yang diperlukan.

Prosedur Bertahap

ETKDGv3, MMFF94, dan UFF merupakan simpul penting dalam alur kerja notebook ilmiah karena menentukan bagaimana struktur molekul disiapkan, dijalankan, dan diperiksa. Pembuatan konformer, optimasi mekanika molekul, pemilihan force field, dan fallback. Hasil yang dituju berupa konformer tiga dimensi dan koordinat XYZ.

317. Tuliskan tujuan operasional ETKDGv3, MMFF94, dan UFF dan tentukan keluaran yang harus tersedia pada akhir sel.
318. Tetapkan parameter pada satu sel konfigurasi agar perubahan dapat ditelusuri dan tidak tersebar di banyak tempat.
319. Siapkan struktur molekul dengan input yang eksplisit; hindari nilai tersembunyi yang hanya tersimpan dalam state kernel.
320. Jalankan pemeriksaan prasyarat, termasuk keberadaan pustaka, bentuk data, tipe nilai, serta konsistensi satuan.
321. Eksekusi proses utama, tangkap status konvergensi atau keberhasilan, dan hentikan alur apabila syarat minimum tidak terpenuhi.
322. Simpan konformer tiga dimensi dan koordinat XYZ dengan nama berkas informatif, kemudian tulis interpretasi yang membedakan fakta komputasi dan kesimpulan ilmiah.

Titik kontrol: Jangan melanjutkan ke interpretasi apabila validasi input atau status proses belum memenuhi kriteria.

Kode Dasar Siap Colab

ETKDGv3, MMFF94, dan UFF merupakan simpul penting dalam alur kerja notebook ilmiah karena menentukan bagaimana struktur molekul disiapkan, dijalankan, dan diperiksa. Pembuatan konformer, optimasi mekanika molekul, pemilihan force field, dan fallback. Hasil yang dituju berupa konformer tiga dimensi dan koordinat XYZ.

Sel kode contoh

```
from rdkit import Chem
from rdkit.Chem import AllChem

mol = Chem.AddHs(Chem.MolFromSmiles("CC(=O)NC1=CC=C(O)C=C1O"))
params = AllChem.ETKDGv3(); params.randomSeed = 2026
if AllChem.EmbedMolecule(mol, params) != 0:
    raise RuntimeError("ETKDGv3 gagal")
if AllChem.MMFFHasAllMoleculeParams(mol):
    status = AllChem.MMFFOptimizeMolecule(mol, maxIters=500)
else:
    status = AllChem.UFFOptimizeMolecule(mol, maxIters=500)
print("Status optimasi force field:", status)
```

Cara menjalankan: Tempatkan kode dalam satu atau beberapa sel sesuai komentar. Untuk baris %pip atau perintah shell, gunakan runtime Google Colab.

Penjelasan Baris Kode Penting

Potongan kode	Fungsi
<code>from rdkit import Chem</code>	Menambahkan pemeriksaan untuk mencegah hasil tidak valid diteruskan.
<code>from rdkit.Chem import AllChem</code>	Mencetak status agar pengguna tidak hanya bergantung pada keluaran visual.
<code>mol = Chem.AddHs(Chem.MolFromSmiles("CC(=O)NC1=C</code>	Mencetak status agar pengguna tidak hanya bergantung pada keluaran visual.
<code>params = AllChem.ETKDGv3(); params.randomSeed =</code>	Menambahkan pemeriksaan untuk mencegah hasil tidak valid diteruskan.
<code>if AllChem.EmbedMolecule(mol, params) != 0:</code>	Mendefinisikan input atau parameter yang akan digunakan pada langkah berikutnya.
<code>raise RuntimeError("ETKDGv3 gagal")</code>	Mencetak status agar pengguna tidak hanya bergantung pada keluaran visual.
<code>if AllChem.MMFFHasAllMoleculeParams(mol):</code>	Menjalankan operasi utama dan menyimpan hasil pada variabel yang jelas.
<code>status = AllChem.MMFFOptimizeMolecule(mol, m</code>	Mendefinisikan input atau parameter yang akan digunakan pada langkah berikutnya.

Nama variabel dipilih agar maknanya terbaca. Hindari singkatan yang tidak dijelaskan, nilai konstan tanpa satuan, serta penggunaan variabel dari sel lama. Komentar harus menerangkan alasan atau asumsi, bukan sekadar mengulang sintaks.

Validasi dan Interpretasi Hasil

Prinsip kerja yang digunakan adalah memisahkan input, proses, validasi, dan ekspor. Pemisahan ini membuat kesalahan lebih mudah dilacak dan mencegah hasil lama dianggap sebagai hasil dari parameter baru. Setiap langkah perlu meninggalkan bukti yang dapat dibaca kembali: nilai parameter, versi pustaka, status proses, nama berkas, serta pesan kesalahan bila terjadi kegagalan.

Aspek validasi	Kriteria lulus
Input	Tipe, bentuk, satuan, muatan, spin, dan jalur berkas dinyatakan eksplisit.
Lingkungan	Versi Python dan pustaka tercatat; backend CPU/GPU terdeteksi.
Proses	Status eksekusi atau konvergensi diperiksa sebelum hasil dipakai.
Numerik	Tidak ada NaN/inf; ukuran array dan jumlah atom konsisten.
Keluaran	Konformer tiga dimensi dan koordinat xyz disimpan dan dapat dibuka kembali.
Interpretasi	Klaim tidak melampaui metode, basis set, atau data yang dihitung.
Interpretasi aman: Sebutkan metode, parameter, satuan, status konvergensi, dan keterbatasan saat menuliskan kesimpulan.	

Kesalahan Umum dan Troubleshooting

Risiko utama pada topik ini adalah struktur 2D, stereokimia, muatan, dan koordinat tidak valid. Karena itu, notebook harus memuat pemeriksaan eksplisit dan bukan hanya mengandalkan keluaran yang tampak masuk akal. Pembaca dianjurkan menjalankan notebook dari keadaan runtime bersih, membandingkan hasil dengan batas kewajaran, dan menyimpan artefak penting di lokasi persisten.

Gejala

Tindakan perbaikan

Sel dijalankan tidak berurutan	Restart session, lalu gunakan Run all dan periksa variabel prasyarat.
Pustaka tidak ditemukan	Jalankan kembali sel %pip, periksa nama paket, lalu restart bila dependensi inti berubah.
Input tampak benar tetapi gagal	Cetak tipe, shape, satuan, muatan, spin, dan beberapa baris pertama data.
Struktur 2d, stereokimia, muatan, dan koordinat tidak valid	Tambahkan validasi sebelum proses utama dan hentikan eksekusi dengan pesan yang spesifik.
Hasil berbeda antar-runtime	Catat versi runtime/paket, random seed, parameter, dan perangkat yang digunakan.
Aturan troubleshooting: Perbaiki satu hal pada satu waktu, simpan pesan kesalahan lengkap, dan catat perubahan versi atau parameter.	

Latihan Modifikasi dan Refleksi

323. Ubah satu parameter pada contoh ETKDGv3, MMFF94, dan UFF, jalankan ulang dari runtime bersih, dan jelaskan perubahan keluaran.
324. Tambahkan satu validasi baru yang mencegah struktur 2D, stereokimia, muatan, dan koordinat tidak valid.
325. Buat berkas ringkasan TXT atau JSON yang memuat input, status proses, waktu, dan hasil utama.
326. Jelaskan bagian mana yang merupakan fakta keluaran program dan bagian mana yang merupakan interpretasi ilmiah.
327. Susun satu kasus uji yang sengaja salah, lalu pastikan notebook menghasilkan pesan kesalahan yang informatif.

Refleksi: Tuliskan apa yang berubah setelah kode dimodifikasi, bukti apa yang menunjukkan keberhasilan, dan bagian mana yang masih memerlukan verifikasi eksternal.

Rubrik	Deskripsi
Skor 4	Kode berjalan dari runtime bersih, validasi lengkap, interpretasi tepat.

Skor 3	Kode berjalan dan hasil tersimpan, tetapi dokumentasi belum lengkap.
Skor 2	Kode hanya berjalan setelah intervensi atau state tertentu.
Skor 1	Kode belum berjalan dan diagnosis kesalahan belum sistematis.

3.2.3 Validasi dan ekspor koordinat

Pada bagian Validasi dan ekspor koordinat, perhatian tidak hanya diarahkan pada keberhasilan menjalankan kode, tetapi juga pada ketepatan asumsi, keterlacakan parameter, dan mutu keluaran. Pemeriksaan jumlah atom, koordinat finite, stereokimia, energi force field, dan ekspor XYZ. Hasil yang dituju berupa energi, geometri, orbital, dan sifat molekul.

Prinsip kerja yang digunakan adalah memisahkan input, proses, validasi, dan ekspor. Pemisahan ini membuat kesalahan lebih mudah dilacak dan mencegah hasil lama dianggap sebagai hasil dari parameter baru. Setiap langkah perlu meninggalkan bukti yang dapat dibaca kembali: nilai parameter, versi pustaka, status proses, nama berkas, serta pesan kesalahan bila terjadi kegagalan.

Tujuan operasional: Pembaca mampu menjelaskan Validasi dan ekspor koordinat, menjalankan contoh minimal, memeriksa keberhasilan, dan menyimpan bukti hasil.

Komponen	Kriteria
Konsep	Pemeriksaan jumlah atom, koordinat finite, stereokimia, energi force field, dan ekspor XYZ.
Input	Parameter dan data dinyatakan eksplisit.
Proses	Langkah dapat diulang dari runtime bersih.
Output	Nilai, berkas, atau visualisasi yang dapat diperiksa.
Batas	Interpretasi mengikuti metode dan konteks.

Landasan Teori dan Batas Penerapan

Pada bagian Validasi dan ekspor koordinat, perhatian tidak hanya diarahkan pada keberhasilan menjalankan kode, tetapi juga pada ketepatan asumsi, keterlacakan parameter, dan mutu keluaran. Pemeriksaan jumlah atom, koordinat finite, stereokimia, energi force field, dan ekspor XYZ. Hasil yang dituju berupa energi, geometri, orbital, dan sifat molekul.

Prinsip kerja yang digunakan adalah memisahkan input, proses, validasi, dan ekspor. Pemisahan ini membuat kesalahan lebih mudah dilacak dan mencegah hasil lama dianggap sebagai hasil dari parameter baru. Setiap langkah perlu meninggalkan bukti yang dapat dibaca kembali: nilai parameter, versi pustaka, status proses, nama berkas, serta pesan kesalahan bila terjadi kegagalan.

Risiko utama pada topik ini adalah SCF tidak konvergen, spin keliru, atau interpretasi lintas metode. Karena itu, notebook harus memuat pemeriksaan eksplisit dan bukan hanya mengandalkan keluaran yang tampak masuk akal. Pembaca dianjurkan menjalankan notebook dari keadaan runtime bersih, membandingkan hasil dengan batas kewajaran, dan menyimpan artefak penting di lokasi persisten.

Batas penerapan: Contoh pada buku berfungsi sebagai pola kerja. Ukuran molekul, biaya komputasi, versi pustaka, dan perangkat keras dapat mengubah waktu serta strategi yang diperlukan.

Prosedur Bertahap

Validasi dan ekspor koordinat merupakan simpul penting dalam alur kerja notebook ilmiah karena menentukan bagaimana perhitungan struktur elektronik disiapkan, dijalankan, dan diperiksa. Pemeriksaan jumlah atom, koordinat finite, stereokimia, energi force field, dan ekspor XYZ. Hasil yang dituju berupa energi, geometri, orbital, dan sifat molekul.

328. Tuliskan tujuan operasional Validasi dan ekspor koordinat dan tentukan keluaran yang harus tersedia pada akhir sel.
329. Tetapkan parameter pada satu sel konfigurasi agar perubahan dapat ditelusuri dan tidak tersebar di banyak tempat.
330. Siapkan perhitungan struktur elektronik dengan input yang eksplisit; hindari nilai tersembunyi yang hanya tersimpan dalam state kernel.
331. Jalankan pemeriksaan prasyarat, termasuk keberadaan pustaka, bentuk data, tipe nilai, serta konsistensi satuan.
332. Eksekusi proses utama, tangkap status konvergensi atau keberhasilan, dan hentikan alur apabila syarat minimum tidak terpenuhi.
333. Simpan energi, geometri, orbital, dan sifat molekul dengan nama berkas informatif, kemudian tulis interpretasi yang membedakan fakta komputasi dan kesimpulan ilmiah.

Titik kontrol: Jangan melanjutkan ke interpretasi apabila validasi input atau status proses belum memenuhi kriteria.

Kode Dasar Siap Colab

Validasi dan ekspor koordinat merupakan simpul penting dalam alur kerja notebook ilmiah karena menentukan bagaimana perhitungan struktur elektronik disiapkan, dijalankan, dan diperiksa. Pemeriksaan jumlah atom, koordinat finite, stereokimia, energi force field, dan ekspor XYZ. Hasil yang dituju berupa energi, geometri, orbital, dan sifat molekul.

Sel kode contoh

```
import numpy as np
from pathlib import Path
from rdkit import Chem

conf = mol.GetConformer()
xyz = np.array([[conf.GetAtomPosition(i).x, conf.GetAtomPosition(i).y,
                conf.GetAtomPosition(i).z] for i in
range(mol.GetNumAtoms())])
if not np.isfinite(xyz).all():
    raise ValueError("Koordinat mengandung NaN/inf")
Chem.MolToXYZFile(mol, "molekul_awal.xyz")
print(Path("molekul_awal.xyz").read_text().splitlines()[4])
```

Cara menjalankan: Tempatkan kode dalam satu atau beberapa sel sesuai komentar. Untuk baris %pip atau perintah shell, gunakan runtime Google Colab.

Penjelasan Baris Kode Penting

Potongan kode	Fungsi
import numpy as np	Menjalankan operasi utama dan menyimpan hasil pada variabel yang jelas.
from pathlib import Path	Mendefinisikan input atau parameter yang akan digunakan pada langkah berikutnya.
from rdkit import Chem	Mendefinisikan input atau parameter yang akan digunakan pada langkah berikutnya.

<code>conf = mol.GetConformer()</code>	Mencetak status agar pengguna tidak hanya bergantung pada keluaran visual.
<code>xyz = np.array([[conf.GetAtomPosition(i).x, conf</code>	Mendefinisikan input atau parameter yang akan digunakan pada langkah berikutnya.
<code>conf.GetAtomPosition(i).z] for</code>	Mendefinisikan input atau parameter yang akan digunakan pada langkah berikutnya.
<code>if not np.isfinite(xyz).all():</code>	Mencetak status agar pengguna tidak hanya bergantung pada keluaran visual.
<code>raise ValueError("Koordinat mengandung NaN/i</code>	Mendefinisikan input atau parameter yang akan digunakan pada langkah berikutnya.

Nama variabel dipilih agar maknanya terbaca. Hindari singkatan yang tidak dijelaskan, nilai konstan tanpa satuan, serta penggunaan variabel dari sel lama. Komentar harus menerangkan alasan atau asumsi, bukan sekadar mengulang sintaks.

Validasi dan Interpretasi Hasil

Prinsip kerja yang digunakan adalah memisahkan input, proses, validasi, dan ekspor. Pemisahan ini membuat kesalahan lebih mudah dilacak dan mencegah hasil lama dianggap sebagai hasil dari parameter baru. Setiap langkah perlu meninggalkan bukti yang dapat dibaca kembali: nilai parameter, versi pustaka, status proses, nama berkas, serta pesan kesalahan bila terjadi kegagalan.

Aspek validasi	Kriteria lulus
Input	Tipe, bentuk, satuan, muatan, spin, dan jalur berkas dinyatakan eksplisit.
Lingkungan	Versi Python dan pustaka tercatat; backend CPU/GPU terdeteksi.
Proses	Status eksekusi atau konvergensi diperiksa sebelum hasil dipakai.

Numerik	Tidak ada NaN/inf; ukuran array dan jumlah atom konsisten.
Keluaran	Energi, geometri, orbital, dan sifat molekul disimpan dan dapat dibuka kembali.
Interpretasi	Klaim tidak melampaui metode, basis set, atau data yang dihitung.

Interpretasi aman: Sebutkan metode, parameter, satuan, status konvergensi, dan keterbatasan saat menuliskan kesimpulan.

Kesalahan Umum dan Troubleshooting

Risiko utama pada topik ini adalah SCF tidak konvergen, spin keliru, atau interpretasi lintas metode. Karena itu, notebook harus memuat pemeriksaan eksplisit dan bukan hanya mengandalkan keluaran yang tampak masuk akal. Pembaca dianjurkan menjalankan notebook dari keadaan runtime bersih, membandingkan hasil dengan batas kewajaran, dan menyimpan artefak penting di lokasi persisten.

Gejala	Tindakan perbaikan
Sel dijalankan tidak berurutan	Restart session, lalu gunakan Run all dan periksa variabel prasyarat.
Pustaka tidak ditemukan	Jalankan kembali sel %pip, periksa nama paket, lalu restart bila dependensi inti berubah.
Input tampak benar tetapi gagal	Cetak tipe, shape, satuan, muatan, spin, dan beberapa baris pertama data.
Scf tidak konvergen, spin keliru, atau interpretasi lintas metode	Tambahkan validasi sebelum proses utama dan hentikan eksekusi dengan pesan yang spesifik.
Hasil berbeda antar-runtime	Catat versi runtime/paket, random seed, parameter, dan perangkat yang digunakan.

Aturan troubleshooting: Perbaiki satu hal pada satu waktu, simpan pesan kesalahan lengkap, dan catat perubahan versi atau parameter.

Latihan Modifikasi dan Refleksi

334. Ubah satu parameter pada contoh Validasi dan ekspor koordinat, jalankan ulang dari runtime bersih, dan jelaskan perubahan keluaran.
335. Tambahkan satu validasi baru yang mencegah SCF tidak konvergen, spin keliru, atau interpretasi lintas metode.
336. Buat berkas ringkasan TXT atau JSON yang memuat input, status proses, waktu, dan hasil utama.
337. Jelaskan bagian mana yang merupakan fakta keluaran program dan bagian mana yang merupakan interpretasi ilmiah.
338. Susun satu kasus uji yang sengaja salah, lalu pastikan notebook menghasilkan pesan kesalahan yang informatif.

Refleksi: Tuliskan apa yang berubah setelah kode dimodifikasi, bukti apa yang menunjukkan keberhasilan, dan bagian mana yang masih memerlukan verifikasi eksternal.

Rubrik	Deskripsi
Skor 4	Kode berjalan dari runtime bersih, validasi lengkap, interpretasi tepat.
Skor 3	Kode berjalan dan hasil tersimpan, tetapi dokumentasi belum lengkap.
Skor 2	Kode hanya berjalan setelah intervensi atau state tertentu.
Skor 1	Kode belum berjalan dan diagnosis kesalahan belum sistematis.

3.3 Visualisasi Molekul dengan py3Dmol

Visualisasi Molekul dengan py3Dmol mengorganisasi tiga komponen yang saling bergantung. Pembahasan bergerak dari konsep, menuju prosedur, kemudian menghasilkan artefak yang dapat divalidasi dan disimpan.

Sub-subbab	Fokus
3.3.1 Menampilkan struktur awal	Model stick, sphere, ball-and-stick, ukuran kanvas, dan pengaturan zoom.
3.3.2 Memberikan label dan informasi geometri	Nomor atom, simbol unsur, panjang ikatan, sudut ikatan, dan komentar struktur.

3.3.3 Membandingkan struktur awal dan akhir

Visualisasi berdampingan, superposisi, serta penjelasan perubahan geometri.

Produk belajar: Satu notebook terstruktur, satu berkas keluaran, satu catatan validasi, dan satu interpretasi singkat yang menyatakan batas hasil.

Rencana Praktik dan Produk Belajar

339. Pelajari Menampilkan struktur awal dan jalankan kode dasar yang menyertainya.
340. Pelajari Memberikan label dan informasi geometri dan jalankan kode dasar yang menyertainya.
341. Pelajari Membandingkan struktur awal dan akhir dan jalankan kode dasar yang menyertainya.

Komponen	Kriteria
Waktu	60-120 menit, bergantung pada komputasi.
Input	Notebook baru atau salinan notebook bab.
Produk	Kode, output, berkas hasil, dan refleksi.
Uji	Restart session lalu Run all.
Dokumentasi	Versi, parameter, status, dan waktu eksekusi.
Kriteria selesai: Semua sel dapat dijalankan berurutan, pesan kesalahan ditangani, dan hasil utama tersimpan di lokasi persisten.	

3.3.1 Menampilkan struktur awal

Pada bagian Menampilkan struktur awal, perhatian tidak hanya diarahkan pada keberhasilan menjalankan kode, tetapi juga pada ketepatan asumsi, keterlacakan parameter, dan mutu keluaran. Model stick, sphere, ball-and-stick, ukuran kanvas, dan pengaturan zoom. Hasil yang dituju berupa hasil yang dapat diperiksa dan disimpan.

Prinsip kerja yang digunakan adalah memisahkan input, proses, validasi, dan ekspor. Pemisahan ini membuat kesalahan lebih mudah dilacak dan mencegah hasil lama dianggap sebagai hasil dari parameter baru. Setiap langkah perlu meninggalkan bukti yang dapat dibaca kembali: nilai parameter, versi pustaka, status proses, nama berkas, serta pesan kesalahan bila terjadi kegagalan.

Tujuan operasional: Pembaca mampu menjelaskan Menampilkan struktur awal, menjalankan contoh minimal, memeriksa keberhasilan, dan menyimpan bukti hasil.

Komponen	Kriteria
Konsep	Model stick, sphere, ball-and-stick, ukuran kanvas, dan pengaturan zoom.
Input	Parameter dan data dinyatakan eksplisit.
Proses	Langkah dapat diulang dari runtime bersih.
Output	Nilai, berkas, atau visualisasi yang dapat diperiksa.
Batas	Interpretasi mengikuti metode dan konteks.

Landasan Teori dan Batas Penerapan

Pembahasan Menampilkan struktur awal ditempatkan dalam konteks praktik Google Colab agar pembaca dapat menghubungkan konsep, prosedur, kode, dan interpretasi secara utuh. Model stick, sphere, ball-and-stick, ukuran kanvas, dan pengaturan zoom. Hasil yang dituju berupa hasil yang dapat diperiksa dan disimpan.

Prinsip kerja yang digunakan adalah memisahkan input, proses, validasi, dan ekspor. Pemisahan ini membuat kesalahan lebih mudah dilacak dan mencegah hasil lama dianggap sebagai hasil dari parameter baru. Setiap langkah perlu meninggalkan bukti yang dapat dibaca kembali: nilai parameter, versi pustaka, status proses, nama berkas, serta pesan kesalahan bila terjadi kegagalan.

Risiko utama pada topik ini adalah parameter tidak terdokumentasi dan keluaran tidak tervalidasi. Karena itu, notebook harus memuat pemeriksaan eksplisit dan bukan hanya mengandalkan keluaran yang tampak masuk akal. Pembaca dianjurkan menjalankan notebook dari keadaan runtime bersih, membandingkan hasil dengan batas kewajaran, dan menyimpan artefak penting di lokasi persisten.

Batas penerapan: Contoh pada buku berfungsi sebagai pola kerja. Ukuran molekul, biaya komputasi, versi pustaka, dan perangkat keras dapat mengubah waktu serta strategi yang diperlukan.

Prosedur Bertahap

Menampilkan struktur awal merupakan simpul penting dalam alur kerja notebook ilmiah karena menentukan bagaimana prosedur komputasi disiapkan, dijalankan, dan diperiksa. Model stick, sphere, ball-and-stick, ukuran kanvas, dan pengaturan zoom. Hasil yang dituju berupa hasil yang dapat diperiksa dan disimpan.

342. Tuliskan tujuan operasional Menampilkan struktur awal dan tentukan keluaran yang harus tersedia pada akhir sel.
343. Tetapkan parameter pada satu sel konfigurasi agar perubahan dapat ditelusuri dan tidak tersebar di banyak tempat.
344. Siapkan prosedur komputasi dengan input yang eksplisit; hindari nilai tersembunyi yang hanya tersimpan dalam state kernel.
345. Jalankan pemeriksaan prasyarat, termasuk keberadaan pustaka, bentuk data, tipe nilai, serta konsistensi satuan.
346. Eksekusi proses utama, tangkap status konvergensi atau keberhasilan, dan hentikan alur apabila syarat minimum tidak terpenuhi.
347. Simpan hasil yang dapat diperiksa dan disimpan dengan nama berkas informatif, kemudian tulis interpretasi yang membedakan fakta komputasi dan kesimpulan ilmiah.

Titik kontrol: Jangan melanjutkan ke interpretasi apabila validasi input atau status proses belum memenuhi kriteria.

Kode Dasar Siap Colab

Pembahasan Menampilkan struktur awal ditempatkan dalam konteks praktik Google Colab agar pembaca dapat menghubungkan konsep, prosedur, kode, dan interpretasi secara utuh. Model stick, sphere, ball-and-stick, ukuran kanvas, dan pengaturan zoom. Hasil yang dituju berupa hasil yang dapat diperiksa dan disimpan.

Sel kode contoh

```
import py3Dmol

xyz_text = open("molekul_awal.xyz", encoding="utf-8").read()
view = py3Dmol.view(width=700, height=450)
view.addModel(xyz_text, "xyz")
view.setStyle({"stick": {}, "sphere": {"scale": 0.30}})
view.zoomTo()
view.show()
```

Cara menjalankan: Tempatkan kode dalam satu atau beberapa sel sesuai komentar. Untuk baris %pip atau perintah shell, gunakan runtime Google Colab.

Penjelasan Baris Kode Penting

Potongan kode	Fungsi
<code>import py3Dmol</code>	Menambahkan pemeriksaan untuk mencegah hasil tidak valid diteruskan.
<code>xyz_text = open("molekul_awal.xyz", encoding="ut"</code>	Menambahkan pemeriksaan untuk mencegah hasil tidak valid diteruskan.
<code>view = py3Dmol.view(width=700, height=450)</code>	Mencetak status agar pengguna tidak hanya bergantung pada keluaran visual.
<code>view.addModel(xyz_text, "xyz")</code>	Mendefinisikan input atau parameter yang akan digunakan pada langkah berikutnya.
<code>view.setStyle({"stick": {}, "sphere": {"scale":</code>	Menambahkan pemeriksaan untuk mencegah hasil tidak valid diteruskan.
<code>view.zoomTo()</code>	Menambahkan pemeriksaan untuk mencegah hasil tidak valid diteruskan.
<code>view.show()</code>	Menambahkan pemeriksaan untuk mencegah hasil tidak valid diteruskan.

Nama variabel dipilih agar maknanya terbaca. Hindari singkatan yang tidak dijelaskan, nilai konstan tanpa satuan, serta penggunaan variabel dari sel lama. Komentar harus menerangkan alasan atau asumsi, bukan sekadar mengulang sintaks.

Validasi dan Interpretasi Hasil

Prinsip kerja yang digunakan adalah memisahkan input, proses, validasi, dan ekspor. Pemisahan ini membuat kesalahan lebih mudah dilacak dan mencegah hasil lama dianggap sebagai hasil dari parameter baru. Setiap langkah perlu meninggalkan bukti yang dapat dibaca kembali: nilai parameter, versi pustaka, status proses, nama berkas, serta pesan kesalahan bila terjadi kegagalan.

Aspek validasi	Kriteria lulus
Input	Tipe, bentuk, satuan, muatan, spin, dan jalur berkas dinyatakan eksplisit.

Lingkungan	Versi Python dan pustaka tercatat; backend CPU/GPU terdeteksi.
Proses	Status eksekusi atau konvergensi diperiksa sebelum hasil dipakai.
Numerik	Tidak ada NaN/inf; ukuran array dan jumlah atom konsisten.
Keluaran	Hasil yang dapat diperiksa dan disimpan disimpan dan dapat dibuka kembali.
Interpretasi	Klaim tidak melampaui metode, basis set, atau data yang dihitung.

Interpretasi aman: Sebutkan metode, parameter, satuan, status konvergensi, dan keterbatasan saat menuliskan kesimpulan.

Kesalahan Umum dan Troubleshooting

Risiko utama pada topik ini adalah parameter tidak terdokumentasi dan keluaran tidak tervalidasi. Karena itu, notebook harus memuat pemeriksaan eksplisit dan bukan hanya mengandalkan keluaran yang tampak masuk akal. Pembaca dianjurkan menjalankan notebook dari keadaan runtime bersih, membandingkan hasil dengan batas kewajaran, dan menyimpan artefak penting di lokasi persisten.

Gejala	Tindakan perbaikan
Sel dijalankan tidak berurutan	Restart session, lalu gunakan Run all dan periksa variabel prasyarat.
Pustaka tidak ditemukan	Jalankan kembali sel %pip, periksa nama paket, lalu restart bila dependensi inti berubah.
Input tampak benar tetapi gagal	Cetak tipe, shape, satuan, muatan, spin, dan beberapa baris pertama data.
Parameter tidak terdokumentasi dan keluaran tidak tervalidasi	Tambahkan validasi sebelum proses utama dan hentikan eksekusi dengan pesan yang spesifik.

Hasil berbeda antar-runtime

Catat versi runtime/paket, random seed, parameter, dan perangkat yang digunakan.

Aturan troubleshooting: Perbaiki satu hal pada satu waktu, simpan pesan kesalahan lengkap, dan catat perubahan versi atau parameter.

Latihan Modifikasi dan Refleksi

348. Ubah satu parameter pada contoh Menampilkan struktur awal, jalankan ulang dari runtime bersih, dan jelaskan perubahan keluaran.
349. Tambahkan satu validasi baru yang mencegah parameter tidak terdokumentasi dan keluaran tidak tervalidasi.
350. Buat berkas ringkasan TXT atau JSON yang memuat input, status proses, waktu, dan hasil utama.
351. Jelaskan bagian mana yang merupakan fakta keluaran program dan bagian mana yang merupakan interpretasi ilmiah.
352. Susun satu kasus uji yang sengaja salah, lalu pastikan notebook menghasilkan pesan kesalahan yang informatif.

Refleksi: Tuliskan apa yang berubah setelah kode dimodifikasi, bukti apa yang menunjukkan keberhasilan, dan bagian mana yang masih memerlukan verifikasi eksternal.

Rubrik	Deskripsi
Skor 4	Kode berjalan dari runtime bersih, validasi lengkap, interpretasi tepat.
Skor 3	Kode berjalan dan hasil tersimpan, tetapi dokumentasi belum lengkap.
Skor 2	Kode hanya berjalan setelah intervensi atau state tertentu.
Skor 1	Kode belum berjalan dan diagnosis kesalahan belum sistematis.

3.3.2 Memberikan label dan informasi geometri

Memberikan label dan informasi geometri merupakan simpul penting dalam alur kerja notebook ilmiah karena menentukan bagaimana prosedur komputasi disiapkan, dijalankan, dan diperiksa. Nomor atom, simbol unsur, panjang ikatan, sudut ikatan, dan komentar struktur. Hasil yang dituju berupa hasil yang dapat diperiksa dan disimpan.

Prinsip kerja yang digunakan adalah memisahkan input, proses, validasi, dan ekspor. Pemisahan ini membuat kesalahan lebih mudah dilacak dan mencegah hasil lama dianggap sebagai hasil dari parameter baru. Setiap langkah perlu meninggalkan bukti yang dapat dibaca kembali: nilai parameter, versi pustaka, status proses, nama berkas, serta pesan kesalahan bila terjadi kegagalan.

Tujuan operasional: Pembaca mampu menjelaskan Memberikan label dan informasi geometri, menjalankan contoh minimal, memeriksa keberhasilan, dan menyimpan bukti hasil.

Komponen	Kriteria
Konsep	Nomor atom, simbol unsur, panjang ikatan, sudut ikatan, dan komentar struktur.
Input	Parameter dan data dinyatakan eksplisit.
Proses	Langkah dapat diulang dari runtime bersih.
Output	Nilai, berkas, atau visualisasi yang dapat diperiksa.
Batas	Interpretasi mengikuti metode dan konteks.

Landasan Teori dan Batas Penerapan

Pembahasan Memberikan label dan informasi geometri ditempatkan dalam konteks praktik Google Colab agar pembaca dapat menghubungkan konsep, prosedur, kode, dan interpretasi secara utuh. Nomor atom, simbol unsur, panjang ikatan, sudut ikatan, dan komentar struktur. Hasil yang dituju berupa hasil yang dapat diperiksa dan disimpan.

Prinsip kerja yang digunakan adalah memisahkan input, proses, validasi, dan ekspor. Pemisahan ini membuat kesalahan lebih mudah dilacak dan mencegah hasil lama dianggap sebagai hasil dari parameter baru. Setiap langkah perlu meninggalkan bukti yang dapat dibaca kembali: nilai parameter, versi pustaka, status proses, nama berkas, serta pesan kesalahan bila terjadi kegagalan.

Risiko utama pada topik ini adalah parameter tidak terdokumentasi dan keluaran tidak tervalidasi. Karena itu, notebook harus memuat pemeriksaan eksplisit dan bukan hanya mengandalkan keluaran yang tampak masuk akal. Pembaca dianjurkan menjalankan notebook dari keadaan runtime bersih, membandingkan hasil dengan batas kewajaran, dan menyimpan artefak penting di lokasi persisten.

Batas penerapan: Contoh pada buku berfungsi sebagai pola kerja. Ukuran molekul, biaya komputasi, versi pustaka, dan perangkat keras dapat mengubah waktu serta strategi yang diperlukan.

Prosedur Bertahap

Memberikan label dan informasi geometri merupakan simpul penting dalam alur kerja notebook ilmiah karena menentukan bagaimana prosedur komputasi disiapkan, dijalankan, dan diperiksa. Nomor atom, simbol unsur, panjang ikatan, sudut ikatan, dan komentar struktur. Hasil yang dituju berupa hasil yang dapat diperiksa dan disimpan.

353. Tuliskan tujuan operasional Memberikan label dan informasi geometri dan tentukan keluaran yang harus tersedia pada akhir sel.
354. Tetapkan parameter pada satu sel konfigurasi agar perubahan dapat ditelusuri dan tidak tersebar di banyak tempat.
355. Siapkan prosedur komputasi dengan input yang eksplisit; hindari nilai tersembunyi yang hanya tersimpan dalam state kernel.
356. Jalankan pemeriksaan prasyarat, termasuk keberadaan pustaka, bentuk data, tipe nilai, serta konsistensi satuan.
357. Eksekusi proses utama, tangkap status konvergensi atau keberhasilan, dan hentikan alur apabila syarat minimum tidak terpenuhi.
358. Simpan hasil yang dapat diperiksa dan disimpan dengan nama berkas informatif, kemudian tulis interpretasi yang membedakan fakta komputasi dan kesimpulan ilmiah.

Titik kontrol: Jangan melanjutkan ke interpretasi apabila validasi input atau status proses belum memenuhi kriteria.

Kode Dasar Siap Colab

Pada bagian Memberikan label dan informasi geometri, perhatian tidak hanya diarahkan pada keberhasilan menjalankan kode, tetapi juga pada ketepatan asumsi, keterlacakan parameter, dan mutu keluaran. Nomor atom, simbol unsur, panjang ikatan, sudut ikatan, dan komentar struktur. Hasil yang dituju berupa hasil yang dapat diperiksa dan disimpan.

Sel kode contoh

```
import numpy as np

def bond_length(coords, i, j):
    return float(np.linalg.norm(coords[i] - coords[j]))

for i, atom in enumerate(mol.GetAtoms()):
    p = conf.GetAtomPosition(i)
```

```
print(f"{i:3d} {atom.GetSymbol():2s} {p.x:9.4f} {p.y:9.4f} {p.z:9.4f}")  
print("Contoh jarak atom 0-1:", bond_length(xyz, 0, 1))
```

Cara menjalankan: Tempatkan kode dalam satu atau beberapa sel sesuai komentar. Untuk baris %pip atau perintah shell, gunakan runtime Google Colab.

Penjelasan Baris Kode Penting

Potongan kode	Fungsi
<code>import numpy as np</code>	Mendefinisikan input atau parameter yang akan digunakan pada langkah berikutnya.
<code>def bond_length(coords, i, j):</code>	Menjalankan operasi utama dan menyimpan hasil pada variabel yang jelas.
<code>return float(np.linalg.norm(coords[i] - coord</code>	Mencetak status agar pengguna tidak hanya bergantung pada keluaran visual.
<code>for i, atom in enumerate(mol.GetAtoms()):</code>	Mendefinisikan input atau parameter yang akan digunakan pada langkah berikutnya.
<code>p = conf.GetAtomPosition(i)</code>	Mencetak status agar pengguna tidak hanya bergantung pada keluaran visual.
<code>print(f"{i:3d} {atom.GetSymbol():2s} {p.x:9.</code>	Menambahkan pemeriksaan untuk mencegah hasil tidak valid diteruskan.
<code>print("Contoh jarak atom 0-1:", bond_length(xyz,</code>	Mencetak status agar pengguna tidak hanya bergantung pada keluaran visual.

Nama variabel dipilih agar maknanya terbaca. Hindari singkatan yang tidak dijelaskan, nilai konstan tanpa satuan, serta penggunaan variabel dari sel lama. Komentar harus menerangkan alasan atau asumsi, bukan sekadar mengulang sintaks.

Validasi dan Interpretasi Hasil

Prinsip kerja yang digunakan adalah memisahkan input, proses, validasi, dan ekspor. Pemisahan ini membuat kesalahan lebih mudah dilacak dan mencegah hasil lama dianggap sebagai hasil dari parameter baru. Setiap langkah perlu meninggalkan bukti

yang dapat dibaca kembali: nilai parameter, versi pustaka, status proses, nama berkas, serta pesan kesalahan bila terjadi kegagalan.

Aspek validasi	Kriteria lulus
Input	Tipe, bentuk, satuan, muatan, spin, dan jalur berkas dinyatakan eksplisit.
Lingkungan	Versi Python dan pustaka tercatat; backend CPU/GPU terdeteksi.
Proses	Status eksekusi atau konvergensi diperiksa sebelum hasil dipakai.
Numerik	Tidak ada NaN/inf; ukuran array dan jumlah atom konsisten.
Keluaran	Hasil yang dapat diperiksa dan disimpan disimpan dan dapat dibuka kembali.
Interpretasi	Klaim tidak melampaui metode, basis set, atau data yang dihitung.
Interpretasi aman: Sebutkan metode, parameter, satuan, status konvergensi, dan keterbatasan saat menuliskan kesimpulan.	

Kesalahan Umum dan Troubleshooting

Risiko utama pada topik ini adalah parameter tidak terdokumentasi dan keluaran tidak tervalidasi. Karena itu, notebook harus memuat pemeriksaan eksplisit dan bukan hanya mengandalkan keluaran yang tampak masuk akal. Pembaca dianjurkan menjalankan notebook dari keadaan runtime bersih, membandingkan hasil dengan batas kewajaran, dan menyimpan artefak penting di lokasi persisten.

Gejala	Tindakan perbaikan
Sel dijalankan tidak berurutan	Restart session, lalu gunakan Run all dan periksa variabel prasyarat.
Pustaka tidak ditemukan	Jalankan kembali sel %pip, periksa nama paket, lalu restart bila dependensi inti berubah.

Input tampak benar tetapi gagal	Cetak tipe, shape, satuan, muatan, spin, dan beberapa baris pertama data.
Parameter tidak terdokumentasi dan keluaran tidak tervalidasi	Tambahkan validasi sebelum proses utama dan hentikan eksekusi dengan pesan yang spesifik.
Hasil berbeda antar-runtime	Catat versi runtime/paket, random seed, parameter, dan perangkat yang digunakan.
Aturan troubleshooting: Perbaiki satu hal pada satu waktu, simpan pesan kesalahan lengkap, dan catat perubahan versi atau parameter.	

Latihan Modifikasi dan Refleksi

359. Ubah satu parameter pada contoh Memberikan label dan informasi geometri, jalankan ulang dari runtime bersih, dan jelaskan perubahan keluaran.
360. Tambahkan satu validasi baru yang mencegah parameter tidak terdokumentasi dan keluaran tidak tervalidasi.
361. Buat berkas ringkasan TXT atau JSON yang memuat input, status proses, waktu, dan hasil utama.
362. Jelaskan bagian mana yang merupakan fakta keluaran program dan bagian mana yang merupakan interpretasi ilmiah.
363. Susun satu kasus uji yang sengaja salah, lalu pastikan notebook menghasilkan pesan kesalahan yang informatif.

Refleksi: Tuliskan apa yang berubah setelah kode dimodifikasi, bukti apa yang menunjukkan keberhasilan, dan bagian mana yang masih memerlukan verifikasi eksternal.

Rubrik	Deskripsi
Skor 4	Kode berjalan dari runtime bersih, validasi lengkap, interpretasi tepat.
Skor 3	Kode berjalan dan hasil tersimpan, tetapi dokumentasi belum lengkap.
Skor 2	Kode hanya berjalan setelah intervensi atau state tertentu.
Skor 1	Kode belum berjalan dan diagnosis kesalahan belum sistematis.

3.3.3 Membandingkan struktur awal dan akhir

Pada bagian Membandingkan struktur awal dan akhir, perhatian tidak hanya diarahkan pada keberhasilan menjalankan kode, tetapi juga pada ketepatan asumsi, keterlacakan parameter, dan mutu keluaran. Visualisasi berdampingan, superposisi, serta penjelasan perubahan geometri. Hasil yang dituju berupa hasil yang dapat diperiksa dan disimpan.

Prinsip kerja yang digunakan adalah memisahkan input, proses, validasi, dan ekspor. Pemisahan ini membuat kesalahan lebih mudah dilacak dan mencegah hasil lama dianggap sebagai hasil dari parameter baru. Setiap langkah perlu meninggalkan bukti yang dapat dibaca kembali: nilai parameter, versi pustaka, status proses, nama berkas, serta pesan kesalahan bila terjadi kegagalan.

Tujuan operasional: Pembaca mampu menjelaskan Membandingkan struktur awal dan akhir, menjalankan contoh minimal, memeriksa keberhasilan, dan menyimpan bukti hasil.

Komponen	Kriteria
Konsep	Visualisasi berdampingan, superposisi, serta penjelasan perubahan geometri.
Input	Parameter dan data dinyatakan eksplisit.
Proses	Langkah dapat diulang dari runtime bersih.
Output	Nilai, berkas, atau visualisasi yang dapat diperiksa.
Batas	Interpretasi mengikuti metode dan konteks.

Landasan Teori dan Batas Penerapan

Membandingkan struktur awal dan akhir merupakan simpul penting dalam alur kerja notebook ilmiah karena menentukan bagaimana prosedur komputasi disiapkan, dijalankan, dan diperiksa. Visualisasi berdampingan, superposisi, serta penjelasan perubahan geometri. Hasil yang dituju berupa hasil yang dapat diperiksa dan disimpan.

Prinsip kerja yang digunakan adalah memisahkan input, proses, validasi, dan ekspor. Pemisahan ini membuat kesalahan lebih mudah dilacak dan mencegah hasil lama dianggap sebagai hasil dari parameter baru. Setiap langkah perlu meninggalkan bukti yang dapat dibaca kembali: nilai parameter, versi pustaka, status proses, nama berkas, serta pesan kesalahan bila terjadi kegagalan.

Risiko utama pada topik ini adalah parameter tidak terdokumentasi dan keluaran tidak tervalidasi. Karena itu, notebook harus memuat pemeriksaan eksplisit dan bukan hanya mengandalkan keluaran yang tampak masuk akal. Pembaca dianjurkan menjalankan notebook dari keadaan runtime bersih, membandingkan hasil dengan batas kewajaran, dan menyimpan artefak penting di lokasi persisten.

Batas penerapan: Contoh pada buku berfungsi sebagai pola kerja. Ukuran molekul, biaya komputasi, versi pustaka, dan perangkat keras dapat mengubah waktu serta strategi yang diperlukan.

Prosedur Bertahap

Pada bagian Membandingkan struktur awal dan akhir, perhatian tidak hanya diarahkan pada keberhasilan menjalankan kode, tetapi juga pada ketepatan asumsi, keterlacakan parameter, dan mutu keluaran. Visualisasi berdampingan, superposisi, serta penjelasan perubahan geometri. Hasil yang dituju berupa hasil yang dapat diperiksa dan disimpan.

364. Tuliskan tujuan operasional Membandingkan struktur awal dan akhir dan tentukan keluaran yang harus tersedia pada akhir sel.
365. Tetapkan parameter pada satu sel konfigurasi agar perubahan dapat ditelusuri dan tidak tersebar di banyak tempat.
366. Siapkan prosedur komputasi dengan input yang eksplisit; hindari nilai tersembunyi yang hanya tersimpan dalam state kernel.
367. Jalankan pemeriksaan prasyarat, termasuk keberadaan pustaka, bentuk data, tipe nilai, serta konsistensi satuan.
368. Eksekusi proses utama, tangkap status konvergensi atau keberhasilan, dan hentikan alur apabila syarat minimum tidak terpenuhi.
369. Simpan hasil yang dapat diperiksa dan disimpan dengan nama berkas informatif, kemudian tulis interpretasi yang membedakan fakta komputasi dan kesimpulan ilmiah.

Titik kontrol: Jangan melanjutkan ke interpretasi apabila validasi input atau status proses belum memenuhi kriteria.

Kode Dasar Siap Colab

Pada bagian Membandingkan struktur awal dan akhir, perhatian tidak hanya diarahkan pada keberhasilan menjalankan kode, tetapi juga pada ketepatan asumsi, keterlacakan parameter, dan mutu keluaran. Visualisasi berdampingan, superposisi, serta penjelasan perubahan geometri. Hasil yang dituju berupa hasil yang dapat diperiksa dan disimpan.

Sel kode contoh

```
from rdkit.Chem import rdMolAlign
```

```
# mol_awal dan mol_akhir harus memiliki urutan atom yang sama
rmsd = rdMolAlign.AlignMol(mol_akhir, mol_awal)
print(f"RMSD setelah penyelarasan: {rmsd:.5f} Å")

# Bandingkan pula panjang ikatan, sudut, dan energi; RMSD saja tidak cukup.
```

Cara menjalankan: Tempatkan kode dalam satu atau beberapa sel sesuai komentar. Untuk baris %pip atau perintah shell, gunakan runtime Google Colab.

Penjelasan Baris Kode Penting

Potongan kode	Fungsi
<code>from rdkit.Chem import rdMolAlign</code>	Menjalankan operasi utama dan menyimpan hasil pada variabel yang jelas.
<code># mol_awal dan mol_akhir harus memiliki urutan a</code>	Menambahkan pemeriksaan untuk mencegah hasil tidak valid diteruskan.
<code>rmsd = rdMolAlign.AlignMol(mol_akhir, mol_awal)</code>	Mencetak status agar pengguna tidak hanya bergantung pada keluaran visual.
<code>print(f"RMSD setelah penyelarasan: {rmsd:.5f} Å")</code>	Menambahkan pemeriksaan untuk mencegah hasil tidak valid diteruskan.
<code># Bandingkan pula panjang ikatan, sudut, dan ene</code>	Mendefinisikan input atau parameter yang akan digunakan pada langkah berikutnya.

Nama variabel dipilih agar maknanya terbaca. Hindari singkatan yang tidak dijelaskan, nilai konstan tanpa satuan, serta penggunaan variabel dari sel lama. Komentar harus menerangkan alasan atau asumsi, bukan sekadar mengulang sintaks.

Validasi dan Interpretasi Hasil

Prinsip kerja yang digunakan adalah memisahkan input, proses, validasi, dan ekspor. Pemisahan ini membuat kesalahan lebih mudah dilacak dan mencegah hasil lama dianggap sebagai hasil dari parameter baru. Setiap langkah perlu meninggalkan bukti yang dapat dibaca kembali: nilai parameter, versi pustaka, status proses, nama berkas, serta pesan kesalahan bila terjadi kegagalan.

Aspek validasi

Kriteria lulus

Input	Tipe, bentuk, satuan, muatan, spin, dan jalur berkas dinyatakan eksplisit.
Lingkungan	Versi Python dan pustaka tercatat; backend CPU/GPU terdeteksi.
Proses	Status eksekusi atau konvergensi diperiksa sebelum hasil dipakai.
Numerik	Tidak ada NaN/inf; ukuran array dan jumlah atom konsisten.
Keluaran	Hasil yang dapat diperiksa dan disimpan disimpan dan dapat dibuka kembali.
Interpretasi	Klaim tidak melampaui metode, basis set, atau data yang dihitung.
Interpretasi aman: Sebutkan metode, parameter, satuan, status konvergensi, dan keterbatasan saat menuliskan kesimpulan.	

Kesalahan Umum dan Troubleshooting

Risiko utama pada topik ini adalah parameter tidak terdokumentasi dan keluaran tidak tervalidasi. Karena itu, notebook harus memuat pemeriksaan eksplisit dan bukan hanya mengandalkan keluaran yang tampak masuk akal. Pembaca dianjurkan menjalankan notebook dari keadaan runtime bersih, membandingkan hasil dengan batas kewajaran, dan menyimpan artefak penting di lokasi persisten.

Gejala	Tindakan perbaikan
Sel dijalankan tidak berurutan	Restart session, lalu gunakan Run all dan periksa variabel prasyarat.
Pustaka tidak ditemukan	Jalankan kembali sel %pip, periksa nama paket, lalu restart bila dependensi inti berubah.
Input tampak benar tetapi gagal	Cetak tipe, shape, satuan, muatan, spin, dan beberapa baris pertama data.

Parameter tidak terdokumentasi dan keluaran tidak tervalidasi	Tambahkan validasi sebelum proses utama dan hentikan eksekusi dengan pesan yang spesifik.
Hasil berbeda antar-runtime	Catat versi runtime/paket, random seed, parameter, dan perangkat yang digunakan.
Aturan troubleshooting: Perbaiki satu hal pada satu waktu, simpan pesan kesalahan lengkap, dan catat perubahan versi atau parameter.	

Latihan Modifikasi dan Refleksi

370. Ubah satu parameter pada contoh Membandingkan struktur awal dan akhir, jalankan ulang dari runtime bersih, dan jelaskan perubahan keluaran.
371. Tambahkan satu validasi baru yang mencegah parameter tidak terdokumentasi dan keluaran tidak tervalidasi.
372. Buat berkas ringkasan TXT atau JSON yang memuat input, status proses, waktu, dan hasil utama.
373. Jelaskan bagian mana yang merupakan fakta keluaran program dan bagian mana yang merupakan interpretasi ilmiah.
374. Susun satu kasus uji yang sengaja salah, lalu pastikan notebook menghasilkan pesan kesalahan yang informatif.

Refleksi: Tuliskan apa yang berubah setelah kode dimodifikasi, bukti apa yang menunjukkan keberhasilan, dan bagian mana yang masih memerlukan verifikasi eksternal.

Rubrik	Deskripsi
Skor 4	Kode berjalan dari runtime bersih, validasi lengkap, interpretasi tepat.
Skor 3	Kode berjalan dan hasil tersimpan, tetapi dokumentasi belum lengkap.
Skor 2	Kode hanya berjalan setelah intervensi atau state tertentu.
Skor 1	Kode belum berjalan dan diagnosis kesalahan belum sistematis.

Rangkuman Bab

Bab ini mengubah identitas kimia menjadi struktur molekul tiga dimensi. Pembaca mempelajari SMILES, muatan, spin, ETKDGv3, MMFF94, UFF, ekspor XYZ, dan visualisasi py3Dmol.

- Representasi SMILES, SDF, MOL, dan XYZ
- Muatan, jumlah elektron, multiplicity, dan spin
- Sumber struktur molekul
- Mengubah SMILES menjadi molekul tiga dimensi
- ETKDGv3, MMFF94, dan UFF
- Validasi dan ekspor koordinat
- Menampilkan struktur awal
- Memberikan label dan informasi geometri
- Membandingkan struktur awal dan akhir

Inti bab: Keberhasilan teknis harus selalu disertai validasi, dokumentasi, dan interpretasi yang proporsional.

Proyek Praktik Terpadu

Proyek terpadu menggabungkan seluruh keterampilan bab dalam satu notebook. Gunakan satu sel konfigurasi, fungsi modular, validasi, logging sederhana, penyimpanan hasil, dan bagian pembahasan.

375. Tentukan pertanyaan atau tujuan komputasi.
376. Siapkan input dan parameter.
377. Jalankan proses utama.
378. Validasi status dan nilai.
379. Buat satu visualisasi atau tabel.
380. Ekspor hasil dan tulis pembahasan.
381. Restart session dan Run all.

Produk akhir: Notebook .ipynb, satu folder hasil, arsip ZIP, dan ringkasan pengujian.

Pertanyaan Analitis

382. Mengapa kode yang menghasilkan angka belum tentu menghasilkan bukti ilmiah?
383. Bagaimana versi runtime dapat memengaruhi reproduksibilitas?
384. Apa perbedaan pemeriksaan sintaks, pemeriksaan numerik, dan validasi kimia?
385. Kapan fallback dapat diterima, dan bagaimana harus dilaporkan?
386. Apa batas interpretasi dari metode atau contoh pada bab ini?
387. Bagaimana merancang notebook agar mudah direview oleh orang lain?

Panduan Jawaban dan Rubrik

Komponen	Bobot dan indikator
Ketepatan konsep	25% - definisi, asumsi, dan satuan benar.
Kualitas kode	25% - modular, terbaca, dan menangani kesalahan.
Validasi	25% - input, proses, dan keluaran diperiksa.
Interpretasi	15% - klaim sesuai bukti dan keterbatasan.
Reproduksibilitas	10% - dapat dijalankan dari runtime bersih.

Panduan jawaban menilai alur penalaran, bukan hanya keluaran akhir. Perbedaan nilai kecil dapat diterima bila versi atau perangkat berbeda dan semua parameter dilaporkan.

Daftar Periksa Penguasaan Bab

Pernyataan	Cek
Saya dapat menjelaskan konsep utama	☐
Saya dapat menjalankan kode dari runtime bersih	☐
Saya memeriksa status dan nilai numerik	☐
Saya menyimpan hasil di lokasi persisten	☐
Saya dapat menjelaskan kesalahan umum	☐
Saya menuliskan batas interpretasi	☐
Saya mencatat versi dan waktu pengujian	☐
Tindak lanjut: Ulangi halaman yang belum dikuasai, lalu kerjakan proyek bab tanpa melihat kode contoh secara langsung.	

4

Optimasi Geometri dan Analisis Sifat Molekul d

 ΔE

BAB 4 - OPTIMASI GEOMETRI DAN ANALISIS SIFAT MOLEKUL DENGAN PYSCF

Bab ini memperkenalkan perhitungan struktur elektronik dengan PySCF. Contoh H₂ dan H₂O digunakan untuk menjelaskan HF, DFT, basis set, optimasi geometri, energi, frekuensi, orbital, dan dipol.

Arah Bab: Setiap konsep diikuti prosedur, kode, penjelasan, validasi, troubleshooting, latihan, dan refleksi. Keluaran numerik tidak boleh dipisahkan dari konteks metode dan batas penerapannya.

Komponen	Kriteria
Orientasi	Memahami konsep dan istilah.
Praktik	Menjalankan kode secara bertahap.
Validasi	Memastikan input, proses, dan keluaran sah.
Interpretasi	Menjelaskan hasil tanpa klaim berlebihan.

Capaian Pembelajaran dan Prasyarat

- Menjelaskan konsep utama bab dengan istilah yang tepat.
- Menjalankan notebook dari runtime bersih dan mendokumentasikan versi.
- Memodifikasi parameter tanpa merusak alur validasi.
- Menyimpan hasil dalam format yang dapat dibuka kembali.
- Membedakan keluaran program, interpretasi, dan keterbatasan.

Prasyarat: Kemampuan menggunakan browser, akun Google, operasi berkas dasar, dan kemauan membaca pesan kesalahan secara teliti.

Komponen	Kriteria
Pengetahuan	Konsep sebelumnya dan satuan dasar.
Keterampilan	Menyalin kode, menjalankan sel, dan membaca output.
Sikap ilmiah	Jujur terhadap status pengujian dan ketidakpastian.
Perangkat	Browser modern dan koneksi internet stabil.

Peta Konsep dan Alur Kerja

Subbab	Alur konsep
4.1 Dasar Teori Perhitungan Struktur Elektronik	Energi potensial dan optimasi geometri → Metode HF dan DFT → Basis set dan parameter numerik
4.2 Notebook Lengkap Optimasi Molekul H ₂ O	Instalasi, struktur awal, dan energi awal → Optimasi B3LYP/def2-TZVP → Frekuensi, geometri, dan visualisasi hasil
4.3 Analisis Keluaran Komputasi	Energi dan konvergensi → HOMO, LUMO, celah energi, dan momen dipol → Interpretasi kritis dan keterbatasan

Peta konsep dibaca dari kiri ke kanan dan dari input menuju keluaran. Hubungan antarkomponen tidak selalu linear: validasi dapat mengembalikan pembaca ke tahap input, sedangkan troubleshooting dapat memerlukan instalasi ulang atau penggantian metode.

Prinsip alur: Notebook yang baik memperlihatkan jalur keputusan: apa yang diperiksa, kapan proses dihentikan, kapan fallback digunakan, dan berkas apa yang disimpan.

Istilah Kunci dan Konvensi

Istilah	Konvensi penggunaan

notebook	Maknanya dibedakan dari istilah sehari-hari yang mirip.
runtime	Nilai atau statusnya dicatat dalam notebook dan laporan.
kernel	Nilai atau statusnya dicatat dalam notebook dan laporan.
sel Markdown	Istilah digunakan secara konsisten dan dijelaskan saat pertama muncul.
sel kode	Nilai atau statusnya dicatat dalam notebook dan laporan.
parameter	Nilai atau statusnya dicatat dalam notebook dan laporan.
validasi	Maknanya dibedakan dari istilah sehari-hari yang mirip.
konvergensi	Istilah digunakan secara konsisten dan dijelaskan saat pertama muncul.

Konvensi satuan: Panjang molekul dinyatakan dalam Angstrom, energi elektronik dalam Hartree, energi orbital dapat dikonversi ke eV, dan perubahan energi dapat dilaporkan dalam kJ mol⁻¹.

Asesmen Diagnostik Awal

388. Apa perbedaan notebook, runtime, dan berkas .ipynb?
389. Mengapa menjalankan sel tidak berurutan dapat menghasilkan keluaran yang menyesatkan?
390. Kapan sebuah nilai numerik boleh ditafsirkan sebagai hasil yang sah?
391. Informasi lingkungan apa yang perlu dicatat agar pekerjaan dapat diulang?
392. Bagaimana membedakan kegagalan instalasi, kegagalan input, dan kegagalan konvergensi?
393. Berkas apa yang harus disimpan sebelum runtime Colab terputus?

Cara menggunakan asesmen: Jawab sebelum memulai bab, tandai tingkat keyakinan, lalu ulangi setelah menyelesaikan proyek bab. Perubahan alasan lebih penting daripada perubahan skor saja.

4.1 Dasar Teori Perhitungan Struktur Elektronik

Dasar Teori Perhitungan Struktur Elektronik mengorganisasi tiga komponen yang saling bergantung. Pembahasan bergerak dari konsep, menuju prosedur, kemudian menghasilkan artefak yang dapat divalidasi dan disimpan.

Sub-subbab	Fokus
4.1.1 Energi potensial dan optimasi geometri	Permukaan energi potensial, gradien energi, titik stasioner, minimum lokal, dan minimum global.
4.1.2 Metode HF dan DFT	RHF, UHF, RKS, UKS, fungsi pertukaran-korelasi, B3LYP, dan keterbatasan energi DFT.
4.1.3 Basis set dan parameter numerik	STO-3G, 6-31G**, def2-SVP, def2-TZVP, grid DFT, toleransi SCF, dan density fitting.
Produk belajar: Satu notebook terstruktur, satu berkas keluaran, satu catatan validasi, dan satu interpretasi singkat yang menyatakan batas hasil.	

Rencana Praktik dan Produk Belajar

394. Pelajari Energi potensial dan optimasi geometri dan jalankan kode dasar yang menyertainya.
395. Pelajari Metode HF dan DFT dan jalankan kode dasar yang menyertainya.
396. Pelajari Basis set dan parameter numerik dan jalankan kode dasar yang menyertainya.

Komponen	Kriteria
Waktu	60-120 menit, bergantung pada komputasi.
Input	Notebook baru atau salinan notebook bab.
Produk	Kode, output, berkas hasil, dan refleksi.
Uji	Restart session lalu Run all.
Dokumentasi	Versi, parameter, status, dan waktu eksekusi.

Kriteria selesai: Semua sel dapat dijalankan berurutan, pesan kesalahan ditangani, dan hasil utama tersimpan di lokasi persisten.

4.1.1 Energi potensial dan optimasi geometri

Pembahasan Energi potensial dan optimasi geometri ditempatkan dalam konteks praktik Google Colab agar pembaca dapat menghubungkan konsep, prosedur, kode, dan interpretasi secara utuh. Permukaan energi potensial, gradien energi, titik stasioner, minimum lokal, dan minimum global. Hasil yang dituju berupa energi, geometri, orbital, dan sifat molekul.

Prinsip kerja yang digunakan adalah memisahkan input, proses, validasi, dan ekspor. Pemisahan ini membuat kesalahan lebih mudah dilacak dan mencegah hasil lama dianggap sebagai hasil dari parameter baru. Setiap langkah perlu meninggalkan bukti yang dapat dibaca kembali: nilai parameter, versi pustaka, status proses, nama berkas, serta pesan kesalahan bila terjadi kegagalan.

Tujuan operasional: Pembaca mampu menjelaskan Energi potensial dan optimasi geometri, menjalankan contoh minimal, memeriksa keberhasilan, dan menyimpan bukti hasil.

Komponen	Kriteria
Konsep	Permukaan energi potensial, gradien energi, titik stasioner, minimum lokal, dan minimum global.
Input	Parameter dan data dinyatakan eksplisit.
Proses	Langkah dapat diulang dari runtime bersih.
Output	Nilai, berkas, atau visualisasi yang dapat diperiksa.
Batas	Interpretasi mengikuti metode dan konteks.

Landasan Teori dan Batas Penerapan

Pada bagian Energi potensial dan optimasi geometri, perhatian tidak hanya diarahkan pada keberhasilan menjalankan kode, tetapi juga pada ketepatan asumsi, keterlacakan parameter, dan mutu keluaran. Permukaan energi potensial, gradien energi, titik stasioner,

minimum lokal, dan minimum global. Hasil yang dituju berupa energi, geometri, orbital, dan sifat molekul.

Prinsip kerja yang digunakan adalah memisahkan input, proses, validasi, dan ekspor. Pemisahan ini membuat kesalahan lebih mudah dilacak dan mencegah hasil lama dianggap sebagai hasil dari parameter baru. Setiap langkah perlu meninggalkan bukti yang dapat dibaca kembali: nilai parameter, versi pustaka, status proses, nama berkas, serta pesan kesalahan bila terjadi kegagalan.

Risiko utama pada topik ini adalah SCF tidak konvergen, spin keliru, atau interpretasi lintas metode. Karena itu, notebook harus memuat pemeriksaan eksplisit dan bukan hanya mengandalkan keluaran yang tampak masuk akal. Pembaca dianjurkan menjalankan notebook dari keadaan runtime bersih, membandingkan hasil dengan batas kewajaran, dan menyimpan artefak penting di lokasi persisten.

Batas penerapan: Contoh pada buku berfungsi sebagai pola kerja. Ukuran molekul, biaya komputasi, versi pustaka, dan perangkat keras dapat mengubah waktu serta strategi yang diperlukan.

Prosedur Bertahap

Energi potensial dan optimasi geometri merupakan simpul penting dalam alur kerja notebook ilmiah karena menentukan bagaimana perhitungan struktur elektronik disiapkan, dijalankan, dan diperiksa. Permukaan energi potensial, gradien energi, titik stasioner, minimum lokal, dan minimum global. Hasil yang dituju berupa energi, geometri, orbital, dan sifat molekul.

397. Tuliskan tujuan operasional Energi potensial dan optimasi geometri dan tentukan keluaran yang harus tersedia pada akhir sel.
398. Tetapkan parameter pada satu sel konfigurasi agar perubahan dapat ditelusuri dan tidak tersebar di banyak tempat.
399. Siapkan perhitungan struktur elektronik dengan input yang eksplisit; hindari nilai tersembunyi yang hanya tersimpan dalam state kernel.
400. Jalankan pemeriksaan prasyarat, termasuk keberadaan pustaka, bentuk data, tipe nilai, serta konsistensi satuan.
401. Eksekusi proses utama, tangkap status konvergensi atau keberhasilan, dan hentikan alur apabila syarat minimum tidak terpenuhi.
402. Simpan energi, geometri, orbital, dan sifat molekul dengan nama berkas informatif, kemudian tulis interpretasi yang membedakan fakta komputasi dan kesimpulan ilmiah.

Titik kontrol: Jangan melanjutkan ke interpretasi apabila validasi input atau status proses belum memenuhi kriteria.

Kode Dasar Siap Colab

Energi potensial dan optimasi geometri merupakan simpul penting dalam alur kerja notebook ilmiah karena menentukan bagaimana perhitungan struktur elektronik disiapkan, dijalankan, dan diperiksa. Permukaan energi potensial, gradien energi, titik stasioner, minimum lokal, dan minimum global. Hasil yang dituju berupa energi, geometri, orbital, dan sifat molekul.

Sel kode contoh

```
import numpy as np

jarak = np.linspace(0.40, 2.00, 17)
energi_rel = (jarak - 0.74)**2
indeks = np.argmin(energi_rel)
print("Perkiraan minimum pada:", jarak[indeks], "Å")

# Optimizer kuantum memakai gradien energi, bukan fungsi kuadrat contoh ini.
```

Cara menjalankan: Tempatkan kode dalam satu atau beberapa sel sesuai komentar. Untuk baris %pip atau perintah shell, gunakan runtime Google Colab.

Penjelasan Baris Kode Penting

Potongan kode	Fungsi
import numpy as np	Menjalankan operasi utama dan menyimpan hasil pada variabel yang jelas.
jarak = np.linspace(0.40, 2.00, 17)	Menambahkan pemeriksaan untuk mencegah hasil tidak valid diteruskan.
energi_rel = (jarak - 0.74)**2	Menambahkan pemeriksaan untuk mencegah hasil tidak valid diteruskan.
indeks = np.argmin(energi_rel)	Menambahkan pemeriksaan untuk mencegah hasil tidak valid diteruskan.
print("Perkiraan minimum pada:", jarak[indeks],	Menjalankan operasi utama dan menyimpan hasil pada variabel yang jelas.
# Optimizer kuantum memakai gradien energi, buka	Mendefinisikan input atau parameter yang akan digunakan pada langkah berikutnya.

Nama variabel dipilih agar maknanya terbaca. Hindari singkatan yang tidak dijelaskan, nilai konstan tanpa satuan, serta penggunaan variabel dari sel lama. Komentar harus menerangkan alasan atau asumsi, bukan sekadar mengulang sintaks.

Validasi dan Interpretasi Hasil

Prinsip kerja yang digunakan adalah memisahkan input, proses, validasi, dan ekspor. Pemisahan ini membuat kesalahan lebih mudah dilacak dan mencegah hasil lama dianggap sebagai hasil dari parameter baru. Setiap langkah perlu meninggalkan bukti yang dapat dibaca kembali: nilai parameter, versi pustaka, status proses, nama berkas, serta pesan kesalahan bila terjadi kegagalan.

Aspek validasi	Kriteria lulus
Input	Tipe, bentuk, satuan, muatan, spin, dan jalur berkas dinyatakan eksplisit.
Lingkungan	Versi Python dan pustaka tercatat; backend CPU/GPU terdeteksi.
Proses	Status eksekusi atau konvergensi diperiksa sebelum hasil dipakai.
Numerik	Tidak ada NaN/inf; ukuran array dan jumlah atom konsisten.
Keluaran	Energi, geometri, orbital, dan sifat molekul disimpan dan dapat dibuka kembali.
Interpretasi	Klaim tidak melampaui metode, basis set, atau data yang dihitung.
Interpretasi aman: Sebutkan metode, parameter, satuan, status konvergensi, dan keterbatasan saat menuliskan kesimpulan.	

Kesalahan Umum dan Troubleshooting

Risiko utama pada topik ini adalah SCF tidak konvergen, spin keliru, atau interpretasi lintas metode. Karena itu, notebook harus memuat pemeriksaan eksplisit dan bukan hanya mengandalkan keluaran yang tampak masuk akal. Pembaca dianjurkan menjalankan notebook dari keadaan runtime bersih, membandingkan hasil dengan batas kewajaran, dan menyimpan artefak penting di lokasi persisten.

Gejala	Tindakan perbaikan
Sel dijalankan tidak berurutan	Restart session, lalu gunakan Run all dan periksa variabel prasyarat.
Pustaka tidak ditemukan	Jalankan kembali sel %pip, periksa nama paket, lalu restart bila dependensi inti berubah.
Input tampak benar tetapi gagal	Cetak tipe, shape, satuan, muatan, spin, dan beberapa baris pertama data.
Scf tidak konvergen, spin keliru, atau interpretasi lintas metode	Tambahkan validasi sebelum proses utama dan hentikan eksekusi dengan pesan yang spesifik.
Hasil berbeda antar-runtime	Catat versi runtime/paket, random seed, parameter, dan perangkat yang digunakan.
Aturan troubleshooting: Perbaiki satu hal pada satu waktu, simpan pesan kesalahan lengkap, dan catat perubahan versi atau parameter.	

Latihan Modifikasi dan Refleksi

403. Ubah satu parameter pada contoh Energi potensial dan optimasi geometri, jalankan ulang dari runtime bersih, dan jelaskan perubahan keluaran.
404. Tambahkan satu validasi baru yang mencegah SCF tidak konvergen, spin keliru, atau interpretasi lintas metode.
405. Buat berkas ringkasan TXT atau JSON yang memuat input, status proses, waktu, dan hasil utama.
406. Jelaskan bagian mana yang merupakan fakta keluaran program dan bagian mana yang merupakan interpretasi ilmiah.
407. Susun satu kasus uji yang sengaja salah, lalu pastikan notebook menghasilkan pesan kesalahan yang informatif.

Refleksi: Tuliskan apa yang berubah setelah kode dimodifikasi, bukti apa yang menunjukkan keberhasilan, dan bagian mana yang masih memerlukan verifikasi eksternal.

Rubrik	Deskripsi
Skor 4	Kode berjalan dari runtime bersih, validasi lengkap, interpretasi tepat.

Skor 3	Kode berjalan dan hasil tersimpan, tetapi dokumentasi belum lengkap.
Skor 2	Kode hanya berjalan setelah intervensi atau state tertentu.
Skor 1	Kode belum berjalan dan diagnosis kesalahan belum sistematis.

4.1.2 Metode HF dan DFT

Pada bagian Metode HF dan DFT, perhatian tidak hanya diarahkan pada keberhasilan menjalankan kode, tetapi juga pada ketepatan asumsi, keterlacakan parameter, dan mutu keluaran. RHF, UHF, RKS, UKS, fungsi pertukaran-korelasi, B3LYP, dan keterbatasan energi DFT. Hasil yang dituju berupa energi, geometri, orbital, dan sifat molekul.

Prinsip kerja yang digunakan adalah memisahkan input, proses, validasi, dan ekspor. Pemisahan ini membuat kesalahan lebih mudah dilacak dan mencegah hasil lama dianggap sebagai hasil dari parameter baru. Setiap langkah perlu meninggalkan bukti yang dapat dibaca kembali: nilai parameter, versi pustaka, status proses, nama berkas, serta pesan kesalahan bila terjadi kegagalan.

Tujuan operasional: Pembaca mampu menjelaskan Metode HF dan DFT, menjalankan contoh minimal, memeriksa keberhasilan, dan menyimpan bukti hasil.

Komponen	Kriteria
Konsep	RHF, UHF, RKS, UKS, fungsi pertukaran-korelasi, B3LYP, dan keterbatasan energi DFT.
Input	Parameter dan data dinyatakan eksplisit.
Proses	Langkah dapat diulang dari runtime bersih.
Output	Nilai, berkas, atau visualisasi yang dapat diperiksa.
Batas	Interpretasi mengikuti metode dan konteks.

Landasan Teori dan Batas Penerapan

Pembahasan Metode HF dan DFT ditempatkan dalam konteks praktik Google Colab agar pembaca dapat menghubungkan konsep, prosedur, kode, dan interpretasi secara utuh. RHF, UHF, RKS, UKS, fungsi pertukaran-korelasi, B3LYP, dan keterbatasan energi DFT. Hasil yang dituju berupa energi, geometri, orbital, dan sifat molekul.

Prinsip kerja yang digunakan adalah memisahkan input, proses, validasi, dan ekspor. Pemisahan ini membuat kesalahan lebih mudah dilacak dan mencegah hasil lama dianggap sebagai hasil dari parameter baru. Setiap langkah perlu meninggalkan bukti yang dapat dibaca kembali: nilai parameter, versi pustaka, status proses, nama berkas, serta pesan kesalahan bila terjadi kegagalan.

Risiko utama pada topik ini adalah SCF tidak konvergen, spin keliru, atau interpretasi lintas metode. Karena itu, notebook harus memuat pemeriksaan eksplisit dan bukan hanya mengandalkan keluaran yang tampak masuk akal. Pembaca dianjurkan menjalankan notebook dari keadaan runtime bersih, membandingkan hasil dengan batas kewajaran, dan menyimpan artefak penting di lokasi persisten.

Batas penerapan: Contoh pada buku berfungsi sebagai pola kerja. Ukuran molekul, biaya komputasi, versi pustaka, dan perangkat keras dapat mengubah waktu serta strategi yang diperlukan.

Prosedur Bertahap

Pembahasan Metode HF dan DFT ditempatkan dalam konteks praktik Google Colab agar pembaca dapat menghubungkan konsep, prosedur, kode, dan interpretasi secara utuh. RHF, UHF, RKS, UKS, fungsi pertukaran-korelasi, B3LYP, dan keterbatasan energi DFT. Hasil yang dituju berupa energi, geometri, orbital, dan sifat molekul.

408. Tuliskan tujuan operasional Metode HF dan DFT dan tentukan keluaran yang harus tersedia pada akhir sel.
409. Tetapkan parameter pada satu sel konfigurasi agar perubahan dapat ditelusuri dan tidak tersebar di banyak tempat.
410. Siapkan perhitungan struktur elektronik dengan input yang eksplisit; hindari nilai tersembunyi yang hanya tersimpan dalam state kernel.
411. Jalankan pemeriksaan prasyarat, termasuk keberadaan pustaka, bentuk data, tipe nilai, serta konsistensi satuan.
412. Eksekusi proses utama, tangkap status konvergensi atau keberhasilan, dan hentikan alur apabila syarat minimum tidak terpenuhi.
413. Simpan energi, geometri, orbital, dan sifat molekul dengan nama berkas informatif, kemudian tulis interpretasi yang membedakan fakta komputasi dan kesimpulan ilmiah.

Titik kontrol: Jangan melanjutkan ke interpretasi apabila validasi input atau status proses belum memenuhi kriteria.

Kode Dasar Siap Colab

Pada bagian Metode HF dan DFT, perhatian tidak hanya diarahkan pada keberhasilan menjalankan kode, tetapi juga pada ketepatan asumsi, keterlacakan parameter, dan mutu keluaran. RHF, UHF, RKS, UKS, fungsi pertukaran-korelasi, B3LYP, dan keterbatasan energi DFT. Hasil yang dituju berupa energi, geometri, orbital, dan sifat molekul.

Sel kode contoh

```
from pyscf import gto, scf, dft

mol = gto.M(atom="H 0 0 0; H 0 0 0.74", basis="sto-3g", spin=0)
hf = scf.RHF(mol); e_hf = hf.kernel()
ks = dft.RKS(mol); ks.xc = "b3lyp"; e_dft = ks.kernel()
print("HF :", e_hf, hf.converged)
print("DFT :", e_dft, ks.converged)
```

Cara menjalankan: Tempatkan kode dalam satu atau beberapa sel sesuai komentar. Untuk baris %pip atau perintah shell, gunakan runtime Google Colab.

Penjelasan Baris Kode Penting

Potongan kode	Fungsi
<code>from pyscf import gto, scf, dft</code>	Mendefinisikan input atau parameter yang akan digunakan pada langkah berikutnya.
<code>mol = gto.M(atom="H 0 0 0; H 0 0 0.74", basis="s</code>	Menambahkan pemeriksaan untuk mencegah hasil tidak valid diteruskan.
<code>hf = scf.RHF(mol); e_hf = hf.kernel()</code>	Menambahkan pemeriksaan untuk mencegah hasil tidak valid diteruskan.
<code>ks = dft.RKS(mol); ks.xc = "b3lyp"; e_dft = ks.k</code>	Mencetak status agar pengguna tidak hanya bergantung pada keluaran visual.
<code>print("HF :", e_hf, hf.converged)</code>	Menjalankan operasi utama dan menyimpan hasil pada variabel yang jelas.

```
print("DFT :", e_dft, ks.converged)
```

Menambahkan pemeriksaan untuk mencegah hasil tidak valid diteruskan.

Nama variabel dipilih agar maknanya terbaca. Hindari singkatan yang tidak dijelaskan, nilai konstan tanpa satuan, serta penggunaan variabel dari sel lama. Komentar harus menerangkan alasan atau asumsi, bukan sekadar mengulang sintaks.

Validasi dan Interpretasi Hasil

Prinsip kerja yang digunakan adalah memisahkan input, proses, validasi, dan ekspor. Pemisahan ini membuat kesalahan lebih mudah dilacak dan mencegah hasil lama dianggap sebagai hasil dari parameter baru. Setiap langkah perlu meninggalkan bukti yang dapat dibaca kembali: nilai parameter, versi pustaka, status proses, nama berkas, serta pesan kesalahan bila terjadi kegagalan.

Aspek validasi	Kriteria lulus
Input	Tipe, bentuk, satuan, muatan, spin, dan jalur berkas dinyatakan eksplisit.
Lingkungan	Versi Python dan pustaka tercatat; backend CPU/GPU terdeteksi.
Proses	Status eksekusi atau konvergensi diperiksa sebelum hasil dipakai.
Numerik	Tidak ada NaN/inf; ukuran array dan jumlah atom konsisten.
Keluaran	Energi, geometri, orbital, dan sifat molekul disimpan dan dapat dibuka kembali.
Interpretasi	Klaim tidak melampaui metode, basis set, atau data yang dihitung.
Interpretasi aman: Sebutkan metode, parameter, satuan, status konvergensi, dan keterbatasan saat menuliskan kesimpulan.	

Kesalahan Umum dan Troubleshooting

Risiko utama pada topik ini adalah SCF tidak konvergen, spin keliru, atau interpretasi lintas metode. Karena itu, notebook harus memuat pemeriksaan eksplisit dan bukan hanya mengandalkan keluaran yang tampak masuk akal. Pembaca dianjurkan menjalankan

notebook dari keadaan runtime bersih, membandingkan hasil dengan batas kewajaran, dan menyimpan artefak penting di lokasi persisten.

Gejala	Tindakan perbaikan
Sel dijalankan tidak berurutan	Restart session, lalu gunakan Run all dan periksa variabel prasyarat.
Pustaka tidak ditemukan	Jalankan kembali sel %pip, periksa nama paket, lalu restart bila dependensi inti berubah.
Input tampak benar tetapi gagal	Cetak tipe, shape, satuan, muatan, spin, dan beberapa baris pertama data.
Scf tidak konvergen, spin keliru, atau interpretasi lintas metode	Tambahkan validasi sebelum proses utama dan hentikan eksekusi dengan pesan yang spesifik.
Hasil berbeda antar-runtime	Catat versi runtime/paket, random seed, parameter, dan perangkat yang digunakan.
Aturan troubleshooting: Perbaiki satu hal pada satu waktu, simpan pesan kesalahan lengkap, dan catat perubahan versi atau parameter.	

Latihan Modifikasi dan Refleksi

414. Ubah satu parameter pada contoh Metode HF dan DFT, jalankan ulang dari runtime bersih, dan jelaskan perubahan keluaran.
415. Tambahkan satu validasi baru yang mencegah SCF tidak konvergen, spin keliru, atau interpretasi lintas metode.
416. Buat berkas ringkasan TXT atau JSON yang memuat input, status proses, waktu, dan hasil utama.
417. Jelaskan bagian mana yang merupakan fakta keluaran program dan bagian mana yang merupakan interpretasi ilmiah.
418. Susun satu kasus uji yang sengaja salah, lalu pastikan notebook menghasilkan pesan kesalahan yang informatif.

Refleksi: Tuliskan apa yang berubah setelah kode dimodifikasi, bukti apa yang menunjukkan keberhasilan, dan bagian mana yang masih memerlukan verifikasi eksternal.

Rubrik

Deskripsi

Skor 4	Kode berjalan dari runtime bersih, validasi lengkap, interpretasi tepat.
Skor 3	Kode berjalan dan hasil tersimpan, tetapi dokumentasi belum lengkap.
Skor 2	Kode hanya berjalan setelah intervensi atau state tertentu.
Skor 1	Kode belum berjalan dan diagnosis kesalahan belum sistematis.

4.1.3 Basis set dan parameter numerik

Pembahasan Basis set dan parameter numerik ditempatkan dalam konteks praktik Google Colab agar pembaca dapat menghubungkan konsep, prosedur, kode, dan interpretasi secara utuh. STO-3G, 6-31G**, def2-SVP, def2-TZVP, grid DFT, toleransi SCF, dan density fitting. Hasil yang dituju berupa energi, geometri, orbital, dan sifat molekul.

Prinsip kerja yang digunakan adalah memisahkan input, proses, validasi, dan ekspor. Pemisahan ini membuat kesalahan lebih mudah dilacak dan mencegah hasil lama dianggap sebagai hasil dari parameter baru. Setiap langkah perlu meninggalkan bukti yang dapat dibaca kembali: nilai parameter, versi pustaka, status proses, nama berkas, serta pesan kesalahan bila terjadi kegagalan.

Tujuan operasional: Pembaca mampu menjelaskan Basis set dan parameter numerik, menjalankan contoh minimal, memeriksa keberhasilan, dan menyimpan bukti hasil.

Komponen	Kriteria
Konsep	STO-3G, 6-31G**, def2-SVP, def2-TZVP, grid DFT, toleransi SCF, dan density fitting.
Input	Parameter dan data dinyatakan eksplisit.
Proses	Langkah dapat diulang dari runtime bersih.
Output	Nilai, berkas, atau visualisasi yang dapat diperiksa.
Batas	Interpretasi mengikuti metode dan konteks.

Landasan Teori dan Batas Penerapan

Pada bagian Basis set dan parameter numerik, perhatian tidak hanya diarahkan pada keberhasilan menjalankan kode, tetapi juga pada ketepatan asumsi, keterlacakan parameter, dan mutu keluaran. STO-3G, 6-31G**, def2-SVP, def2-TZVP, grid DFT, toleransi SCF, dan density fitting. Hasil yang dituju berupa energi, geometri, orbital, dan sifat molekul.

Prinsip kerja yang digunakan adalah memisahkan input, proses, validasi, dan ekspor. Pemisahan ini membuat kesalahan lebih mudah dilacak dan mencegah hasil lama dianggap sebagai hasil dari parameter baru. Setiap langkah perlu meninggalkan bukti yang dapat dibaca kembali: nilai parameter, versi pustaka, status proses, nama berkas, serta pesan kesalahan bila terjadi kegagalan.

Risiko utama pada topik ini adalah SCF tidak konvergen, spin keliru, atau interpretasi lintas metode. Karena itu, notebook harus memuat pemeriksaan eksplisit dan bukan hanya mengandalkan keluaran yang tampak masuk akal. Pembaca dianjurkan menjalankan notebook dari keadaan runtime bersih, membandingkan hasil dengan batas kewajaran, dan menyimpan artefak penting di lokasi persisten.

Batas penerapan: Contoh pada buku berfungsi sebagai pola kerja. Ukuran molekul, biaya komputasi, versi pustaka, dan perangkat keras dapat mengubah waktu serta strategi yang diperlukan.

Prosedur Bertahap

Pembahasan Basis set dan parameter numerik ditempatkan dalam konteks praktik Google Colab agar pembaca dapat menghubungkan konsep, prosedur, kode, dan interpretasi secara utuh. STO-3G, 6-31G**, def2-SVP, def2-TZVP, grid DFT, toleransi SCF, dan density fitting. Hasil yang dituju berupa energi, geometri, orbital, dan sifat molekul.

419. Tuliskan tujuan operasional Basis set dan parameter numerik dan tentukan keluaran yang harus tersedia pada akhir sel.
420. Tetapkan parameter pada satu sel konfigurasi agar perubahan dapat ditelusuri dan tidak tersebar di banyak tempat.
421. Siapkan perhitungan struktur elektronik dengan input yang eksplisit; hindari nilai tersembunyi yang hanya tersimpan dalam state kernel.
422. Jalankan pemeriksaan prasyarat, termasuk keberadaan pustaka, bentuk data, tipe nilai, serta konsistensi satuan.
423. Eksekusi proses utama, tangkap status konvergensi atau keberhasilan, dan hentikan alur apabila syarat minimum tidak terpenuhi.
424. Simpan energi, geometri, orbital, dan sifat molekul dengan nama berkas informatif, kemudian tulis interpretasi yang membedakan fakta komputasi dan kesimpulan ilmiah.

Titik kontrol: Jangan melanjutkan ke interpretasi apabila validasi input atau status proses belum memenuhi kriteria.

Kode Dasar Siap Colab

Pembahasan Basis set dan parameter numerik ditempatkan dalam konteks praktik Google Colab agar pembaca dapat menghubungkan konsep, prosedur, kode, dan interpretasi secara utuh. STO-3G, 6-31G**, def2-SVP, def2-TZVP, grid DFT, toleransi SCF, dan density fitting. Hasil yang dituju berupa energi, geometri, orbital, dan sifat molekul.

Sel kode contoh

```
from pyscf import gto, dft

for basis in ["sto-3g", "6-31g**", "def2-svp", "def2-tzvp"]:
    mol = gto.M(atom="H 0 0 0; H 0 0 0.74", basis=basis, spin=0, verbose=0)
    mf = dft.RKS(mol).density_fit()
    mf.xc = "b3lyp"; mf.grids.level = 4
    mf.conv_tol = 1e-9; mf.max_cycle = 200
    print(basis, mf.kernel(), mf.converged)
```

Cara menjalankan: Tempatkan kode dalam satu atau beberapa sel sesuai komentar. Untuk baris %pip atau perintah shell, gunakan runtime Google Colab.

Penjelasan Baris Kode Penting

Potongan kode	Fungsi
<code>from pyscf import gto, dft</code>	Mendefinisikan input atau parameter yang akan digunakan pada langkah berikutnya.
<code>for basis in ["sto-3g", "6-31g**", "def2-svp", "]</code>	Mencetak status agar pengguna tidak hanya bergantung pada keluaran visual.
<code>mol = gto.M(atom="H 0 0 0; H 0 0 0.74", basi</code>	Mendefinisikan input atau parameter yang akan digunakan pada langkah berikutnya.
<code>mf = dft.RKS(mol).density_fit()</code>	Mendefinisikan input atau parameter yang akan digunakan pada langkah berikutnya.
<code>mf.xc = "b3lyp"; mf.grids.level = 4</code>	Mencetak status agar pengguna tidak hanya bergantung pada keluaran visual.

<code>mf.conv_tol = 1e-9; mf.max_cycle = 200</code>	Menambahkan pemeriksaan untuk mencegah hasil tidak valid diteruskan.
<code>print(basis, mf.kernel(), mf.converged)</code>	Menjalankan operasi utama dan menyimpan hasil pada variabel yang jelas.

Nama variabel dipilih agar maknanya terbaca. Hindari singkatan yang tidak dijelaskan, nilai konstan tanpa satuan, serta penggunaan variabel dari sel lama. Komentar harus menerangkan alasan atau asumsi, bukan sekadar mengulang sintaks.

Validasi dan Interpretasi Hasil

Prinsip kerja yang digunakan adalah memisahkan input, proses, validasi, dan ekspor. Pemisahan ini membuat kesalahan lebih mudah dilacak dan mencegah hasil lama dianggap sebagai hasil dari parameter baru. Setiap langkah perlu meninggalkan bukti yang dapat dibaca kembali: nilai parameter, versi pustaka, status proses, nama berkas, serta pesan kesalahan bila terjadi kegagalan.

Aspek validasi	Kriteria lulus
Input	Tipe, bentuk, satuan, muatan, spin, dan jalur berkas dinyatakan eksplisit.
Lingkungan	Versi Python dan pustaka tercatat; backend CPU/GPU terdeteksi.
Proses	Status eksekusi atau konvergensi diperiksa sebelum hasil dipakai.
Numerik	Tidak ada NaN/inf; ukuran array dan jumlah atom konsisten.
Keluaran	Energi, geometri, orbital, dan sifat molekul disimpan dan dapat dibuka kembali.
Interpretasi	Klaim tidak melampaui metode, basis set, atau data yang dihitung.
Interpretasi aman: Sebutkan metode, parameter, satuan, status konvergensi, dan keterbatasan saat menuliskan kesimpulan.	

Kesalahan Umum dan Troubleshooting

Risiko utama pada topik ini adalah SCF tidak konvergen, spin keliru, atau interpretasi lintas metode. Karena itu, notebook harus memuat pemeriksaan eksplisit dan bukan hanya mengandalkan keluaran yang tampak masuk akal. Pembaca dianjurkan menjalankan notebook dari keadaan runtime bersih, membandingkan hasil dengan batas kewajaran, dan menyimpan artefak penting di lokasi persisten.

Gejala	Tindakan perbaikan
Sel dijalankan tidak berurutan	Restart session, lalu gunakan Run all dan periksa variabel prasyarat.
Pustaka tidak ditemukan	Jalankan kembali sel %pip, periksa nama paket, lalu restart bila dependensi inti berubah.
Input tampak benar tetapi gagal	Cetak tipe, shape, satuan, muatan, spin, dan beberapa baris pertama data.
Scf tidak konvergen, spin keliru, atau interpretasi lintas metode	Tambahkan validasi sebelum proses utama dan hentikan eksekusi dengan pesan yang spesifik.
Hasil berbeda antar-runtime	Catat versi runtime/paket, random seed, parameter, dan perangkat yang digunakan.
Aturan troubleshooting: Perbaiki satu hal pada satu waktu, simpan pesan kesalahan lengkap, dan catat perubahan versi atau parameter.	

Latihan Modifikasi dan Refleksi

425. Ubah satu parameter pada contoh Basis set dan parameter numerik, jalankan ulang dari runtime bersih, dan jelaskan perubahan keluaran.
426. Tambahkan satu validasi baru yang mencegah SCF tidak konvergen, spin keliru, atau interpretasi lintas metode.
427. Buat berkas ringkasan TXT atau JSON yang memuat input, status proses, waktu, dan hasil utama.
428. Jelaskan bagian mana yang merupakan fakta keluaran program dan bagian mana yang merupakan interpretasi ilmiah.
429. Susun satu kasus uji yang sengaja salah, lalu pastikan notebook menghasilkan pesan kesalahan yang informatif.

Refleksi: Tuliskan apa yang berubah setelah kode dimodifikasi, bukti apa yang menunjukkan keberhasilan, dan bagian mana yang masih memerlukan verifikasi eksternal.

Rubrik	Deskripsi
Skor 4	Kode berjalan dari runtime bersih, validasi lengkap, interpretasi tepat.
Skor 3	Kode berjalan dan hasil tersimpan, tetapi dokumentasi belum lengkap.
Skor 2	Kode hanya berjalan setelah intervensi atau state tertentu.
Skor 1	Kode belum berjalan dan diagnosis kesalahan belum sistematis.

4.2 Notebook Lengkap Optimasi Molekul H₂O

Notebook Lengkap Optimasi Molekul H₂O mengorganisasi tiga komponen yang saling bergantung. Pembahasan bergerak dari konsep, menuju prosedur, kemudian menghasilkan artefak yang dapat divalidasi dan disimpan.

Sub-subbab	Fokus
4.2.1 Instalasi, struktur awal, dan energi awal	Pemasangan PySCF, geomeTRIC, dan py3Dmol; pembuatan molekul; pemeriksaan elektron dan spin.
4.2.2 Optimasi B3LYP/def2-TZVP	Perhitungan SCF awal, optimasi geometri, pembuatan metode baru pada geometri optimum, dan energi final.
4.2.3 Frekuensi, geometri, dan visualisasi hasil	Panjang ikatan O-H, sudut H-O-H, frekuensi vibrasi, pemeriksaan frekuensi imajiner, struktur XYZ, dan tampilan 3D.
Produk belajar: Satu notebook terstruktur, satu berkas keluaran, satu catatan validasi, dan satu interpretasi singkat yang menyatakan batas hasil.	

Rencana Praktik dan Produk Belajar

430. Pelajari Instalasi, struktur awal, dan energi awal dan jalankan kode dasar yang menyertainya.
431. Pelajari Optimasi B3LYP/def2-TZVP dan jalankan kode dasar yang menyertainya.
432. Pelajari Frekuensi, geometri, dan visualisasi hasil dan jalankan kode dasar yang menyertainya.

Komponen	Kriteria
Waktu	60-120 menit, bergantung pada komputasi.
Input	Notebook baru atau salinan notebook bab.
Produk	Kode, output, berkas hasil, dan refleksi.
Uji	Restart session lalu Run all.
Dokumentasi	Versi, parameter, status, dan waktu eksekusi.
Kriteria selesai: Semua sel dapat dijalankan berurutan, pesan kesalahan ditangani, dan hasil utama tersimpan di lokasi persisten.	

4.2.1 Instalasi, struktur awal, dan energi awal

Pembahasan Instalasi, struktur awal, dan energi awal ditempatkan dalam konteks praktik Google Colab agar pembaca dapat menghubungkan konsep, prosedur, kode, dan interpretasi secara utuh. Pemasangan PySCF, geomeTRIC, dan py3Dmol; pembuatan molekul; pemeriksaan elektron dan spin. Hasil yang dituju berupa energi, geometri, orbital, dan sifat molekul.

Prinsip kerja yang digunakan adalah memisahkan input, proses, validasi, dan ekspor. Pemisahan ini membuat kesalahan lebih mudah dilacak dan mencegah hasil lama dianggap sebagai hasil dari parameter baru. Setiap langkah perlu meninggalkan bukti yang dapat dibaca kembali: nilai parameter, versi pustaka, status proses, nama berkas, serta pesan kesalahan bila terjadi kegagalan.

Tujuan operasional: Pembaca mampu menjelaskan Instalasi, struktur awal, dan energi awal, menjalankan contoh minimal, memeriksa keberhasilan, dan menyimpan bukti hasil.

Komponen**Kriteria**

Konsep	Pemasangan PySCF, geomeTRIC, dan py3Dmol; pembuatan molekul; pemeriksaan elektron dan spin.
Input	Parameter dan data dinyatakan eksplisit.
Proses	Langkah dapat diulang dari runtime bersih.
Output	Nilai, berkas, atau visualisasi yang dapat diperiksa.
Batas	Interpretasi mengikuti metode dan konteks.

Landasan Teori dan Batas Penerapan

Pada bagian Instalasi, struktur awal, dan energi awal, perhatian tidak hanya diarahkan pada keberhasilan menjalankan kode, tetapi juga pada ketepatan asumsi, keterlacakan parameter, dan mutu keluaran. Pemasangan PySCF, geomeTRIC, dan py3Dmol; pembuatan molekul; pemeriksaan elektron dan spin. Hasil yang dituju berupa energi, geometri, orbital, dan sifat molekul.

Prinsip kerja yang digunakan adalah memisahkan input, proses, validasi, dan ekspor. Pemisahan ini membuat kesalahan lebih mudah dilacak dan mencegah hasil lama dianggap sebagai hasil dari parameter baru. Setiap langkah perlu meninggalkan bukti yang dapat dibaca kembali: nilai parameter, versi pustaka, status proses, nama berkas, serta pesan kesalahan bila terjadi kegagalan.

Risiko utama pada topik ini adalah SCF tidak konvergen, spin keliru, atau interpretasi lintas metode. Karena itu, notebook harus memuat pemeriksaan eksplisit dan bukan hanya mengandalkan keluaran yang tampak masuk akal. Pembaca dianjurkan menjalankan notebook dari keadaan runtime bersih, membandingkan hasil dengan batas kewajaran, dan menyimpan artefak penting di lokasi persisten.

Batas penerapan: Contoh pada buku berfungsi sebagai pola kerja. Ukuran molekul, biaya komputasi, versi pustaka, dan perangkat keras dapat mengubah waktu serta strategi yang diperlukan.

Prosedur Bertahap

Instalasi, struktur awal, dan energi awal merupakan simpul penting dalam alur kerja notebook ilmiah karena menentukan bagaimana perhitungan struktur elektronik disiapkan, dijalankan, dan diperiksa. Pemasangan PySCF, geomeTRIC, dan py3Dmol;

pembuatan molekul; pemeriksaan elektron dan spin. Hasil yang dituju berupa energi, geometri, orbital, dan sifat molekul.

433. Tuliskan tujuan operasional Instalasi, struktur awal, dan energi awal dan tentukan keluaran yang harus tersedia pada akhir sel.
434. Tetapkan parameter pada satu sel konfigurasi agar perubahan dapat ditelusuri dan tidak tersebar di banyak tempat.
435. Siapkan perhitungan struktur elektronik dengan input yang eksplisit; hindari nilai tersembunyi yang hanya tersimpan dalam state kernel.
436. Jalankan pemeriksaan prasyarat, termasuk keberadaan pustaka, bentuk data, tipe nilai, serta konsistensi satuan.
437. Eksekusi proses utama, tangkap status konvergensi atau keberhasilan, dan hentikan alur apabila syarat minimum tidak terpenuhi.
438. Simpan energi, geometri, orbital, dan sifat molekul dengan nama berkas informatif, kemudian tulis interpretasi yang membedakan fakta komputasi dan kesimpulan ilmiah.

Titik kontrol: Jangan melanjutkan ke interpretasi apabila validasi input atau status proses belum memenuhi kriteria.

Kode Dasar Siap Colab

Pada bagian Instalasi, struktur awal, dan energi awal, perhatian tidak hanya diarahkan pada keberhasilan menjalankan kode, tetapi juga pada ketepatan asumsi, keterlacakan parameter, dan mutu keluaran. Pemasangan PySCF, geomeTRIC, dan py3Dmol; pembuatan molekul; pemeriksaan elektron dan spin. Hasil yang dituju berupa energi, geometri, orbital, dan sifat molekul.

Sel kode contoh

```
%pip -q install --prefer-binary pyscf geometric py3Dmol

from pyscf import gto, dft

mol_h2o = gto.M(
    atom="O 0 0 0; H 0 0.757 0.587; H 0 -0.757 0.587",
    unit="Angstrom", basis="def2-tzvp", charge=0, spin=0
)
mf0 = dft.RKS(mol_h2o); mf0.xc = "b3lyp"
e0 = mf0.kernel()
print("Energi awal:", e0, "konvergen:", mf0.converged)
```

Cara menjalankan: Tempatkan kode dalam satu atau beberapa sel sesuai komentar. Untuk baris %pip atau perintah shell, gunakan runtime Google Colab.

Penjelasan Baris Kode Penting

Potongan kode	Fungsi
<code>%pip -q install --prefer-binary pyscf geometric</code>	Menambahkan pemeriksaan untuk mencegah hasil tidak valid diteruskan.
<code>from pyscf import gto, dft</code>	Mendefinisikan input atau parameter yang akan digunakan pada langkah berikutnya.
<code>mol_h2o = gto.M(</code>	Menambahkan pemeriksaan untuk mencegah hasil tidak valid diteruskan.
<code>atom="O 0 0 0; H 0 0.757 0.587; H 0 - 0.757 0</code>	Menambahkan pemeriksaan untuk mencegah hasil tidak valid diteruskan.
<code>unit="Angstrom", basis="def2-tzvp", charge=0</code>	Menjalankan operasi utama dan menyimpan hasil pada variabel yang jelas.
<code>)</code>	Mencetak status agar pengguna tidak hanya bergantung pada keluaran visual.
<code>mf0 = dft.RKS(mol_h2o); mf0.xc = "b3lyp"</code>	Menjalankan operasi utama dan menyimpan hasil pada variabel yang jelas.
<code>e0 = mf0.kernel()</code>	Mencetak status agar pengguna tidak hanya bergantung pada keluaran visual.

Nama variabel dipilih agar maknanya terbaca. Hindari singkatan yang tidak dijelaskan, nilai konstan tanpa satuan, serta penggunaan variabel dari sel lama. Komentar harus menerangkan alasan atau asumsi, bukan sekadar mengulang sintaks.

Validasi dan Interpretasi Hasil

Prinsip kerja yang digunakan adalah memisahkan input, proses, validasi, dan ekspor. Pemisahan ini membuat kesalahan lebih mudah dilacak dan mencegah hasil lama dianggap sebagai hasil dari parameter baru. Setiap langkah perlu meninggalkan bukti yang dapat dibaca kembali: nilai parameter, versi pustaka, status proses, nama berkas, serta pesan kesalahan bila terjadi kegagalan.

Aspek validasi**Kriteria lulus**

Input	Tipe, bentuk, satuan, muatan, spin, dan jalur berkas dinyatakan eksplisit.
Lingkungan	Versi Python dan pustaka tercatat; backend CPU/GPU terdeteksi.
Proses	Status eksekusi atau konvergensi diperiksa sebelum hasil dipakai.
Numerik	Tidak ada NaN/inf; ukuran array dan jumlah atom konsisten.
Keluaran	Energi, geometri, orbital, dan sifat molekul disimpan dan dapat dibuka kembali.
Interpretasi	Klaim tidak melampaui metode, basis set, atau data yang dihitung.
Interpretasi aman: Sebutkan metode, parameter, satuan, status konvergensi, dan keterbatasan saat menuliskan kesimpulan.	

Kesalahan Umum dan Troubleshooting

Risiko utama pada topik ini adalah SCF tidak konvergen, spin keliru, atau interpretasi lintas metode. Karena itu, notebook harus memuat pemeriksaan eksplisit dan bukan hanya mengandalkan keluaran yang tampak masuk akal. Pembaca dianjurkan menjalankan notebook dari keadaan runtime bersih, membandingkan hasil dengan batas kewajaran, dan menyimpan artefak penting di lokasi persisten.

Gejala	Tindakan perbaikan
Sel dijalankan tidak berurutan	Restart session, lalu gunakan Run all dan periksa variabel prasyarat.
Pustaka tidak ditemukan	Jalankan kembali sel %pip, periksa nama paket, lalu restart bila dependensi inti berubah.
Input tampak benar tetapi gagal	Cetak tipe, shape, satuan, muatan, spin, dan beberapa baris pertama data.

Scf tidak konvergen, spin keliru, atau interpretasi lintas metode	Tambahkan validasi sebelum proses utama dan hentikan eksekusi dengan pesan yang spesifik.
Hasil berbeda antar-runtime	Catat versi runtime/paket, random seed, parameter, dan perangkat yang digunakan.
Aturan troubleshooting: Perbaiki satu hal pada satu waktu, simpan pesan kesalahan lengkap, dan catat perubahan versi atau parameter.	

Latihan Modifikasi dan Refleksi

439. Ubah satu parameter pada contoh Instalasi, struktur awal, dan energi awal, jalankan ulang dari runtime bersih, dan jelaskan perubahan keluaran.
440. Tambahkan satu validasi baru yang mencegah SCF tidak konvergen, spin keliru, atau interpretasi lintas metode.
441. Buat berkas ringkasan TXT atau JSON yang memuat input, status proses, waktu, dan hasil utama.
442. Jelaskan bagian mana yang merupakan fakta keluaran program dan bagian mana yang merupakan interpretasi ilmiah.
443. Susun satu kasus uji yang sengaja salah, lalu pastikan notebook menghasilkan pesan kesalahan yang informatif.

Refleksi: Tuliskan apa yang berubah setelah kode dimodifikasi, bukti apa yang menunjukkan keberhasilan, dan bagian mana yang masih memerlukan verifikasi eksternal.

Rubrik	Deskripsi
Skor 4	Kode berjalan dari runtime bersih, validasi lengkap, interpretasi tepat.
Skor 3	Kode berjalan dan hasil tersimpan, tetapi dokumentasi belum lengkap.
Skor 2	Kode hanya berjalan setelah intervensi atau state tertentu.
Skor 1	Kode belum berjalan dan diagnosis kesalahan belum sistematis.

4.2.2 Optimasi B3LYP/def2-TZVP

Pembahasan Optimasi B3LYP/def2-TZVP ditempatkan dalam konteks praktik Google Colab agar pembaca dapat menghubungkan konsep, prosedur, kode, dan interpretasi secara utuh. Perhitungan SCF awal, optimasi geometri, pembuatan metode baru pada geometri optimum, dan energi final. Hasil yang dituju berupa energi, geometri, orbital, dan sifat molekul.

Prinsip kerja yang digunakan adalah memisahkan input, proses, validasi, dan ekspor. Pemisahan ini membuat kesalahan lebih mudah dilacak dan mencegah hasil lama dianggap sebagai hasil dari parameter baru. Setiap langkah perlu meninggalkan bukti yang dapat dibaca kembali: nilai parameter, versi pustaka, status proses, nama berkas, serta pesan kesalahan bila terjadi kegagalan.

Tujuan operasional: Pembaca mampu menjelaskan Optimasi B3LYP/def2-TZVP, menjalankan contoh minimal, memeriksa keberhasilan, dan menyimpan bukti hasil.

Komponen	Kriteria
Konsep	Perhitungan SCF awal, optimasi geometri, pembuatan metode baru pada geometri optimum, dan energi final.
Input	Parameter dan data dinyatakan eksplisit.
Proses	Langkah dapat diulang dari runtime bersih.
Output	Nilai, berkas, atau visualisasi yang dapat diperiksa.
Batas	Interpretasi mengikuti metode dan konteks.

Landasan Teori dan Batas Penerapan

Pembahasan Optimasi B3LYP/def2-TZVP ditempatkan dalam konteks praktik Google Colab agar pembaca dapat menghubungkan konsep, prosedur, kode, dan interpretasi secara utuh. Perhitungan SCF awal, optimasi geometri, pembuatan metode baru pada geometri optimum, dan energi final. Hasil yang dituju berupa energi, geometri, orbital, dan sifat molekul.

Prinsip kerja yang digunakan adalah memisahkan input, proses, validasi, dan ekspor. Pemisahan ini membuat kesalahan lebih mudah dilacak dan mencegah hasil lama dianggap sebagai hasil dari parameter baru. Setiap langkah perlu meninggalkan bukti

yang dapat dibaca kembali: nilai parameter, versi pustaka, status proses, nama berkas, serta pesan kesalahan bila terjadi kegagalan.

Risiko utama pada topik ini adalah SCF tidak konvergen, spin keliru, atau interpretasi lintas metode. Karena itu, notebook harus memuat pemeriksaan eksplisit dan bukan hanya mengandalkan keluaran yang tampak masuk akal. Pembaca dianjurkan menjalankan notebook dari keadaan runtime bersih, membandingkan hasil dengan batas kewajaran, dan menyimpan artefak penting di lokasi persisten.

Batas penerapan: Contoh pada buku berfungsi sebagai pola kerja. Ukuran molekul, biaya komputasi, versi pustaka, dan perangkat keras dapat mengubah waktu serta strategi yang diperlukan.

Prosedur Bertahap

Optimasi B3LYP/def2-TZVP merupakan simpul penting dalam alur kerja notebook ilmiah karena menentukan bagaimana perhitungan struktur elektronik disiapkan, dijalankan, dan diperiksa. Perhitungan SCF awal, optimasi geometri, pembuatan metode baru pada geometri optimum, dan energi final. Hasil yang dituju berupa energi, geometri, orbital, dan sifat molekul.

444. Tuliskan tujuan operasional Optimasi B3LYP/def2-TZVP dan tentukan keluaran yang harus tersedia pada akhir sel.
445. Tetapkan parameter pada satu sel konfigurasi agar perubahan dapat ditelusuri dan tidak tersebar di banyak tempat.
446. Siapkan perhitungan struktur elektronik dengan input yang eksplisit; hindari nilai tersembunyi yang hanya tersimpan dalam state kernel.
447. Jalankan pemeriksaan prasyarat, termasuk keberadaan pustaka, bentuk data, tipe nilai, serta konsistensi satuan.
448. Eksekusi proses utama, tangkap status konvergensi atau keberhasilan, dan hentikan alur apabila syarat minimum tidak terpenuhi.
449. Simpan energi, geometri, orbital, dan sifat molekul dengan nama berkas informatif, kemudian tulis interpretasi yang membedakan fakta komputasi dan kesimpulan ilmiah.

Titik kontrol: Jangan melanjutkan ke interpretasi apabila validasi input atau status proses belum memenuhi kriteria.

Kode Dasar Siap Colab

Pembahasan Optimasi B3LYP/def2-TZVP ditempatkan dalam konteks praktik Google Colab agar pembaca dapat menghubungkan konsep, prosedur, kode, dan interpretasi secara utuh. Perhitungan SCF awal, optimasi geometri, pembuatan metode baru pada geometri optimum, dan energi final. Hasil yang dituju berupa energi, geometri, orbital, dan sifat molekul.

Sel kode contoh

```
from pyscf import dft
from pyscf.geomopt.geometric_solver import optimize

mf_opt = dft.RKS(mol_h2o).density_fit()
mf_opt.xc = "b3lyp"; mf_opt.grids.level = 4
mf_opt.conv_tol = 1e-9; mf_opt.max_cycle = 200
mol_opt = optimize(mf_opt, maxsteps=100)

mf_final = dft.RKS(mol_opt).density_fit()
mf_final.xc = "b3lyp"
e_final = mf_final.kernel()
print("Energi optimum:", e_final)
```

Cara menjalankan: Tempatkan kode dalam satu atau beberapa sel sesuai komentar. Untuk baris %pip atau perintah shell, gunakan runtime Google Colab.

Penjelasan Baris Kode Penting

Potongan kode	Fungsi
from pyscf import dft	Mencetak status agar pengguna tidak hanya bergantung pada keluaran visual.
from pyscf.geomopt.geometric_solver import optim	Mendefinisikan input atau parameter yang akan digunakan pada langkah berikutnya.
mf_opt = dft.RKS(mol_h2o).density_fit()	Menambahkan pemeriksaan untuk mencegah hasil tidak valid diteruskan.
mf_opt.xc = "b3lyp"; mf_opt.grids.level = 4	Menambahkan pemeriksaan untuk mencegah hasil tidak valid diteruskan.
mf_opt.conv_tol = 1e-9; mf_opt.max_cycle = 200	Mencetak status agar pengguna tidak hanya bergantung pada keluaran visual.
mol_opt = optimize(mf_opt, maxsteps=100)	Mencetak status agar pengguna tidak hanya bergantung pada keluaran visual.
mf_final = dft.RKS(mol_opt).density_fit()	Mendefinisikan input atau parameter yang akan digunakan pada langkah berikutnya.

```
mf_final.xc = "b3lyp"
```

Mencetak status agar pengguna tidak hanya bergantung pada keluaran visual.

Nama variabel dipilih agar maknanya terbaca. Hindari singkatan yang tidak dijelaskan, nilai konstan tanpa satuan, serta penggunaan variabel dari sel lama. Komentar harus menerangkan alasan atau asumsi, bukan sekadar mengulang sintaks.

Validasi dan Interpretasi Hasil

Prinsip kerja yang digunakan adalah memisahkan input, proses, validasi, dan ekspor. Pemisahan ini membuat kesalahan lebih mudah dilacak dan mencegah hasil lama dianggap sebagai hasil dari parameter baru. Setiap langkah perlu meninggalkan bukti yang dapat dibaca kembali: nilai parameter, versi pustaka, status proses, nama berkas, serta pesan kesalahan bila terjadi kegagalan.

Aspek validasi	Kriteria lulus
Input	Tipe, bentuk, satuan, muatan, spin, dan jalur berkas dinyatakan eksplisit.
Lingkungan	Versi Python dan pustaka tercatat; backend CPU/GPU terdeteksi.
Proses	Status eksekusi atau konvergensi diperiksa sebelum hasil dipakai.
Numerik	Tidak ada NaN/inf; ukuran array dan jumlah atom konsisten.
Keluaran	Energi, geometri, orbital, dan sifat molekul disimpan dan dapat dibuka kembali.
Interpretasi	Klaim tidak melampaui metode, basis set, atau data yang dihitung.
Interpretasi aman: Sebutkan metode, parameter, satuan, status konvergensi, dan keterbatasan saat menuliskan kesimpulan.	

Kesalahan Umum dan Troubleshooting

Risiko utama pada topik ini adalah SCF tidak konvergen, spin keliru, atau interpretasi lintas metode. Karena itu, notebook harus memuat pemeriksaan eksplisit dan bukan hanya mengandalkan keluaran yang tampak masuk akal. Pembaca dianjurkan menjalankan

notebook dari keadaan runtime bersih, membandingkan hasil dengan batas kewajaran, dan menyimpan artefak penting di lokasi persisten.

Gejala	Tindakan perbaikan
Sel dijalankan tidak berurutan	Restart session, lalu gunakan Run all dan periksa variabel prasyarat.
Pustaka tidak ditemukan	Jalankan kembali sel %pip, periksa nama paket, lalu restart bila dependensi inti berubah.
Input tampak benar tetapi gagal	Cetak tipe, shape, satuan, muatan, spin, dan beberapa baris pertama data.
Scf tidak konvergen, spin keliru, atau interpretasi lintas metode	Tambahkan validasi sebelum proses utama dan hentikan eksekusi dengan pesan yang spesifik.
Hasil berbeda antar-runtime	Catat versi runtime/paket, random seed, parameter, dan perangkat yang digunakan.
Aturan troubleshooting: Perbaiki satu hal pada satu waktu, simpan pesan kesalahan lengkap, dan catat perubahan versi atau parameter.	

Latihan Modifikasi dan Refleksi

450. Ubah satu parameter pada contoh Optimasi B3LYP/def2-TZVP, jalankan ulang dari runtime bersih, dan jelaskan perubahan keluaran.
451. Tambahkan satu validasi baru yang mencegah SCF tidak konvergen, spin keliru, atau interpretasi lintas metode.
452. Buat berkas ringkasan TXT atau JSON yang memuat input, status proses, waktu, dan hasil utama.
453. Jelaskan bagian mana yang merupakan fakta keluaran program dan bagian mana yang merupakan interpretasi ilmiah.
454. Susun satu kasus uji yang sengaja salah, lalu pastikan notebook menghasilkan pesan kesalahan yang informatif.

Refleksi: Tuliskan apa yang berubah setelah kode dimodifikasi, bukti apa yang menunjukkan keberhasilan, dan bagian mana yang masih memerlukan verifikasi eksternal.

Rubrik

Deskripsi

Skor 4	Kode berjalan dari runtime bersih, validasi lengkap, interpretasi tepat.
Skor 3	Kode berjalan dan hasil tersimpan, tetapi dokumentasi belum lengkap.
Skor 2	Kode hanya berjalan setelah intervensi atau state tertentu.
Skor 1	Kode belum berjalan dan diagnosis kesalahan belum sistematis.

4.2.3 Frekuensi, geometri, dan visualisasi hasil

Pada bagian Frekuensi, geometri, dan visualisasi hasil, perhatian tidak hanya diarahkan pada keberhasilan menjalankan kode, tetapi juga pada ketepatan asumsi, keterlacakan parameter, dan mutu keluaran. Panjang ikatan O-H, sudut H-O-H, frekuensi vibrasi, pemeriksaan frekuensi imajiner, struktur XYZ, dan tampilan 3D. Hasil yang dituju berupa hasil yang dapat diperiksa dan disimpan.

Prinsip kerja yang digunakan adalah memisahkan input, proses, validasi, dan ekspor. Pemisahan ini membuat kesalahan lebih mudah dilacak dan mencegah hasil lama dianggap sebagai hasil dari parameter baru. Setiap langkah perlu meninggalkan bukti yang dapat dibaca kembali: nilai parameter, versi pustaka, status proses, nama berkas, serta pesan kesalahan bila terjadi kegagalan.

Tujuan operasional: Pembaca mampu menjelaskan Frekuensi, geometri, dan visualisasi hasil, menjalankan contoh minimal, memeriksa keberhasilan, dan menyimpan bukti hasil.

Komponen	Kriteria
Konsep	Panjang ikatan O-H, sudut H-O-H, frekuensi vibrasi, pemeriksaan frekuensi imajiner, struktur XYZ, dan tampilan 3D.
Input	Parameter dan data dinyatakan eksplisit.
Proses	Langkah dapat diulang dari runtime bersih.
Output	Nilai, berkas, atau visualisasi yang dapat diperiksa.

Batas

Interpretasi mengikuti metode dan konteks.

Landasan Teori dan Batas Penerapan

Pada bagian Frekuensi, geometri, dan visualisasi hasil, perhatian tidak hanya diarahkan pada keberhasilan menjalankan kode, tetapi juga pada ketepatan asumsi, keterlacakan parameter, dan mutu keluaran. Panjang ikatan O-H, sudut H-O-H, frekuensi vibrasi, pemeriksaan frekuensi imajiner, struktur XYZ, dan tampilan 3D. Hasil yang dituju berupa hasil yang dapat diperiksa dan disimpan.

Prinsip kerja yang digunakan adalah memisahkan input, proses, validasi, dan ekspor. Pemisahan ini membuat kesalahan lebih mudah dilacak dan mencegah hasil lama dianggap sebagai hasil dari parameter baru. Setiap langkah perlu meninggalkan bukti yang dapat dibaca kembali: nilai parameter, versi pustaka, status proses, nama berkas, serta pesan kesalahan bila terjadi kegagalan.

Risiko utama pada topik ini adalah parameter tidak terdokumentasi dan keluaran tidak tervalidasi. Karena itu, notebook harus memuat pemeriksaan eksplisit dan bukan hanya mengandalkan keluaran yang tampak masuk akal. Pembaca dianjurkan menjalankan notebook dari keadaan runtime bersih, membandingkan hasil dengan batas kewajaran, dan menyimpan artefak penting di lokasi persisten.

Batas penerapan: Contoh pada buku berfungsi sebagai pola kerja. Ukuran molekul, biaya komputasi, versi pustaka, dan perangkat keras dapat mengubah waktu serta strategi yang diperlukan.

Prosedur Bertahap

Pembahasan Frekuensi, geometri, dan visualisasi hasil ditempatkan dalam konteks praktik Google Colab agar pembaca dapat menghubungkan konsep, prosedur, kode, dan interpretasi secara utuh. Panjang ikatan O-H, sudut H-O-H, frekuensi vibrasi, pemeriksaan frekuensi imajiner, struktur XYZ, dan tampilan 3D. Hasil yang dituju berupa hasil yang dapat diperiksa dan disimpan.

455. Tuliskan tujuan operasional Frekuensi, geometri, dan visualisasi hasil dan tentukan keluaran yang harus tersedia pada akhir sel.
456. Tetapkan parameter pada satu sel konfigurasi agar perubahan dapat ditelusuri dan tidak tersebar di banyak tempat.
457. Siapkan prosedur komputasi dengan input yang eksplisit; hindari nilai tersembunyi yang hanya tersimpan dalam state kernel.
458. Jalankan pemeriksaan prasyarat, termasuk keberadaan pustaka, bentuk data, tipe nilai, serta konsistensi satuan.
459. Eksekusi proses utama, tangkap status konvergensi atau keberhasilan, dan hentikan alur apabila syarat minimum tidak terpenuhi.

460. Simpan hasil yang dapat diperiksa dan disimpan dengan nama berkas informatif, kemudian tulis interpretasi yang membedakan fakta komputasi dan kesimpulan ilmiah.

Titik kontrol: Jangan melanjutkan ke interpretasi apabila validasi input atau status proses belum memenuhi kriteria.

Kode Dasar Siap Colab

Pembahasan Frekuensi, geometri, dan visualisasi hasil ditempatkan dalam konteks praktik Google Colab agar pembaca dapat menghubungkan konsep, prosedur, kode, dan interpretasi secara utuh. Panjang ikatan O-H, sudut H-O-H, frekuensi vibrasi, pemeriksaan frekuensi imajiner, struktur XYZ, dan tampilan 3D. Hasil yang dituju berupa hasil yang dapat diperiksa dan disimpan.

Sel kode contoh

```
import numpy as np
from pyscf.hessian import thermo

coords = mol_opt.atom_coords(unit="Angstrom")
r_oh = np.linalg.norm(coords[1] - coords[0])
print("Panjang O-H:", r_oh, "Å")

# Frekuensi memerlukan Hessian pada metode yang telah konvergen.
hess = mf_final.Hessian().kernel()
freq_info = thermo.harmonic_analysis(mol_opt, hess)
print("Frekuensi (cm-1):", freq_info["freq_wavenumber"])
```

Cara menjalankan: Tempatkan kode dalam satu atau beberapa sel sesuai komentar. Untuk baris %pip atau perintah shell, gunakan runtime Google Colab.

Penjelasan Baris Kode Penting

Potongan kode	Fungsi
import numpy as np	Mencetak status agar pengguna tidak hanya bergantung pada keluaran visual.
from pyscf.hessian import thermo	Menambahkan pemeriksaan untuk mencegah hasil tidak valid diteruskan.

<code>coords = mol_opt.atom_coords(unit="Angstrom")</code>	Mencetak status agar pengguna tidak hanya bergantung pada keluaran visual.
<code>r_oh = np.linalg.norm(coords[1] - coords[0])</code>	Mencetak status agar pengguna tidak hanya bergantung pada keluaran visual.
<code>print("Panjang O-H:", r_oh, "Å")</code>	Mendefinisikan input atau parameter yang akan digunakan pada langkah berikutnya.
<code># Frekuensi memerlukan Hessian pada metode yang</code>	Mendefinisikan input atau parameter yang akan digunakan pada langkah berikutnya.
<code>hess = mf_final.Hessian().kernel()</code>	Menjalankan operasi utama dan menyimpan hasil pada variabel yang jelas.
<code>freq_info = thermo.harmonic_analysis(mol_opt, he</code>	Menjalankan operasi utama dan menyimpan hasil pada variabel yang jelas.

Nama variabel dipilih agar maknanya terbaca. Hindari singkatan yang tidak dijelaskan, nilai konstan tanpa satuan, serta penggunaan variabel dari sel lama. Komentar harus menerangkan alasan atau asumsi, bukan sekadar mengulang sintaks.

Validasi dan Interpretasi Hasil

Prinsip kerja yang digunakan adalah memisahkan input, proses, validasi, dan ekspor. Pemisahan ini membuat kesalahan lebih mudah dilacak dan mencegah hasil lama dianggap sebagai hasil dari parameter baru. Setiap langkah perlu meninggalkan bukti yang dapat dibaca kembali: nilai parameter, versi pustaka, status proses, nama berkas, serta pesan kesalahan bila terjadi kegagalan.

Aspek validasi	Kriteria lulus
Input	Tipe, bentuk, satuan, muatan, spin, dan jalur berkas dinyatakan eksplisit.
Lingkungan	Versi Python dan pustaka tercatat; backend CPU/GPU terdeteksi.

Proses	Status eksekusi atau konvergensi diperiksa sebelum hasil dipakai.
Numerik	Tidak ada NaN/inf; ukuran array dan jumlah atom konsisten.
Keluaran	Hasil yang dapat diperiksa dan disimpan disimpan dan dapat dibuka kembali.
Interpretasi	Klaim tidak melampaui metode, basis set, atau data yang dihitung.

Interpretasi aman: Sebutkan metode, parameter, satuan, status konvergensi, dan keterbatasan saat menuliskan kesimpulan.

Kesalahan Umum dan Troubleshooting

Risiko utama pada topik ini adalah parameter tidak terdokumentasi dan keluaran tidak tervalidasi. Karena itu, notebook harus memuat pemeriksaan eksplisit dan bukan hanya mengandalkan keluaran yang tampak masuk akal. Pembaca dianjurkan menjalankan notebook dari keadaan runtime bersih, membandingkan hasil dengan batas kewajaran, dan menyimpan artefak penting di lokasi persisten.

Gejala	Tindakan perbaikan
Sel dijalankan tidak berurutan	Restart session, lalu gunakan Run all dan periksa variabel prasyarat.
Pustaka tidak ditemukan	Jalankan kembali sel %pip, periksa nama paket, lalu restart bila dependensi inti berubah.
Input tampak benar tetapi gagal	Cetak tipe, shape, satuan, muatan, spin, dan beberapa baris pertama data.
Parameter tidak terdokumentasi dan keluaran tidak tervalidasi	Tambahkan validasi sebelum proses utama dan hentikan eksekusi dengan pesan yang spesifik.
Hasil berbeda antar-runtime	Catat versi runtime/paket, random seed, parameter, dan perangkat yang digunakan.

Aturan troubleshooting: Perbaiki satu hal pada satu waktu, simpan pesan kesalahan lengkap, dan catat perubahan versi atau parameter.

Latihan Modifikasi dan Refleksi

461. Ubah satu parameter pada contoh Frekuensi, geometri, dan visualisasi hasil, jalankan ulang dari runtime bersih, dan jelaskan perubahan keluaran.
462. Tambahkan satu validasi baru yang mencegah parameter tidak terdokumentasi dan keluaran tidak tervalidasi.
463. Buat berkas ringkasan TXT atau JSON yang memuat input, status proses, waktu, dan hasil utama.
464. Jelaskan bagian mana yang merupakan fakta keluaran program dan bagian mana yang merupakan interpretasi ilmiah.
465. Susun satu kasus uji yang sengaja salah, lalu pastikan notebook menghasilkan pesan kesalahan yang informatif.

Refleksi: Tuliskan apa yang berubah setelah kode dimodifikasi, bukti apa yang menunjukkan keberhasilan, dan bagian mana yang masih memerlukan verifikasi eksternal.

Rubrik	Deskripsi
Skor 4	Kode berjalan dari runtime bersih, validasi lengkap, interpretasi tepat.
Skor 3	Kode berjalan dan hasil tersimpan, tetapi dokumentasi belum lengkap.
Skor 2	Kode hanya berjalan setelah intervensi atau state tertentu.
Skor 1	Kode belum berjalan dan diagnosis kesalahan belum sistematis.

4.3 Analisis Keluaran Komputasi

Analisis Keluaran Komputasi mengorganisasi tiga komponen yang saling bergantung. Pembahasan bergerak dari konsep, menuju prosedur, kemudian menghasilkan artefak yang dapat divalidasi dan disimpan.

Sub-subbab

Fokus

4.3.1 Energi dan konvergensi	Energi struktur awal, energi optimum, perubahan energi dalam Hartree dan kJ mol ⁻¹ , status SCF, serta langkah optimasi.
4.3.2 HOMO, LUMO, celah energi, dan momen dipol	Okupansi orbital, konversi Hartree ke eV, kanal alfa-beta untuk sistem open-shell, dan dipol dalam Debye.
4.3.3 Interpretasi kritis dan keterbatasan	Perbedaan energi absolut dan relatif, ketergantungan basis set, celah Kohn-Sham, minimum lokal, dan kebutuhan analisis frekuensi.
Produk belajar: Satu notebook terstruktur, satu berkas keluaran, satu catatan validasi, dan satu interpretasi singkat yang menyatakan batas hasil.	

Rencana Praktik dan Produk Belajar

466. Pelajari Energi dan konvergensi dan jalankan kode dasar yang menyertainya.
467. Pelajari HOMO, LUMO, celah energi, dan momen dipol dan jalankan kode dasar yang menyertainya.
468. Pelajari Interpretasi kritis dan keterbatasan dan jalankan kode dasar yang menyertainya.

Komponen	Kriteria
Waktu	60-120 menit, bergantung pada komputasi.
Input	Notebook baru atau salinan notebook bab.
Produk	Kode, output, berkas hasil, dan refleksi.
Uji	Restart session lalu Run all.
Dokumentasi	Versi, parameter, status, dan waktu eksekusi.
Kriteria selesai: Semua sel dapat dijalankan berurutan, pesan kesalahan ditangani, dan hasil utama tersimpan di lokasi persisten.	

4.3.1 Energi dan konvergensi

Pembahasan Energi dan konvergensi ditempatkan dalam konteks praktik Google Colab agar pembaca dapat menghubungkan konsep, prosedur, kode, dan interpretasi secara utuh. Energi struktur awal, energi optimum, perubahan energi dalam Hartree dan kJ mol⁻¹, status SCF, serta langkah optimasi. Hasil yang dituju berupa energi, geometri, orbital, dan sifat molekul.

Prinsip kerja yang digunakan adalah memisahkan input, proses, validasi, dan ekspor. Pemisahan ini membuat kesalahan lebih mudah dilacak dan mencegah hasil lama dianggap sebagai hasil dari parameter baru. Setiap langkah perlu meninggalkan bukti yang dapat dibaca kembali: nilai parameter, versi pustaka, status proses, nama berkas, serta pesan kesalahan bila terjadi kegagalan.

Tujuan operasional: Pembaca mampu menjelaskan Energi dan konvergensi, menjalankan contoh minimal, memeriksa keberhasilan, dan menyimpan bukti hasil.

Komponen	Kriteria
Konsep	Energi struktur awal, energi optimum, perubahan energi dalam Hartree dan kJ mol ⁻¹ , status SCF, serta langkah optimasi.
Input	Parameter dan data dinyatakan eksplisit.
Proses	Langkah dapat diulang dari runtime bersih.
Output	Nilai, berkas, atau visualisasi yang dapat diperiksa.
Batas	Interpretasi mengikuti metode dan konteks.

Landasan Teori dan Batas Penerapan

Energi dan konvergensi merupakan simpul penting dalam alur kerja notebook ilmiah karena menentukan bagaimana perhitungan struktur elektronik disiapkan, dijalankan, dan diperiksa. Energi struktur awal, energi optimum, perubahan energi dalam Hartree dan kJ mol⁻¹, status SCF, serta langkah optimasi. Hasil yang dituju berupa energi, geometri, orbital, dan sifat molekul.

Prinsip kerja yang digunakan adalah memisahkan input, proses, validasi, dan ekspor. Pemisahan ini membuat kesalahan lebih mudah dilacak dan mencegah hasil lama dianggap sebagai hasil dari parameter baru. Setiap langkah perlu meninggalkan bukti

yang dapat dibaca kembali: nilai parameter, versi pustaka, status proses, nama berkas, serta pesan kesalahan bila terjadi kegagalan.

Risiko utama pada topik ini adalah SCF tidak konvergen, spin keliru, atau interpretasi lintas metode. Karena itu, notebook harus memuat pemeriksaan eksplisit dan bukan hanya mengandalkan keluaran yang tampak masuk akal. Pembaca dianjurkan menjalankan notebook dari keadaan runtime bersih, membandingkan hasil dengan batas kewajaran, dan menyimpan artefak penting di lokasi persisten.

Batas penerapan: Contoh pada buku berfungsi sebagai pola kerja. Ukuran molekul, biaya komputasi, versi pustaka, dan perangkat keras dapat mengubah waktu serta strategi yang diperlukan.

Prosedur Bertahap

Pembahasan Energi dan konvergensi ditempatkan dalam konteks praktik Google Colab agar pembaca dapat menghubungkan konsep, prosedur, kode, dan interpretasi secara utuh. Energi struktur awal, energi optimum, perubahan energi dalam Hartree dan kJ mol⁻¹, status SCF, serta langkah optimasi. Hasil yang dituju berupa energi, geometri, orbital, dan sifat molekul.

469. Tuliskan tujuan operasional Energi dan konvergensi dan tentukan keluaran yang harus tersedia pada akhir sel.
470. Tetapkan parameter pada satu sel konfigurasi agar perubahan dapat ditelusuri dan tidak tersebar di banyak tempat.
471. Siapkan perhitungan struktur elektronik dengan input yang eksplisit; hindari nilai tersembunyi yang hanya tersimpan dalam state kernel.
472. Jalankan pemeriksaan prasyarat, termasuk keberadaan pustaka, bentuk data, tipe nilai, serta konsistensi satuan.
473. Eksekusi proses utama, tangkap status konvergensi atau keberhasilan, dan hentikan alur apabila syarat minimum tidak terpenuhi.
474. Simpan energi, geometri, orbital, dan sifat molekul dengan nama berkas informatif, kemudian tulis interpretasi yang membedakan fakta komputasi dan kesimpulan ilmiah.

Titik kontrol: Jangan melanjutkan ke interpretasi apabila validasi input atau status proses belum memenuhi kriteria.

Kode Dasar Siap Colab

Energi dan konvergensi merupakan simpul penting dalam alur kerja notebook ilmiah karena menentukan bagaimana perhitungan struktur elektronik disiapkan, dijalankan, dan diperiksa. Energi struktur awal, energi optimum, perubahan energi dalam Hartree dan kJ mol⁻¹, status SCF, serta langkah optimasi. Hasil yang dituju berupa energi, geometri, orbital, dan sifat molekul.

Sel kode contoh

```
HARTREE_TO_KJMOL = 2625.499638

if not mf_final.converged:
    raise RuntimeError("SCF final tidak konvergen")
delta_e = e_final - e0
print(f"E awal : {e0:.12f} Eh")
print(f"E akhir : {e_final:.12f} Eh")
print(f"Delta E : {delta_e:.12f} Eh = {delta_e*HARTREE_TO_KJMOL:.3f} kJ/mol")
```

Cara menjalankan: Tempatkan kode dalam satu atau beberapa sel sesuai komentar. Untuk baris %pip atau perintah shell, gunakan runtime Google Colab.

Penjelasan Baris Kode Penting

Potongan kode	Fungsi
HARTREE_TO_KJMOL = 2625.499638	Menjalankan operasi utama dan menyimpan hasil pada variabel yang jelas.
if not mf_final.converged:	Menjalankan operasi utama dan menyimpan hasil pada variabel yang jelas.
raise RuntimeError("SCF final tidak konverge")	Mendefinisikan input atau parameter yang akan digunakan pada langkah berikutnya.
delta_e = e_final - e0	Menambahkan pemeriksaan untuk mencegah hasil tidak valid diteruskan.
print(f"E awal : {e0:.12f} Eh")	Mendefinisikan input atau parameter yang akan digunakan pada langkah berikutnya.
print(f"E akhir : {e_final:.12f} Eh")	Mencetak status agar pengguna tidak hanya bergantung pada keluaran visual.
print(f"Delta E : {delta_e:.12f} Eh = {delta_e*H	Menjalankan operasi utama dan menyimpan hasil pada variabel yang jelas.

Nama variabel dipilih agar maknanya terbaca. Hindari singkatan yang tidak dijelaskan, nilai konstan tanpa satuan, serta penggunaan variabel dari sel lama. Komentar harus menerangkan alasan atau asumsi, bukan sekadar mengulang sintaks.

Validasi dan Interpretasi Hasil

Prinsip kerja yang digunakan adalah memisahkan input, proses, validasi, dan ekspor. Pemisahan ini membuat kesalahan lebih mudah dilacak dan mencegah hasil lama dianggap sebagai hasil dari parameter baru. Setiap langkah perlu meninggalkan bukti yang dapat dibaca kembali: nilai parameter, versi pustaka, status proses, nama berkas, serta pesan kesalahan bila terjadi kegagalan.

Aspek validasi	Kriteria lulus
Input	Tipe, bentuk, satuan, muatan, spin, dan jalur berkas dinyatakan eksplisit.
Lingkungan	Versi Python dan pustaka tercatat; backend CPU/GPU terdeteksi.
Proses	Status eksekusi atau konvergensi diperiksa sebelum hasil dipakai.
Numerik	Tidak ada NaN/inf; ukuran array dan jumlah atom konsisten.
Keluaran	Energi, geometri, orbital, dan sifat molekul disimpan dan dapat dibuka kembali.
Interpretasi	Klaim tidak melampaui metode, basis set, atau data yang dihitung.
Interpretasi aman: Sebutkan metode, parameter, satuan, status konvergensi, dan keterbatasan saat menuliskan kesimpulan.	

Kesalahan Umum dan Troubleshooting

Risiko utama pada topik ini adalah SCF tidak konvergen, spin keliru, atau interpretasi lintas metode. Karena itu, notebook harus memuat pemeriksaan eksplisit dan bukan hanya mengandalkan keluaran yang tampak masuk akal. Pembaca dianjurkan menjalankan notebook dari keadaan runtime bersih, membandingkan hasil dengan batas kewajaran, dan menyimpan artefak penting di lokasi persisten.

Gejala	Tindakan perbaikan
Sel dijalankan tidak berurutan	Restart session, lalu gunakan Run all dan periksa variabel prasyarat.

Pustaka tidak ditemukan	Jalankan kembali sel %pip, periksa nama paket, lalu restart bila dependensi inti berubah.
Input tampak benar tetapi gagal	Cetak tipe, shape, satuan, muatan, spin, dan beberapa baris pertama data.
Scf tidak konvergen, spin keliru, atau interpretasi lintas metode	Tambahkan validasi sebelum proses utama dan hentikan eksekusi dengan pesan yang spesifik.
Hasil berbeda antar-runtime	Catat versi runtime/paket, random seed, parameter, dan perangkat yang digunakan.
Aturan troubleshooting: Perbaiki satu hal pada satu waktu, simpan pesan kesalahan lengkap, dan catat perubahan versi atau parameter.	

Latihan Modifikasi dan Refleksi

475. Ubah satu parameter pada contoh Energi dan konvergensi, jalankan ulang dari runtime bersih, dan jelaskan perubahan keluaran.
476. Tambahkan satu validasi baru yang mencegah SCF tidak konvergen, spin keliru, atau interpretasi lintas metode.
477. Buat berkas ringkasan TXT atau JSON yang memuat input, status proses, waktu, dan hasil utama.
478. Jelaskan bagian mana yang merupakan fakta keluaran program dan bagian mana yang merupakan interpretasi ilmiah.
479. Susun satu kasus uji yang sengaja salah, lalu pastikan notebook menghasilkan pesan kesalahan yang informatif.

Refleksi: Tuliskan apa yang berubah setelah kode dimodifikasi, bukti apa yang menunjukkan keberhasilan, dan bagian mana yang masih memerlukan verifikasi eksternal.

Rubrik	Deskripsi
Skor 4	Kode berjalan dari runtime bersih, validasi lengkap, interpretasi tepat.
Skor 3	Kode berjalan dan hasil tersimpan, tetapi dokumentasi belum lengkap.

Skor 2	Kode hanya berjalan setelah intervensi atau state tertentu.
Skor 1	Kode belum berjalan dan diagnosis kesalahan belum sistematis.

4.3.2 HOMO, LUMO, celah energi, dan momen dipol

HOMO, LUMO, celah energi, dan momen dipol merupakan simpul penting dalam alur kerja notebook ilmiah karena menentukan bagaimana perhitungan struktur elektronik disiapkan, dijalankan, dan diperiksa. Okupansi orbital, konversi Hartree ke eV, kanal alfa-beta untuk sistem open-shell, dan dipol dalam Debye. Hasil yang dituju berupa energi, geometri, orbital, dan sifat molekul.

Prinsip kerja yang digunakan adalah memisahkan input, proses, validasi, dan ekspor. Pemisahan ini membuat kesalahan lebih mudah dilacak dan mencegah hasil lama dianggap sebagai hasil dari parameter baru. Setiap langkah perlu meninggalkan bukti yang dapat dibaca kembali: nilai parameter, versi pustaka, status proses, nama berkas, serta pesan kesalahan bila terjadi kegagalan.

Tujuan operasional: Pembaca mampu menjelaskan HOMO, LUMO, celah energi, dan momen dipol, menjalankan contoh minimal, memeriksa keberhasilan, dan menyimpan bukti hasil.

Komponen	Kriteria
Konsep	Okupansi orbital, konversi Hartree ke eV, kanal alfa-beta untuk sistem open-shell, dan dipol dalam Debye.
Input	Parameter dan data dinyatakan eksplisit.
Proses	Langkah dapat diulang dari runtime bersih.
Output	Nilai, berkas, atau visualisasi yang dapat diperiksa.
Batas	Interpretasi mengikuti metode dan konteks.

Landasan Teori dan Batas Penerapan

Pembahasan HOMO, LUMO, celah energi, dan momen dipol ditempatkan dalam konteks praktik Google Colab agar pembaca dapat menghubungkan konsep, prosedur, kode, dan

interpretasi secara utuh. Okupansi orbital, konversi Hartree ke eV, kanal alfa-beta untuk sistem open-shell, dan dipol dalam Debye. Hasil yang dituju berupa energi, geometri, orbital, dan sifat molekul.

Prinsip kerja yang digunakan adalah memisahkan input, proses, validasi, dan ekspor. Pemisahan ini membuat kesalahan lebih mudah dilacak dan mencegah hasil lama dianggap sebagai hasil dari parameter baru. Setiap langkah perlu meninggalkan bukti yang dapat dibaca kembali: nilai parameter, versi pustaka, status proses, nama berkas, serta pesan kesalahan bila terjadi kegagalan.

Risiko utama pada topik ini adalah SCF tidak konvergen, spin keliru, atau interpretasi lintas metode. Karena itu, notebook harus memuat pemeriksaan eksplisit dan bukan hanya mengandalkan keluaran yang tampak masuk akal. Pembaca dianjurkan menjalankan notebook dari keadaan runtime bersih, membandingkan hasil dengan batas kewajaran, dan menyimpan artefak penting di lokasi persisten.

Batas penerapan: Contoh pada buku berfungsi sebagai pola kerja. Ukuran molekul, biaya komputasi, versi pustaka, dan perangkat keras dapat mengubah waktu serta strategi yang diperlukan.

Prosedur Bertahap

Pembahasan HOMO, LUMO, celah energi, dan momen dipol ditempatkan dalam konteks praktik Google Colab agar pembaca dapat menghubungkan konsep, prosedur, kode, dan interpretasi secara utuh. Okupansi orbital, konversi Hartree ke eV, kanal alfa-beta untuk sistem open-shell, dan dipol dalam Debye. Hasil yang dituju berupa energi, geometri, orbital, dan sifat molekul.

480. Tuliskan tujuan operasional HOMO, LUMO, celah energi, dan momen dipol dan tentukan keluaran yang harus tersedia pada akhir sel.
481. Tetapkan parameter pada satu sel konfigurasi agar perubahan dapat ditelusuri dan tidak tersebar di banyak tempat.
482. Siapkan perhitungan struktur elektronik dengan input yang eksplisit; hindari nilai tersembunyi yang hanya tersimpan dalam state kernel.
483. Jalankan pemeriksaan prasyarat, termasuk keberadaan pustaka, bentuk data, tipe nilai, serta konsistensi satuan.
484. Eksekusi proses utama, tangkap status konvergensi atau keberhasilan, dan hentikan alur apabila syarat minimum tidak terpenuhi.
485. Simpan energi, geometri, orbital, dan sifat molekul dengan nama berkas informatif, kemudian tulis interpretasi yang membedakan fakta komputasi dan kesimpulan ilmiah.

Titik kontrol: Jangan melanjutkan ke interpretasi apabila validasi input atau status proses belum memenuhi kriteria.

Kode Dasar Siap Colab

HOMO, LUMO, celah energi, dan momen dipol merupakan simpul penting dalam alur kerja notebook ilmiah karena menentukan bagaimana perhitungan struktur elektronik disiapkan, dijalankan, dan diperiksa. Okupansi orbital, konversi Hartree ke eV, kanal alfa-beta untuk sistem open-shell, dan dipol dalam Debye. Hasil yang dituju berupa energi, geometri, orbital, dan sifat molekul.

Sel kode contoh

```
import numpy as np

occ = np.asarray(mf_final.mo_occ)
eps = np.asarray(mf_final.mo_energy)
homo = eps[np.where(occ > 0)[0][-1]]
lumo = eps[np.where(occ == 0)[0][0]]
print("HOMO (eV):", homo * 27.211386245988)
print("LUMO (eV):", lumo * 27.211386245988)
print("Gap (eV):", (lumo-homo) * 27.211386245988)
print("Dipol (Debye):", mf_final.dip_moment(unit="Debye"))
```

Cara menjalankan: Tempatkan kode dalam satu atau beberapa sel sesuai komentar. Untuk baris %pip atau perintah shell, gunakan runtime Google Colab.

Penjelasan Baris Kode Penting

Potongan kode	Fungsi
import numpy as np	Menambahkan pemeriksaan untuk mencegah hasil tidak valid diteruskan.
occ = np.asarray(mf_final.mo_occ)	Mencetak status agar pengguna tidak hanya bergantung pada keluaran visual.
eps = np.asarray(mf_final.mo_energy)	Menjalankan operasi utama dan menyimpan hasil pada variabel yang jelas.
homo = eps[np.where(occ > 0)[0][-1]]	Mendefinisikan input atau parameter yang akan digunakan pada langkah berikutnya.
lumo = eps[np.where(occ == 0)[0][0]]	Menjalankan operasi utama dan menyimpan hasil pada variabel yang jelas.

<pre>print("HOMO (eV):", homo * 27.211386245988)</pre>	Mendefinisikan input atau parameter yang akan digunakan pada langkah berikutnya.
<pre>print("LUMO (eV):", lumo * 27.211386245988)</pre>	Mencetak status agar pengguna tidak hanya bergantung pada keluaran visual.
<pre>print("Gap (eV):", (lumo-homo) * 27.211386245988)</pre>	Menjalankan operasi utama dan menyimpan hasil pada variabel yang jelas.

Nama variabel dipilih agar maknanya terbaca. Hindari singkatan yang tidak dijelaskan, nilai konstan tanpa satuan, serta penggunaan variabel dari sel lama. Komentar harus menerangkan alasan atau asumsi, bukan sekadar mengulang sintaks.

Validasi dan Interpretasi Hasil

Prinsip kerja yang digunakan adalah memisahkan input, proses, validasi, dan ekspor. Pemisahan ini membuat kesalahan lebih mudah dilacak dan mencegah hasil lama dianggap sebagai hasil dari parameter baru. Setiap langkah perlu meninggalkan bukti yang dapat dibaca kembali: nilai parameter, versi pustaka, status proses, nama berkas, serta pesan kesalahan bila terjadi kegagalan.

Aspek validasi	Kriteria lulus
Input	Tipe, bentuk, satuan, muatan, spin, dan jalur berkas dinyatakan eksplisit.
Lingkungan	Versi Python dan pustaka tercatat; backend CPU/GPU terdeteksi.
Proses	Status eksekusi atau konvergensi diperiksa sebelum hasil dipakai.
Numerik	Tidak ada NaN/inf; ukuran array dan jumlah atom konsisten.
Keluaran	Energi, geometri, orbital, dan sifat molekul disimpan dan dapat dibuka kembali.
Interpretasi	Klaim tidak melampaui metode, basis set, atau data yang dihitung.

Interpretasi aman: Sebutkan metode, parameter, satuan, status konvergensi, dan keterbatasan saat menuliskan kesimpulan.

Kesalahan Umum dan Troubleshooting

Risiko utama pada topik ini adalah SCF tidak konvergen, spin keliru, atau interpretasi lintas metode. Karena itu, notebook harus memuat pemeriksaan eksplisit dan bukan hanya mengandalkan keluaran yang tampak masuk akal. Pembaca dianjurkan menjalankan notebook dari keadaan runtime bersih, membandingkan hasil dengan batas kewajaran, dan menyimpan artefak penting di lokasi persisten.

Gejala	Tindakan perbaikan
Sel dijalankan tidak berurutan	Restart session, lalu gunakan Run all dan periksa variabel prasyarat.
Pustaka tidak ditemukan	Jalankan kembali sel %pip, periksa nama paket, lalu restart bila dependensi inti berubah.
Input tampak benar tetapi gagal	Cetak tipe, shape, satuan, muatan, spin, dan beberapa baris pertama data.
Scf tidak konvergen, spin keliru, atau interpretasi lintas metode	Tambahkan validasi sebelum proses utama dan hentikan eksekusi dengan pesan yang spesifik.
Hasil berbeda antar-runtime	Catat versi runtime/paket, random seed, parameter, dan perangkat yang digunakan.
Aturan troubleshooting: Perbaiki satu hal pada satu waktu, simpan pesan kesalahan lengkap, dan catat perubahan versi atau parameter.	

Latihan Modifikasi dan Refleksi

486. Ubah satu parameter pada contoh HOMO, LUMO, celah energi, dan momen dipol, jalankan ulang dari runtime bersih, dan jelaskan perubahan keluaran.
487. Tambahkan satu validasi baru yang mencegah SCF tidak konvergen, spin keliru, atau interpretasi lintas metode.
488. Buat berkas ringkasan TXT atau JSON yang memuat input, status proses, waktu, dan hasil utama.
489. Jelaskan bagian mana yang merupakan fakta keluaran program dan bagian mana yang merupakan interpretasi ilmiah.

490. Susun satu kasus uji yang sengaja salah, lalu pastikan notebook menghasilkan pesan kesalahan yang informatif.

Refleksi: Tuliskan apa yang berubah setelah kode dimodifikasi, bukti apa yang menunjukkan keberhasilan, dan bagian mana yang masih memerlukan verifikasi eksternal.

Rubrik	Deskripsi
Skor 4	Kode berjalan dari runtime bersih, validasi lengkap, interpretasi tepat.
Skor 3	Kode berjalan dan hasil tersimpan, tetapi dokumentasi belum lengkap.
Skor 2	Kode hanya berjalan setelah intervensi atau state tertentu.
Skor 1	Kode belum berjalan dan diagnosis kesalahan belum sistematis.

4.3.3 Interpretasi kritis dan keterbatasan

Pembahasan Interpretasi kritis dan keterbatasan ditempatkan dalam konteks praktik Google Colab agar pembaca dapat menghubungkan konsep, prosedur, kode, dan interpretasi secara utuh. Perbedaan energi absolut dan relatif, ketergantungan basis set, celah Kohn-Sham, minimum lokal, dan kebutuhan analisis frekuensi. Hasil yang dituju berupa energi, geometri, orbital, dan sifat molekul.

Prinsip kerja yang digunakan adalah memisahkan input, proses, validasi, dan ekspor. Pemisahan ini membuat kesalahan lebih mudah dilacak dan mencegah hasil lama dianggap sebagai hasil dari parameter baru. Setiap langkah perlu meninggalkan bukti yang dapat dibaca kembali: nilai parameter, versi pustaka, status proses, nama berkas, serta pesan kesalahan bila terjadi kegagalan.

Tujuan operasional: Pembaca mampu menjelaskan Interpretasi kritis dan keterbatasan, menjalankan contoh minimal, memeriksa keberhasilan, dan menyimpan bukti hasil.

Komponen	Kriteria
Konsep	Perbedaan energi absolut dan relatif, ketergantungan basis set, celah Kohn-Sham, minimum lokal, dan kebutuhan analisis frekuensi.

Input	Parameter dan data dinyatakan eksplisit.
Proses	Langkah dapat diulang dari runtime bersih.
Output	Nilai, berkas, atau visualisasi yang dapat diperiksa.
Batas	Interpretasi mengikuti metode dan konteks.

Landasan Teori dan Batas Penerapan

Pada bagian Interpretasi kritis dan keterbatasan, perhatian tidak hanya diarahkan pada keberhasilan menjalankan kode, tetapi juga pada ketepatan asumsi, keterlacakan parameter, dan mutu keluaran. Perbedaan energi absolut dan relatif, ketergantungan basis set, celah Kohn-Sham, minimum lokal, dan kebutuhan analisis frekuensi. Hasil yang dituju berupa energi, geometri, orbital, dan sifat molekul.

Prinsip kerja yang digunakan adalah memisahkan input, proses, validasi, dan ekspor. Pemisahan ini membuat kesalahan lebih mudah dilacak dan mencegah hasil lama dianggap sebagai hasil dari parameter baru. Setiap langkah perlu meninggalkan bukti yang dapat dibaca kembali: nilai parameter, versi pustaka, status proses, nama berkas, serta pesan kesalahan bila terjadi kegagalan.

Risiko utama pada topik ini adalah SCF tidak konvergen, spin keliru, atau interpretasi lintas metode. Karena itu, notebook harus memuat pemeriksaan eksplisit dan bukan hanya mengandalkan keluaran yang tampak masuk akal. Pembaca dianjurkan menjalankan notebook dari keadaan runtime bersih, membandingkan hasil dengan batas kewajaran, dan menyimpan artefak penting di lokasi persisten.

Batas penerapan: Contoh pada buku berfungsi sebagai pola kerja. Ukuran molekul, biaya komputasi, versi pustaka, dan perangkat keras dapat mengubah waktu serta strategi yang diperlukan.

Prosedur Bertahap

Interpretasi kritis dan keterbatasan merupakan simpul penting dalam alur kerja notebook ilmiah karena menentukan bagaimana perhitungan struktur elektronik disiapkan, dijalankan, dan diperiksa. Perbedaan energi absolut dan relatif, ketergantungan basis set, celah Kohn-Sham, minimum lokal, dan kebutuhan analisis frekuensi. Hasil yang dituju berupa energi, geometri, orbital, dan sifat molekul.

491. Tuliskan tujuan operasional Interpretasi kritis dan keterbatasan dan tentukan keluaran yang harus tersedia pada akhir sel.

492. Tetapkan parameter pada satu sel konfigurasi agar perubahan dapat ditelusuri dan tidak tersebar di banyak tempat.
493. Siapkan perhitungan struktur elektronik dengan input yang eksplisit; hindari nilai tersembunyi yang hanya tersimpan dalam state kernel.
494. Jalankan pemeriksaan prasyarat, termasuk keberadaan pustaka, bentuk data, tipe nilai, serta konsistensi satuan.
495. Eksekusi proses utama, tangkap status konvergensi atau keberhasilan, dan hentikan alur apabila syarat minimum tidak terpenuhi.
496. Simpan energi, geometri, orbital, dan sifat molekul dengan nama berkas informatif, kemudian tulis interpretasi yang membedakan fakta komputasi dan kesimpulan ilmiah.

Titik kontrol: Jangan melanjutkan ke interpretasi apabila validasi input atau status proses belum memenuhi kriteria.

Kode Dasar Siap Colab

Pembahasan Interpretasi kritis dan keterbatasan ditempatkan dalam konteks praktik Google Colab agar pembaca dapat menghubungkan konsep, prosedur, kode, dan interpretasi secara utuh. Perbedaan energi absolut dan relatif, ketergantungan basis set, celah Kohn-Sham, minimum lokal, dan kebutuhan analisis frekuensi. Hasil yang dituju berupa energi, geometri, orbital, dan sifat molekul.

Sel kode contoh

```
result = {
    "method": "B3LYP",
    "basis": "def2-TZVP",
    "scf_converged": bool(mf_final.converged),
    "geometry_is_stationary": "perlu diverifikasi dengan frekuensi",
    "homo_lumo_note": "energi orbital Kohn-Sham, bukan gap eksperimen langsung",
}
for key, value in result.items():
    print(f"{key}: {value}")
```

Cara menjalankan: Tempatkan kode dalam satu atau beberapa sel sesuai komentar. Untuk baris %pip atau perintah shell, gunakan runtime Google Colab.

Penjelasan Baris Kode Penting

Potongan kode	Fungsi
<pre>result = {</pre>	Mendefinisikan input atau parameter yang akan digunakan pada langkah berikutnya.

"method": "B3LYP",	Menambahkan pemeriksaan untuk mencegah hasil tidak valid diteruskan.
"basis": "def2-TZVP",	Mendefinisikan input atau parameter yang akan digunakan pada langkah berikutnya.
"scf_converged": bool(mf_final.converged),	Mencetak status agar pengguna tidak hanya bergantung pada keluaran visual.
"geometry_is_stationary": "perlu diverifikas	Mendefinisikan input atau parameter yang akan digunakan pada langkah berikutnya.
"homo_lumo_note": "energi orbital Kohn- Sham,	Menambahkan pemeriksaan untuk mencegah hasil tidak valid diteruskan.
}	Mencetak status agar pengguna tidak hanya bergantung pada keluaran visual.
for key, value in result.items():	Menambahkan pemeriksaan untuk mencegah hasil tidak valid diteruskan.

Nama variabel dipilih agar maknanya terbaca. Hindari singkatan yang tidak dijelaskan, nilai konstan tanpa satuan, serta penggunaan variabel dari sel lama. Komentar harus menerangkan alasan atau asumsi, bukan sekadar mengulang sintaks.

Validasi dan Interpretasi Hasil

Prinsip kerja yang digunakan adalah memisahkan input, proses, validasi, dan ekspor. Pemisahan ini membuat kesalahan lebih mudah dilacak dan mencegah hasil lama dianggap sebagai hasil dari parameter baru. Setiap langkah perlu meninggalkan bukti yang dapat dibaca kembali: nilai parameter, versi pustaka, status proses, nama berkas, serta pesan kesalahan bila terjadi kegagalan.

Aspek validasi	Kriteria lulus
Input	Tipe, bentuk, satuan, muatan, spin, dan jalur berkas dinyatakan eksplisit.
Lingkungan	Versi Python dan pustaka tercatat; backend CPU/GPU terdeteksi.

Proses	Status eksekusi atau konvergensi diperiksa sebelum hasil dipakai.
Numerik	Tidak ada NaN/inf; ukuran array dan jumlah atom konsisten.
Keluaran	Energi, geometri, orbital, dan sifat molekul disimpan dan dapat dibuka kembali.
Interpretasi	Klaim tidak melampaui metode, basis set, atau data yang dihitung.

Interpretasi aman: Sebutkan metode, parameter, satuan, status konvergensi, dan keterbatasan saat menuliskan kesimpulan.

Kesalahan Umum dan Troubleshooting

Risiko utama pada topik ini adalah SCF tidak konvergen, spin keliru, atau interpretasi lintas metode. Karena itu, notebook harus memuat pemeriksaan eksplisit dan bukan hanya mengandalkan keluaran yang tampak masuk akal. Pembaca dianjurkan menjalankan notebook dari keadaan runtime bersih, membandingkan hasil dengan batas kewajaran, dan menyimpan artefak penting di lokasi persisten.

Gejala	Tindakan perbaikan
Sel dijalankan tidak berurutan	Restart session, lalu gunakan Run all dan periksa variabel prasyarat.
Pustaka tidak ditemukan	Jalankan kembali sel %pip, periksa nama paket, lalu restart bila dependensi inti berubah.
Input tampak benar tetapi gagal	Cetak tipe, shape, satuan, muatan, spin, dan beberapa baris pertama data.
Scf tidak konvergen, spin keliru, atau interpretasi lintas metode	Tambahkan validasi sebelum proses utama dan hentikan eksekusi dengan pesan yang spesifik.
Hasil berbeda antar-runtime	Catat versi runtime/paket, random seed, parameter, dan perangkat yang digunakan.

Aturan troubleshooting: Perbaiki satu hal pada satu waktu, simpan pesan kesalahan lengkap, dan catat perubahan versi atau parameter.

Latihan Modifikasi dan Refleksi

497. Ubah satu parameter pada contoh Interpretasi kritis dan keterbatasan, jalankan ulang dari runtime bersih, dan jelaskan perubahan keluaran.
498. Tambahkan satu validasi baru yang mencegah SCF tidak konvergen, spin keliru, atau interpretasi lintas metode.
499. Buat berkas ringkasan TXT atau JSON yang memuat input, status proses, waktu, dan hasil utama.
500. Jelaskan bagian mana yang merupakan fakta keluaran program dan bagian mana yang merupakan interpretasi ilmiah.
501. Susun satu kasus uji yang sengaja salah, lalu pastikan notebook menghasilkan pesan kesalahan yang informatif.

Refleksi: Tuliskan apa yang berubah setelah kode dimodifikasi, bukti apa yang menunjukkan keberhasilan, dan bagian mana yang masih memerlukan verifikasi eksternal.

Rubrik	Deskripsi
Skor 4	Kode berjalan dari runtime bersih, validasi lengkap, interpretasi tepat.
Skor 3	Kode berjalan dan hasil tersimpan, tetapi dokumentasi belum lengkap.
Skor 2	Kode hanya berjalan setelah intervensi atau state tertentu.
Skor 1	Kode belum berjalan dan diagnosis kesalahan belum sistematis.

Rangkuman Bab

Bab ini memperkenalkan perhitungan struktur elektronik dengan PySCF. Contoh H₂ dan H₂O digunakan untuk menjelaskan HF, DFT, basis set, optimasi geometri, energi, frekuensi, orbital, dan dipol.

- Energi potensial dan optimasi geometri
- Metode HF dan DFT
- Basis set dan parameter numerik
- Instalasi, struktur awal, dan energi awal

- Optimasi B3LYP/def2-TZVP
- Frekuensi, geometri, dan visualisasi hasil
- Energi dan konvergensi
- HOMO, LUMO, celah energi, dan momen dipol
- Interpretasi kritis dan keterbatasan

Inti bab: Keberhasilan teknis harus selalu disertai validasi, dokumentasi, dan interpretasi yang proporsional.

Proyek Praktik Terpadu

Proyek terpadu menggabungkan seluruh keterampilan bab dalam satu notebook. Gunakan satu sel konfigurasi, fungsi modular, validasi, logging sederhana, penyimpanan hasil, dan bagian pembahasan.

502. Tentukan pertanyaan atau tujuan komputasi.
503. Siapkan input dan parameter.
504. Jalankan proses utama.
505. Validasi status dan nilai.
506. Buat satu visualisasi atau tabel.
507. Ekspor hasil dan tulis pembahasan.
508. Restart session dan Run all.

Produk akhir: Notebook .ipynb, satu folder hasil, arsip ZIP, dan ringkasan pengujian.

Pertanyaan Analitis

509. Mengapa kode yang menghasilkan angka belum tentu menghasilkan bukti ilmiah?
510. Bagaimana versi runtime dapat memengaruhi reproduksibilitas?
511. Apa perbedaan pemeriksaan sintaks, pemeriksaan numerik, dan validasi kimia?
512. Kapan fallback dapat diterima, dan bagaimana harus dilaporkan?
513. Apa batas interpretasi dari metode atau contoh pada bab ini?
514. Bagaimana merancang notebook agar mudah direview oleh orang lain?

Panduan Jawaban dan Rubrik

Komponen	Bobot dan indikator
Ketepatan konsep	25% - definisi, asumsi, dan satuan benar.
Kualitas kode	25% - modular, terbaca, dan menangani kesalahan.

Validasi	25% - input, proses, dan keluaran diperiksa.
Interpretasi	15% - klaim sesuai bukti dan keterbatasan.
Reproduksibilitas	10% - dapat dijalankan dari runtime bersih.

Panduan jawaban menilai alur penalaran, bukan hanya keluaran akhir. Perbedaan nilai kecil dapat diterima bila versi atau perangkat berbeda dan semua parameter dilaporkan.

Daftar Periksa Penguasaan Bab

Pernyataan	Cek
Saya dapat menjelaskan konsep utama	☐
Saya dapat menjalankan kode dari runtime bersih	☐
Saya memeriksa status dan nilai numerik	☐
Saya menyimpan hasil di lokasi persisten	☐
Saya dapat menjelaskan kesalahan umum	☐
Saya menuliskan batas interpretasi	☐
Saya mencatat versi dan waktu pengujian	☐

Tindak lanjut: Ulangi halaman yang belum dikuasai, lalu kerjakan proyek bab tanpa melihat kode contoh secara langsung.

5

GPU4PySCF, Validasi, dan Proyek Lengkap Parasetamol

BAB 5 - GPU4PYSCF, VALIDASI, DAN PROYEK LENGKAP PARASETAMOL

Bab ini membahas penggunaan GPU secara selektif, validasi notebook, dan proyek parasetamol. Fokus utama adalah kompatibilitas, fallback CPU, pelaporan pengujian, dan interpretasi yang jujur.

Arah Bab: Setiap konsep diikuti prosedur, kode, penjelasan, validasi, troubleshooting, latihan, dan refleksi. Keluaran numerik tidak boleh dipisahkan dari konteks metode dan batas penerapannya.

Komponen	Kriteria
Orientasi	Memahami konsep dan istilah.
Praktik	Menjalankan kode secara bertahap.
Validasi	Memastikan input, proses, dan keluaran sah.
Interpretasi	Menjelaskan hasil tanpa klaim berlebihan.

Capaian Pembelajaran dan Prasyarat

- Menjelaskan konsep utama bab dengan istilah yang tepat.
- Menjalankan notebook dari runtime bersih dan mendokumentasikan versi.
- Memodifikasi parameter tanpa merusak alur validasi.
- Menyimpan hasil dalam format yang dapat dibuka kembali.
- Membedakan keluaran program, interpretasi, dan keterbatasan.

Prasyarat: Kemampuan menggunakan browser, akun Google, operasi berkas dasar, dan kemauan membaca pesan kesalahan secara teliti.

Komponen	Kriteria
Pengetahuan	Konsep sebelumnya dan satuan dasar.
Keterampilan	Menyalin kode, menjalankan sel, dan membaca output.
Sikap ilmiah	Jujur terhadap status pengujian dan ketidakpastian.
Perangkat	Browser modern dan koneksi internet stabil.

Peta Konsep dan Alur Kerja

Subbab	Alur konsep
5.1 Akselerasi GPU di Google Colab	Mengaktifkan dan mendeteksi GPU → Instalasi GPU4PySCF yang kompatibel → Perbandingan CPU, GPU, dan mode otomatis
5.2 Validasi Notebook dan Reproduksiabilitas	Validasi format dan sintaks → Validasi kimia dan numerik → Dokumentasi versi dan pelaporan pengujian
5.3 Proyek Akhir: Optimasi dan Analisis Parasetamol	Pembuatan struktur awal parasetamol → Optimasi DFT B3LYP/def2-TZVP → Analisis, visualisasi, dan ekspor seluruh hasil

Peta konsep dibaca dari kiri ke kanan dan dari input menuju keluaran. Hubungan antarkomponen tidak selalu linear: validasi dapat mengembalikan pembaca ke tahap input, sedangkan troubleshooting dapat memerlukan instalasi ulang atau penggantian metode.

Prinsip alur: Notebook yang baik memperlihatkan jalur keputusan: apa yang diperiksa, kapan proses dihentikan, kapan fallback digunakan, dan berkas apa yang disimpan.

Istilah Kunci dan Konvensi

Istilah	Konvensi penggunaan

notebook	Maknanya dibedakan dari istilah sehari-hari yang mirip.
runtime	Maknanya dibedakan dari istilah sehari-hari yang mirip.
kernel	Istilah digunakan secara konsisten dan dijelaskan saat pertama muncul.
sel Markdown	Nilai atau statusnya dicatat dalam notebook dan laporan.
sel kode	Nilai atau statusnya dicatat dalam notebook dan laporan.
parameter	Istilah digunakan secara konsisten dan dijelaskan saat pertama muncul.
validasi	Nilai atau statusnya dicatat dalam notebook dan laporan.
konvergensi	Istilah digunakan secara konsisten dan dijelaskan saat pertama muncul.
Konvensi satuan: Panjang molekul dinyatakan dalam Angstrom, energi elektronik dalam Hartree, energi orbital dapat dikonversi ke eV, dan perubahan energi dapat dilaporkan dalam kJ mol ⁻¹ .	

Asesmen Diagnostik Awal

515. Apa perbedaan notebook, runtime, dan berkas .ipynb?
516. Mengapa menjalankan sel tidak berurutan dapat menghasilkan keluaran yang menyesatkan?
517. Kapan sebuah nilai numerik boleh ditafsirkan sebagai hasil yang sah?
518. Informasi lingkungan apa yang perlu dicatat agar pekerjaan dapat diulang?
519. Bagaimana membedakan kegagalan instalasi, kegagalan input, dan kegagalan konvergensi?
520. Berkas apa yang harus disimpan sebelum runtime Colab terputus?

Cara menggunakan asesmen: Jawab sebelum memulai bab, tandai tingkat keyakinan, lalu ulangi setelah menyelesaikan proyek bab. Perubahan alasan lebih penting daripada perubahan skor saja.

5.1 Akselerasi GPU di Google Colab

Akselerasi GPU di Google Colab mengorganisasi tiga komponen yang saling bergantung. Pembahasan bergerak dari konsep, menuju prosedur, kemudian menghasilkan artefak yang dapat divalidasi dan disimpan.

Sub-subbab	Fokus
5.1.1 Mengaktifkan dan mendeteksi GPU	Pemilihan akselerator, penggunaan nvidia-smi, versi driver, CUDA, CuPy, dan cuTENSOR.
5.1.2 Instalasi GPU4PySCF yang kompatibel	Pemilihan paket sesuai versi CUDA dan larangan mencampur dependensi secara sembarangan.
5.1.3 Perbandingan CPU, GPU, dan mode otomatis	Benchmark waktu, penggunaan memori, dukungan metode, fallback CPU, dan penanganan GPU yang tidak tersedia.
Produk belajar: Satu notebook terstruktur, satu berkas keluaran, satu catatan validasi, dan satu interpretasi singkat yang menyatakan batas hasil.	

Rencana Praktik dan Produk Belajar

521. Pelajari Mengaktifkan dan mendeteksi GPU dan jalankan kode dasar yang menyertainya.
522. Pelajari Instalasi GPU4PySCF yang kompatibel dan jalankan kode dasar yang menyertainya.
523. Pelajari Perbandingan CPU, GPU, dan mode otomatis dan jalankan kode dasar yang menyertainya.

Komponen	Kriteria
Waktu	60-120 menit, bergantung pada komputasi.
Input	Notebook baru atau salinan notebook bab.
Produk	Kode, output, berkas hasil, dan refleksi.
Uji	Restart session lalu Run all.

Dokumentasi

Versi, parameter, status, dan waktu eksekusi.

Kriteria selesai: Semua sel dapat dijalankan berurutan, pesan kesalahan ditangani, dan hasil utama tersimpan di lokasi persisten.

5.1.1 Mengaktifkan dan mendeteksi GPU

Mengaktifkan dan mendeteksi GPU merupakan simpul penting dalam alur kerja notebook ilmiah karena menentukan bagaimana backend komputasi disiapkan, dijalankan, dan diperiksa. Pemilihan akselerator, penggunaan nvidia-smi, versi driver, CUDA, CuPy, dan cuTENSOR. Hasil yang dituju berupa informasi perangkat, waktu eksekusi, dan status penggunaan GPU.

Prinsip kerja yang digunakan adalah memisahkan input, proses, validasi, dan ekspor. Pemisahan ini membuat kesalahan lebih mudah dilacak dan mencegah hasil lama dianggap sebagai hasil dari parameter baru. Setiap langkah perlu meninggalkan bukti yang dapat dibaca kembali: nilai parameter, versi pustaka, status proses, nama berkas, serta pesan kesalahan bila terjadi kegagalan.

Tujuan operasional: Pembaca mampu menjelaskan Mengaktifkan dan mendeteksi GPU, menjalankan contoh minimal, memeriksa keberhasilan, dan menyimpan bukti hasil.

Komponen	Kriteria
Konsep	Pemilihan akselerator, penggunaan nvidia-smi, versi driver, CUDA, CuPy, dan cuTENSOR.
Input	Parameter dan data dinyatakan eksplisit.
Proses	Langkah dapat diulang dari runtime bersih.
Output	Nilai, berkas, atau visualisasi yang dapat diperiksa.
Batas	Interpretasi mengikuti metode dan konteks.

Landasan Teori dan Batas Penerapan

Pada bagian Mengaktifkan dan mendeteksi GPU, perhatian tidak hanya diarahkan pada keberhasilan menjalankan kode, tetapi juga pada ketepatan asumsi, keterlacakan parameter, dan mutu keluaran. Pemilihan akselerator, penggunaan nvidia-smi, versi

driver, CUDA, CuPy, dan cuTENSOR. Hasil yang dituju berupa informasi perangkat, waktu eksekusi, dan status penggunaan GPU.

Prinsip kerja yang digunakan adalah memisahkan input, proses, validasi, dan ekspor. Pemisahan ini membuat kesalahan lebih mudah dilacak dan mencegah hasil lama dianggap sebagai hasil dari parameter baru. Setiap langkah perlu meninggalkan bukti yang dapat dibaca kembali: nilai parameter, versi pustaka, status proses, nama berkas, serta pesan kesalahan bila terjadi kegagalan.

Risiko utama pada topik ini adalah paket CUDA tidak kompatibel atau GPU tidak benar-benar digunakan. Karena itu, notebook harus memuat pemeriksaan eksplisit dan bukan hanya mengandalkan keluaran yang tampak masuk akal. Pembaca dianjurkan menjalankan notebook dari keadaan runtime bersih, membandingkan hasil dengan batas kewajaran, dan menyimpan artefak penting di lokasi persisten.

Batas penerapan: Contoh pada buku berfungsi sebagai pola kerja. Ukuran molekul, biaya komputasi, versi pustaka, dan perangkat keras dapat mengubah waktu serta strategi yang diperlukan.

Prosedur Bertahap

Mengaktifkan dan mendeteksi GPU merupakan simpul penting dalam alur kerja notebook ilmiah karena menentukan bagaimana backend komputasi disiapkan, dijalankan, dan diperiksa. Pemilihan akselerator, penggunaan nvidia-smi, versi driver, CUDA, CuPy, dan cuTENSOR. Hasil yang dituju berupa informasi perangkat, waktu eksekusi, dan status penggunaan GPU.

524. Tuliskan tujuan operasional Mengaktifkan dan mendeteksi GPU dan tentukan keluaran yang harus tersedia pada akhir sel.
525. Tetapkan parameter pada satu sel konfigurasi agar perubahan dapat ditelusuri dan tidak tersebar di banyak tempat.
526. Siapkan backend komputasi dengan input yang eksplisit; hindari nilai tersembunyi yang hanya tersimpan dalam state kernel.
527. Jalankan pemeriksaan prasyarat, termasuk keberadaan pustaka, bentuk data, tipe nilai, serta konsistensi satuan.
528. Eksekusi proses utama, tangkap status konvergensi atau keberhasilan, dan hentikan alur apabila syarat minimum tidak terpenuhi.
529. Simpan informasi perangkat, waktu eksekusi, dan status penggunaan GPU dengan nama berkas informatif, kemudian tulis interpretasi yang membedakan fakta komputasi dan kesimpulan ilmiah.

Titik kontrol: Jangan melanjutkan ke interpretasi apabila validasi input atau status proses belum memenuhi kriteria.

Kode Dasar Siap Colab

Pembahasan Mengaktifkan dan mendeteksi GPU ditempatkan dalam konteks praktik Google Colab agar pembaca dapat menghubungkan konsep, prosedur, kode, dan interpretasi secara utuh. Pemilihan akselerator, penggunaan nvidia-smi, versi driver, CUDA, CuPy, dan cuTENSOR. Hasil yang dituju berupa informasi perangkat, waktu eksekusi, dan status penggunaan GPU.

Sel kode contoh

```
import shutil, subprocess

if shutil.which("nvidia-smi"):
    subprocess.run(["nvidia-smi"], check=False)
    DEVICE = "gpu"
else:
    DEVICE = "cpu"
print("Perangkat terpilih:", DEVICE)

# Runtime GPU tidak menjamin pustaka menghitung di GPU.
```

Cara menjalankan: Tempatkan kode dalam satu atau beberapa sel sesuai komentar. Untuk baris %pip atau perintah shell, gunakan runtime Google Colab.

Penjelasan Baris Kode Penting

Potongan kode	Fungsi
import shutil, subprocess	Mendefinisikan input atau parameter yang akan digunakan pada langkah berikutnya.
if shutil.which("nvidia-smi"):	Mencetak status agar pengguna tidak hanya bergantung pada keluaran visual.
subprocess.run(["nvidia-smi"], check=False)	Mendefinisikan input atau parameter yang akan digunakan pada langkah berikutnya.
DEVICE = "gpu"	Menambahkan pemeriksaan untuk mencegah hasil tidak valid diteruskan.
else:	Menjalankan operasi utama dan menyimpan hasil pada variabel yang jelas.
DEVICE = "cpu"	Mendefinisikan input atau parameter yang akan digunakan pada langkah berikutnya.

<code>print("Perangkat terpilih:", DEVICE)</code>	Mendefinisikan input atau parameter yang akan digunakan pada langkah berikutnya.
<code># Runtime GPU tidak menjamin pustaka menghitung</code>	Menambahkan pemeriksaan untuk mencegah hasil tidak valid diteruskan.

Nama variabel dipilih agar maknanya terbaca. Hindari singkatan yang tidak dijelaskan, nilai konstan tanpa satuan, serta penggunaan variabel dari sel lama. Komentar harus menerangkan alasan atau asumsi, bukan sekadar mengulang sintaks.

Validasi dan Interpretasi Hasil

Prinsip kerja yang digunakan adalah memisahkan input, proses, validasi, dan ekspor. Pemisahan ini membuat kesalahan lebih mudah dilacak dan mencegah hasil lama dianggap sebagai hasil dari parameter baru. Setiap langkah perlu meninggalkan bukti yang dapat dibaca kembali: nilai parameter, versi pustaka, status proses, nama berkas, serta pesan kesalahan bila terjadi kegagalan.

Aspek validasi	Kriteria lulus
Input	Tipe, bentuk, satuan, muatan, spin, dan jalur berkas dinyatakan eksplisit.
Lingkungan	Versi Python dan pustaka tercatat; backend CPU/GPU terdeteksi.
Proses	Status eksekusi atau konvergensi diperiksa sebelum hasil dipakai.
Numerik	Tidak ada NaN/inf; ukuran array dan jumlah atom konsisten.
Keluaran	Informasi perangkat, waktu eksekusi, dan status penggunaan gpu disimpan dan dapat dibuka kembali.
Interpretasi	Klaim tidak melampaui metode, basis set, atau data yang dihitung.
Interpretasi aman: Sebutkan metode, parameter, satuan, status konvergensi, dan keterbatasan saat menuliskan kesimpulan.	

Kesalahan Umum dan Troubleshooting

Risiko utama pada topik ini adalah paket CUDA tidak kompatibel atau GPU tidak benar-benar digunakan. Karena itu, notebook harus memuat pemeriksaan eksplisit dan bukan hanya mengandalkan keluaran yang tampak masuk akal. Pembaca dianjurkan menjalankan notebook dari keadaan runtime bersih, membandingkan hasil dengan batas kewajaran, dan menyimpan artefak penting di lokasi persisten.

Gejala	Tindakan perbaikan
Sel dijalankan tidak berurutan	Restart session, lalu gunakan Run all dan periksa variabel prasyarat.
Pustaka tidak ditemukan	Jalankan kembali sel %pip, periksa nama paket, lalu restart bila dependensi inti berubah.
Input tampak benar tetapi gagal	Cetak tipe, shape, satuan, muatan, spin, dan beberapa baris pertama data.
Paket cuda tidak kompatibel atau gpu tidak benar-benar digunakan	Tambahkan validasi sebelum proses utama dan hentikan eksekusi dengan pesan yang spesifik.
Hasil berbeda antar-runtime	Catat versi runtime/paket, random seed, parameter, dan perangkat yang digunakan.
Aturan troubleshooting: Perbaiki satu hal pada satu waktu, simpan pesan kesalahan lengkap, dan catat perubahan versi atau parameter.	

Latihan Modifikasi dan Refleksi

530. Ubah satu parameter pada contoh Mengaktifkan dan mendeteksi GPU, jalankan ulang dari runtime bersih, dan jelaskan perubahan keluaran.
531. Tambahkan satu validasi baru yang mencegah paket CUDA tidak kompatibel atau GPU tidak benar-benar digunakan.
532. Buat berkas ringkasan TXT atau JSON yang memuat input, status proses, waktu, dan hasil utama.
533. Jelaskan bagian mana yang merupakan fakta keluaran program dan bagian mana yang merupakan interpretasi ilmiah.
534. Susun satu kasus uji yang sengaja salah, lalu pastikan notebook menghasilkan pesan kesalahan yang informatif.

Refleksi: Tuliskan apa yang berubah setelah kode dimodifikasi, bukti apa yang menunjukkan keberhasilan, dan bagian mana yang masih memerlukan verifikasi eksternal.

Rubrik	Deskripsi
Skor 4	Kode berjalan dari runtime bersih, validasi lengkap, interpretasi tepat.
Skor 3	Kode berjalan dan hasil tersimpan, tetapi dokumentasi belum lengkap.
Skor 2	Kode hanya berjalan setelah intervensi atau state tertentu.
Skor 1	Kode belum berjalan dan diagnosis kesalahan belum sistematis.

5.1.2 Instalasi GPU4PySCF yang kompatibel

Instalasi GPU4PySCF yang kompatibel merupakan simpul penting dalam alur kerja notebook ilmiah karena menentukan bagaimana perhitungan struktur elektronik disiapkan, dijalankan, dan diperiksa. Pemilihan paket sesuai versi CUDA dan larangan mencampur dependensi secara sembarangan. Hasil yang dituju berupa energi, geometri, orbital, dan sifat molekul.

Prinsip kerja yang digunakan adalah memisahkan input, proses, validasi, dan ekspor. Pemisahan ini membuat kesalahan lebih mudah dilacak dan mencegah hasil lama dianggap sebagai hasil dari parameter baru. Setiap langkah perlu meninggalkan bukti yang dapat dibaca kembali: nilai parameter, versi pustaka, status proses, nama berkas, serta pesan kesalahan bila terjadi kegagalan.

Tujuan operasional: Pembaca mampu menjelaskan Instalasi GPU4PySCF yang kompatibel, menjalankan contoh minimal, memeriksa keberhasilan, dan menyimpan bukti hasil.

Komponen	Kriteria
Konsep	Pemilihan paket sesuai versi CUDA dan larangan mencampur dependensi secara sembarangan.
Input	Parameter dan data dinyatakan eksplisit.

Proses	Langkah dapat diulang dari runtime bersih.
Output	Nilai, berkas, atau visualisasi yang dapat diperiksa.
Batas	Interpretasi mengikuti metode dan konteks.

Landasan Teori dan Batas Penerapan

Pembahasan Instalasi GPU4PySCF yang kompatibel ditempatkan dalam konteks praktik Google Colab agar pembaca dapat menghubungkan konsep, prosedur, kode, dan interpretasi secara utuh. Pemilihan paket sesuai versi CUDA dan larangan mencampur dependensi secara sembarangan. Hasil yang dituju berupa energi, geometri, orbital, dan sifat molekul.

Prinsip kerja yang digunakan adalah memisahkan input, proses, validasi, dan ekspor. Pemisahan ini membuat kesalahan lebih mudah dilacak dan mencegah hasil lama dianggap sebagai hasil dari parameter baru. Setiap langkah perlu meninggalkan bukti yang dapat dibaca kembali: nilai parameter, versi pustaka, status proses, nama berkas, serta pesan kesalahan bila terjadi kegagalan.

Risiko utama pada topik ini adalah SCF tidak konvergen, spin keliru, atau interpretasi lintas metode. Karena itu, notebook harus memuat pemeriksaan eksplisit dan bukan hanya mengandalkan keluaran yang tampak masuk akal. Pembaca dianjurkan menjalankan notebook dari keadaan runtime bersih, membandingkan hasil dengan batas kewajaran, dan menyimpan artefak penting di lokasi persisten.

Batas penerapan: Contoh pada buku berfungsi sebagai pola kerja. Ukuran molekul, biaya komputasi, versi pustaka, dan perangkat keras dapat mengubah waktu serta strategi yang diperlukan.

Prosedur Bertahap

Pembahasan Instalasi GPU4PySCF yang kompatibel ditempatkan dalam konteks praktik Google Colab agar pembaca dapat menghubungkan konsep, prosedur, kode, dan interpretasi secara utuh. Pemilihan paket sesuai versi CUDA dan larangan mencampur dependensi secara sembarangan. Hasil yang dituju berupa energi, geometri, orbital, dan sifat molekul.

535. Tuliskan tujuan operasional Instalasi GPU4PySCF yang kompatibel dan tentukan keluaran yang harus tersedia pada akhir sel.
536. Tetapkan parameter pada satu sel konfigurasi agar perubahan dapat ditelusuri dan tidak tersebar di banyak tempat.

537. Siapkan perhitungan struktur elektronik dengan input yang eksplisit; hindari nilai tersembunyi yang hanya tersimpan dalam state kernel.
538. Jalankan pemeriksaan prasyarat, termasuk keberadaan pustaka, bentuk data, tipe nilai, serta konsistensi satuan.
539. Eksekusi proses utama, tangkap status konvergensi atau keberhasilan, dan hentikan alur apabila syarat minimum tidak terpenuhi.
540. Simpan energi, geometri, orbital, dan sifat molekul dengan nama berkas informatif, kemudian tulis interpretasi yang membedakan fakta komputasi dan kesimpulan ilmiah.

Titik kontrol: Jangan melanjutkan ke interpretasi apabila validasi input atau status proses belum memenuhi kriteria.

Kode Dasar Siap Colab

Instalasi GPU4PySCF yang kompatibel merupakan simpul penting dalam alur kerja notebook ilmiah karena menentukan bagaimana perhitungan struktur elektronik disiapkan, dijalankan, dan diperiksa. Pemilihan paket sesuai versi CUDA dan larangan mencampur dependensi secara sembarangan. Hasil yang dituju berupa energi, geometri, orbital, dan sifat molekul.

Sel kode contoh

```
# Deteksi versi CUDA terlebih dahulu dengan nvidia-smi.  
# Contoh paket biner resmi yang terdokumentasi:  
%pip -q install gpu4pyscf-cuda12x cutensor-cu12  
  
import gpu4pyscf  
print("GPU4PySCF berhasil diimpor")  
  
# Jangan memasang paket cuda11x dan cuda12x dalam runtime yang sama.
```

Cara menjalankan: Tempatkan kode dalam satu atau beberapa sel sesuai komentar. Untuk baris %pip atau perintah shell, gunakan runtime Google Colab.

Penjelasan Baris Kode Penting

Potongan kode	Fungsi
# Deteksi versi CUDA terlebih dahulu dengan nvidia-smi	Mendefinisikan input atau parameter yang akan digunakan pada langkah berikutnya.
# Contoh paket biner resmi yang terdokumentasi:	Menjalankan operasi utama dan menyimpan hasil pada variabel yang jelas.

<code>%pip -q install gpu4pyscf-cuda12x cutensor-cu12</code>	Menambahkan pemeriksaan untuk mencegah hasil tidak valid diteruskan.
<code>import gpu4pyscf</code>	Menjalankan operasi utama dan menyimpan hasil pada variabel yang jelas.
<code>print("GPU4PySCF berhasil diimpor")</code>	Mendefinisikan input atau parameter yang akan digunakan pada langkah berikutnya.
<code># Jangan memasang paket cuda11x dan cuda12x dala</code>	Mendefinisikan input atau parameter yang akan digunakan pada langkah berikutnya.

Nama variabel dipilih agar maknanya terbaca. Hindari singkatan yang tidak dijelaskan, nilai konstan tanpa satuan, serta penggunaan variabel dari sel lama. Komentar harus menerangkan alasan atau asumsi, bukan sekadar mengulang sintaks.

Validasi dan Interpretasi Hasil

Prinsip kerja yang digunakan adalah memisahkan input, proses, validasi, dan ekspor. Pemisahan ini membuat kesalahan lebih mudah dilacak dan mencegah hasil lama dianggap sebagai hasil dari parameter baru. Setiap langkah perlu meninggalkan bukti yang dapat dibaca kembali: nilai parameter, versi pustaka, status proses, nama berkas, serta pesan kesalahan bila terjadi kegagalan.

Aspek validasi	Kriteria lulus
Input	Tipe, bentuk, satuan, muatan, spin, dan jalur berkas dinyatakan eksplisit.
Lingkungan	Versi Python dan pustaka tercatat; backend CPU/GPU terdeteksi.
Proses	Status eksekusi atau konvergensi diperiksa sebelum hasil dipakai.
Numerik	Tidak ada NaN/inf; ukuran array dan jumlah atom konsisten.
Keluaran	Energi, geometri, orbital, dan sifat molekul disimpan dan dapat dibuka kembali.

Interpretasi

Klaim tidak melampaui metode, basis set, atau data yang dihitung.

Interpretasi aman: Sebutkan metode, parameter, satuan, status konvergensi, dan keterbatasan saat menuliskan kesimpulan.

Kesalahan Umum dan Troubleshooting

Risiko utama pada topik ini adalah SCF tidak konvergen, spin keliru, atau interpretasi lintas metode. Karena itu, notebook harus memuat pemeriksaan eksplisit dan bukan hanya mengandalkan keluaran yang tampak masuk akal. Pembaca dianjurkan menjalankan notebook dari keadaan runtime bersih, membandingkan hasil dengan batas kewajaran, dan menyimpan artefak penting di lokasi persisten.

Gejala	Tindakan perbaikan
Sel dijalankan tidak berurutan	Restart session, lalu gunakan Run all dan periksa variabel prasyarat.
Pustaka tidak ditemukan	Jalankan kembali sel %pip, periksa nama paket, lalu restart bila dependensi inti berubah.
Input tampak benar tetapi gagal	Cetak tipe, shape, satuan, muatan, spin, dan beberapa baris pertama data.
Scf tidak konvergen, spin keliru, atau interpretasi lintas metode	Tambahkan validasi sebelum proses utama dan hentikan eksekusi dengan pesan yang spesifik.
Hasil berbeda antar-runtime	Catat versi runtime/paket, random seed, parameter, dan perangkat yang digunakan.
Aturan troubleshooting: Perbaiki satu hal pada satu waktu, simpan pesan kesalahan lengkap, dan catat perubahan versi atau parameter.	

Latihan Modifikasi dan Refleksi

541. Ubah satu parameter pada contoh Instalasi GPU4PySCF yang kompatibel, jalankan ulang dari runtime bersih, dan jelaskan perubahan keluaran.
542. Tambahkan satu validasi baru yang mencegah SCF tidak konvergen, spin keliru, atau interpretasi lintas metode.

543. Buat berkas ringkasan TXT atau JSON yang memuat input, status proses, waktu, dan hasil utama.
544. Jelaskan bagian mana yang merupakan fakta keluaran program dan bagian mana yang merupakan interpretasi ilmiah.
545. Susun satu kasus uji yang sengaja salah, lalu pastikan notebook menghasilkan pesan kesalahan yang informatif.

Refleksi: Tuliskan apa yang berubah setelah kode dimodifikasi, bukti apa yang menunjukkan keberhasilan, dan bagian mana yang masih memerlukan verifikasi eksternal.

Rubrik	Deskripsi
Skor 4	Kode berjalan dari runtime bersih, validasi lengkap, interpretasi tepat.
Skor 3	Kode berjalan dan hasil tersimpan, tetapi dokumentasi belum lengkap.
Skor 2	Kode hanya berjalan setelah intervensi atau state tertentu.
Skor 1	Kode belum berjalan dan diagnosis kesalahan belum sistematis.

5.1.3 Perbandingan CPU, GPU, dan mode otomatis

Perbandingan CPU, GPU, dan mode otomatis merupakan simpul penting dalam alur kerja notebook ilmiah karena menentukan bagaimana backend komputasi disiapkan, dijalankan, dan diperiksa. Benchmark waktu, penggunaan memori, dukungan metode, fallback CPU, dan penanganan GPU yang tidak tersedia. Hasil yang dituju berupa informasi perangkat, waktu eksekusi, dan status penggunaan GPU.

Prinsip kerja yang digunakan adalah memisahkan input, proses, validasi, dan ekspor. Pemisahan ini membuat kesalahan lebih mudah dilacak dan mencegah hasil lama dianggap sebagai hasil dari parameter baru. Setiap langkah perlu meninggalkan bukti yang dapat dibaca kembali: nilai parameter, versi pustaka, status proses, nama berkas, serta pesan kesalahan bila terjadi kegagalan.

Tujuan operasional: Pembaca mampu menjelaskan Perbandingan CPU, GPU, dan mode otomatis, menjalankan contoh minimal, memeriksa keberhasilan, dan menyimpan bukti hasil.

Komponen	Kriteria

Konsep	Benchmark waktu, penggunaan memori, dukungan metode, fallback CPU, dan penanganan GPU yang tidak tersedia.
Input	Parameter dan data dinyatakan eksplisit.
Proses	Langkah dapat diulang dari runtime bersih.
Output	Nilai, berkas, atau visualisasi yang dapat diperiksa.
Batas	Interpretasi mengikuti metode dan konteks.

Landasan Teori dan Batas Penerapan

Pada bagian Perbandingan CPU, GPU, dan mode otomatis, perhatian tidak hanya diarahkan pada keberhasilan menjalankan kode, tetapi juga pada ketepatan asumsi, keterlacakan parameter, dan mutu keluaran. Benchmark waktu, penggunaan memori, dukungan metode, fallback CPU, dan penanganan GPU yang tidak tersedia. Hasil yang dituju berupa informasi perangkat, waktu eksekusi, dan status penggunaan GPU.

Prinsip kerja yang digunakan adalah memisahkan input, proses, validasi, dan ekspor. Pemisahan ini membuat kesalahan lebih mudah dilacak dan mencegah hasil lama dianggap sebagai hasil dari parameter baru. Setiap langkah perlu meninggalkan bukti yang dapat dibaca kembali: nilai parameter, versi pustaka, status proses, nama berkas, serta pesan kesalahan bila terjadi kegagalan.

Risiko utama pada topik ini adalah paket CUDA tidak kompatibel atau GPU tidak benar-benar digunakan. Karena itu, notebook harus memuat pemeriksaan eksplisit dan bukan hanya mengandalkan keluaran yang tampak masuk akal. Pembaca dianjurkan menjalankan notebook dari keadaan runtime bersih, membandingkan hasil dengan batas kewajaran, dan menyimpan artefak penting di lokasi persisten.

Batas penerapan: Contoh pada buku berfungsi sebagai pola kerja. Ukuran molekul, biaya komputasi, versi pustaka, dan perangkat keras dapat mengubah waktu serta strategi yang diperlukan.

Prosedur Bertahap

Pada bagian Perbandingan CPU, GPU, dan mode otomatis, perhatian tidak hanya diarahkan pada keberhasilan menjalankan kode, tetapi juga pada ketepatan asumsi, keterlacakan parameter, dan mutu keluaran. Benchmark waktu, penggunaan memori,

dukungan metode, fallback CPU, dan penanganan GPU yang tidak tersedia. Hasil yang dituju berupa informasi perangkat, waktu eksekusi, dan status penggunaan GPU.

546. Tuliskan tujuan operasional Perbandingan CPU, GPU, dan mode otomatis dan tentukan keluaran yang harus tersedia pada akhir sel.
547. Tetapkan parameter pada satu sel konfigurasi agar perubahan dapat ditelusuri dan tidak tersebar di banyak tempat.
548. Siapkan backend komputasi dengan input yang eksplisit; hindari nilai tersembunyi yang hanya tersimpan dalam state kernel.
549. Jalankan pemeriksaan prasyarat, termasuk keberadaan pustaka, bentuk data, tipe nilai, serta konsistensi satuan.
550. Eksekusi proses utama, tangkap status konvergensi atau keberhasilan, dan hentikan alur apabila syarat minimum tidak terpenuhi.
551. Simpan informasi perangkat, waktu eksekusi, dan status penggunaan GPU dengan nama berkas informatif, kemudian tulis interpretasi yang membedakan fakta komputasi dan kesimpulan ilmiah.

Titik kontrol: Jangan melanjutkan ke interpretasi apabila validasi input atau status proses belum memenuhi kriteria.

Kode Dasar Siap Colab

Perbandingan CPU, GPU, dan mode otomatis merupakan simpul penting dalam alur kerja notebook ilmiah karena menentukan bagaimana backend komputasi disiapkan, dijalankan, dan diperiksa. Benchmark waktu, penggunaan memori, dukungan metode, fallback CPU, dan penanganan GPU yang tidak tersedia. Hasil yang dituju berupa informasi perangkat, waktu eksekusi, dan status penggunaan GPU.

Sel kode contoh

```
from time import perf_counter

def timed(label, fn):
    t0 = perf_counter(); value = fn(); dt = perf_counter()-t0
    print(f"{label:8s}: {dt:10.3f} s")
    return value, dt

# cpu_value, cpu_time = timed("CPU", run_cpu)
# gpu_value, gpu_time = timed("GPU", run_gpu)
# Bandingkan energi dan toleransi numerik, bukan waktu saja.
```

Cara menjalankan: Tempatkan kode dalam satu atau beberapa sel sesuai komentar. Untuk baris %pip atau perintah shell, gunakan runtime Google Colab.

Penjelasan Baris Kode Penting

Potongan kode	Fungsi
<code>from time import perf_counter</code>	Mendefinisikan input atau parameter yang akan digunakan pada langkah berikutnya.
<code>def timed(label, fn):</code>	Menambahkan pemeriksaan untuk mencegah hasil tidak valid diteruskan.
<code>t0 = perf_counter(); value = fn(); dt = perf</code>	Mencetak status agar pengguna tidak hanya bergantung pada keluaran visual.
<code>print(f'{label:8s}: {dt:10.3f} s')</code>	Mencetak status agar pengguna tidak hanya bergantung pada keluaran visual.
<code>return value, dt</code>	Menambahkan pemeriksaan untuk mencegah hasil tidak valid diteruskan.
<code># cpu_value, cpu_time = timed("CPU", run_cpu)</code>	Mencetak status agar pengguna tidak hanya bergantung pada keluaran visual.
<code># gpu_value, gpu_time = timed("GPU", run_gpu)</code>	Menambahkan pemeriksaan untuk mencegah hasil tidak valid diteruskan.
<code># Bandingkan energi dan toleransi numerik, bukan</code>	Menambahkan pemeriksaan untuk mencegah hasil tidak valid diteruskan.

Nama variabel dipilih agar maknanya terbaca. Hindari singkatan yang tidak dijelaskan, nilai konstan tanpa satuan, serta penggunaan variabel dari sel lama. Komentar harus menerangkan alasan atau asumsi, bukan sekadar mengulang sintaks.

Validasi dan Interpretasi Hasil

Prinsip kerja yang digunakan adalah memisahkan input, proses, validasi, dan ekspor. Pemisahan ini membuat kesalahan lebih mudah dilacak dan mencegah hasil lama dianggap sebagai hasil dari parameter baru. Setiap langkah perlu meninggalkan bukti yang dapat dibaca kembali: nilai parameter, versi pustaka, status proses, nama berkas, serta pesan kesalahan bila terjadi kegagalan.

Aspek validasi**Kriteria lulus**

Input	Tipe, bentuk, satuan, muatan, spin, dan jalur berkas dinyatakan eksplisit.
Lingkungan	Versi Python dan pustaka tercatat; backend CPU/GPU terdeteksi.
Proses	Status eksekusi atau konvergensi diperiksa sebelum hasil dipakai.
Numerik	Tidak ada NaN/inf; ukuran array dan jumlah atom konsisten.
Keluaran	Informasi perangkat, waktu eksekusi, dan status penggunaan gpu disimpan dan dapat dibuka kembali.
Interpretasi	Klaim tidak melampaui metode, basis set, atau data yang dihitung.
Interpretasi aman: Sebutkan metode, parameter, satuan, status konvergensi, dan keterbatasan saat menuliskan kesimpulan.	

Kesalahan Umum dan Troubleshooting

Risiko utama pada topik ini adalah paket CUDA tidak kompatibel atau GPU tidak benar-benar digunakan. Karena itu, notebook harus memuat pemeriksaan eksplisit dan bukan hanya mengandalkan keluaran yang tampak masuk akal. Pembaca dianjurkan menjalankan notebook dari keadaan runtime bersih, membandingkan hasil dengan batas kewajaran, dan menyimpan artefak penting di lokasi persisten.

Gejala	Tindakan perbaikan
Sel dijalankan tidak berurutan	Restart session, lalu gunakan Run all dan periksa variabel prasyarat.
Pustaka tidak ditemukan	Jalankan kembali sel %pip, periksa nama paket, lalu restart bila dependensi inti berubah.
Input tampak benar tetapi gagal	Cetak tipe, shape, satuan, muatan, spin, dan beberapa baris pertama data.

Paket cuda tidak kompatibel atau gpu tidak benar-benar digunakan	Tambahkan validasi sebelum proses utama dan hentikan eksekusi dengan pesan yang spesifik.
Hasil berbeda antar-runtime	Catat versi runtime/paket, random seed, parameter, dan perangkat yang digunakan.
Aturan troubleshooting: Perbaiki satu hal pada satu waktu, simpan pesan kesalahan lengkap, dan catat perubahan versi atau parameter.	

Latihan Modifikasi dan Refleksi

552. Ubah satu parameter pada contoh Perbandingan CPU, GPU, dan mode otomatis, jalankan ulang dari runtime bersih, dan jelaskan perubahan keluaran.
553. Tambahkan satu validasi baru yang mencegah paket CUDA tidak kompatibel atau GPU tidak benar-benar digunakan.
554. Buat berkas ringkasan TXT atau JSON yang memuat input, status proses, waktu, dan hasil utama.
555. Jelaskan bagian mana yang merupakan fakta keluaran program dan bagian mana yang merupakan interpretasi ilmiah.
556. Susun satu kasus uji yang sengaja salah, lalu pastikan notebook menghasilkan pesan kesalahan yang informatif.

Refleksi: Tuliskan apa yang berubah setelah kode dimodifikasi, bukti apa yang menunjukkan keberhasilan, dan bagian mana yang masih memerlukan verifikasi eksternal.

Rubrik	Deskripsi
Skor 4	Kode berjalan dari runtime bersih, validasi lengkap, interpretasi tepat.
Skor 3	Kode berjalan dan hasil tersimpan, tetapi dokumentasi belum lengkap.
Skor 2	Kode hanya berjalan setelah intervensi atau state tertentu.
Skor 1	Kode belum berjalan dan diagnosis kesalahan belum sistematis.

5.2 Validasi Notebook dan Reproduksiabilitas

Validasi Notebook dan Reproduksiabilitas mengorganisasi tiga komponen yang saling bergantung. Pembahasan bergerak dari konsep, menuju prosedur, kemudian menghasilkan artefak yang dapat divalidasi dan disimpan.

Sub-subbab	Fokus
5.2.1 Validasi format dan sintaks	Pembuatan notebook menggunakan nbformat, pemeriksaan cell ID, nbformat.validate(), ast.parse(), dan kompilasi sel kode.
5.2.2 Validasi kimia dan numerik	Pemeriksaan unsur, basis set, elektron, spin, SCF, nilai finite, koordinat, serta konsistensi jumlah atom file XYZ.
5.2.3 Dokumentasi versi dan pelaporan pengujian	Pencatatan versi Python, pustaka, runtime, perangkat keras, waktu eksekusi, dan bagian yang benar-benar telah diuji.
Produk belajar: Satu notebook terstruktur, satu berkas keluaran, satu catatan validasi, dan satu interpretasi singkat yang menyatakan batas hasil.	

Rencana Praktik dan Produk Belajar

557. Pelajari Validasi format dan sintaks dan jalankan kode dasar yang menyertainya.
558. Pelajari Validasi kimia dan numerik dan jalankan kode dasar yang menyertainya.
559. Pelajari Dokumentasi versi dan pelaporan pengujian dan jalankan kode dasar yang menyertainya.

Komponen	Kriteria
Waktu	60-120 menit, bergantung pada komputasi.
Input	Notebook baru atau salinan notebook bab.
Produk	Kode, output, berkas hasil, dan refleksi.
Uji	Restart session lalu Run all.

Dokumentasi

Versi, parameter, status, dan waktu eksekusi.

Kriteria selesai: Semua sel dapat dijalankan berurutan, pesan kesalahan ditangani, dan hasil utama tersimpan di lokasi persisten.

5.2.1 Validasi format dan sintaks

Pembahasan Validasi format dan sintaks ditempatkan dalam konteks praktik Google Colab agar pembaca dapat menghubungkan konsep, prosedur, kode, dan interpretasi secara utuh. Pembuatan notebook menggunakan nbformat, pemeriksaan cell ID, nbformat.validate(), ast.parse(), dan kompilasi sel kode. Hasil yang dituju berupa sel Markdown, sel kode, output, dan metadata.

Prinsip kerja yang digunakan adalah memisahkan input, proses, validasi, dan ekspor. Pemisahan ini membuat kesalahan lebih mudah dilacak dan mencegah hasil lama dianggap sebagai hasil dari parameter baru. Setiap langkah perlu meninggalkan bukti yang dapat dibaca kembali: nilai parameter, versi pustaka, status proses, nama berkas, serta pesan kesalahan bila terjadi kegagalan.

Tujuan operasional: Pembaca mampu menjelaskan Validasi format dan sintaks, menjalankan contoh minimal, memeriksa keberhasilan, dan menyimpan bukti hasil.

Komponen	Kriteria
Konsep	Pembuatan notebook menggunakan nbformat, pemeriksaan cell ID, nbformat.validate(), ast.parse(), dan kompilasi sel kode.
Input	Parameter dan data dinyatakan eksplisit.
Proses	Langkah dapat diulang dari runtime bersih.
Output	Nilai, berkas, atau visualisasi yang dapat diperiksa.
Batas	Interpretasi mengikuti metode dan konteks.

Landasan Teori dan Batas Penerapan

Validasi format dan sintaks merupakan simpul penting dalam alur kerja notebook ilmiah karena menentukan bagaimana notebook disiapkan, dijalankan, dan diperiksa. Pembuatan notebook menggunakan nbformat, pemeriksaan cell ID, nbformat.validate(), ast.parse(), dan kompilasi sel kode. Hasil yang dituju berupa sel Markdown, sel kode, output, dan metadata.

Prinsip kerja yang digunakan adalah memisahkan input, proses, validasi, dan ekspor. Pemisahan ini membuat kesalahan lebih mudah dilacak dan mencegah hasil lama dianggap sebagai hasil dari parameter baru. Setiap langkah perlu meninggalkan bukti yang dapat dibaca kembali: nilai parameter, versi pustaka, status proses, nama berkas, serta pesan kesalahan bila terjadi kegagalan.

Risiko utama pada topik ini adalah state kernel tersembunyi dan urutan eksekusi. Karena itu, notebook harus memuat pemeriksaan eksplisit dan bukan hanya mengandalkan keluaran yang tampak masuk akal. Pembaca dianjurkan menjalankan notebook dari keadaan runtime bersih, membandingkan hasil dengan batas kewajaran, dan menyimpan artefak penting di lokasi persisten.

Batas penerapan: Contoh pada buku berfungsi sebagai pola kerja. Ukuran molekul, biaya komputasi, versi pustaka, dan perangkat keras dapat mengubah waktu serta strategi yang diperlukan.

Prosedur Bertahap

Validasi format dan sintaks merupakan simpul penting dalam alur kerja notebook ilmiah karena menentukan bagaimana notebook disiapkan, dijalankan, dan diperiksa. Pembuatan notebook menggunakan nbformat, pemeriksaan cell ID, nbformat.validate(), ast.parse(), dan kompilasi sel kode. Hasil yang dituju berupa sel Markdown, sel kode, output, dan metadata.

560. Tuliskan tujuan operasional Validasi format dan sintaks dan tentukan keluaran yang harus tersedia pada akhir sel.
561. Tetapkan parameter pada satu sel konfigurasi agar perubahan dapat ditelusuri dan tidak tersebar di banyak tempat.
562. Siapkan notebook dengan input yang eksplisit; hindari nilai tersembunyi yang hanya tersimpan dalam state kernel.
563. Jalankan pemeriksaan prasyarat, termasuk keberadaan pustaka, bentuk data, tipe nilai, serta konsistensi satuan.
564. Eksekusi proses utama, tangkap status konvergensi atau keberhasilan, dan hentikan alur apabila syarat minimum tidak terpenuhi.
565. Simpan sel Markdown, sel kode, output, dan metadata dengan nama berkas informatif, kemudian tulis interpretasi yang membedakan fakta komputasi dan kesimpulan ilmiah.

Titik kontrol: Jangan melanjutkan ke interpretasi apabila validasi input atau status proses belum memenuhi kriteria.

Kode Dasar Siap Colab

Pembahasan Validasi format dan sintaks ditempatkan dalam konteks praktik Google Colab agar pembaca dapat menghubungkan konsep, prosedur, kode, dan interpretasi secara utuh. Pembuatan notebook menggunakan nbformat, pemeriksaan cell ID, nbformat.validate(), ast.parse(), dan kompilasi sel kode. Hasil yang dituju berupa sel Markdown, sel kode, output, dan metadata.

Sel kode contoh

```
import ast, nbformat

nb = nbformat.v4.new_notebook()
nb.cells = [
    nbformat.v4.new_markdown_cell("# Notebook tervalidasi"),
    nbformat.v4.new_code_cell("x = 2\nprint(x**2)"),
]
nbformat.validate(nb)
for cell in nb.cells:
    if cell.cell_type == "code":
        ast.parse(cell.source)
nbformat.write(nb, "notebook_valid.ipynb")
```

Cara menjalankan: Tempatkan kode dalam satu atau beberapa sel sesuai komentar. Untuk baris %pip atau perintah shell, gunakan runtime Google Colab.

Penjelasan Baris Kode Penting

Potongan kode	Fungsi
import ast, nbformat	Menambahkan pemeriksaan untuk mencegah hasil tidak valid diteruskan.
nb = nbformat.v4.new_notebook()	Menjalankan operasi utama dan menyimpan hasil pada variabel yang jelas.
nb.cells = [	Menambahkan pemeriksaan untuk mencegah hasil tidak valid diteruskan.
nbformat.v4.new_markdown_cell("# Notebook te	Menambahkan pemeriksaan untuk mencegah hasil tidak valid diteruskan.

<code>nbformat.v4.new_code_cell("x = 2\nprint(x**2</code>	Mendefinisikan input atau parameter yang akan digunakan pada langkah berikutnya.
<code>]</code>	Mencetak status agar pengguna tidak hanya bergantung pada keluaran visual.
<code>nbformat.validate(nb)</code>	Mencetak status agar pengguna tidak hanya bergantung pada keluaran visual.
<code>for cell in nb.cells:</code>	Mencetak status agar pengguna tidak hanya bergantung pada keluaran visual.

Nama variabel dipilih agar maknanya terbaca. Hindari singkatan yang tidak dijelaskan, nilai konstan tanpa satuan, serta penggunaan variabel dari sel lama. Komentar harus menerangkan alasan atau asumsi, bukan sekadar mengulang sintaks.

Validasi dan Interpretasi Hasil

Prinsip kerja yang digunakan adalah memisahkan input, proses, validasi, dan ekspor. Pemisahan ini membuat kesalahan lebih mudah dilacak dan mencegah hasil lama dianggap sebagai hasil dari parameter baru. Setiap langkah perlu meninggalkan bukti yang dapat dibaca kembali: nilai parameter, versi pustaka, status proses, nama berkas, serta pesan kesalahan bila terjadi kegagalan.

Aspek validasi	Kriteria lulus
Input	Tipe, bentuk, satuan, muatan, spin, dan jalur berkas dinyatakan eksplisit.
Lingkungan	Versi Python dan pustaka tercatat; backend CPU/GPU terdeteksi.
Proses	Status eksekusi atau konvergensi diperiksa sebelum hasil dipakai.
Numerik	Tidak ada NaN/inf; ukuran array dan jumlah atom konsisten.
Keluaran	Sel markdown, sel kode, output, dan metadata disimpan dan dapat dibuka kembali.

Interpretasi

Klaim tidak melampaui metode, basis set, atau data yang dihitung.

Interpretasi aman: Sebutkan metode, parameter, satuan, status konvergensi, dan keterbatasan saat menuliskan kesimpulan.

Kesalahan Umum dan Troubleshooting

Risiko utama pada topik ini adalah state kernel tersembunyi dan urutan eksekusi. Karena itu, notebook harus memuat pemeriksaan eksplisit dan bukan hanya mengandalkan keluaran yang tampak masuk akal. Pembaca dianjurkan menjalankan notebook dari keadaan runtime bersih, membandingkan hasil dengan batas kewajaran, dan menyimpan artefak penting di lokasi persisten.

Gejala	Tindakan perbaikan
Sel dijalankan tidak berurutan	Restart session, lalu gunakan Run all dan periksa variabel prasyarat.
Pustaka tidak ditemukan	Jalankan kembali sel %pip, periksa nama paket, lalu restart bila dependensi inti berubah.
Input tampak benar tetapi gagal	Cetak tipe, shape, satuan, muatan, spin, dan beberapa baris pertama data.
State kernel tersembunyi dan urutan eksekusi	Tambahkan validasi sebelum proses utama dan hentikan eksekusi dengan pesan yang spesifik.
Hasil berbeda antar-runtime	Catat versi runtime/paket, random seed, parameter, dan perangkat yang digunakan.
Aturan troubleshooting: Perbaiki satu hal pada satu waktu, simpan pesan kesalahan lengkap, dan catat perubahan versi atau parameter.	

Latihan Modifikasi dan Refleksi

566. Ubah satu parameter pada contoh Validasi format dan sintaks, jalankan ulang dari runtime bersih, dan jelaskan perubahan keluaran.
567. Tambahkan satu validasi baru yang mencegah state kernel tersembunyi dan urutan eksekusi.

568. Buat berkas ringkasan TXT atau JSON yang memuat input, status proses, waktu, dan hasil utama.
569. Jelaskan bagian mana yang merupakan fakta keluaran program dan bagian mana yang merupakan interpretasi ilmiah.
570. Susun satu kasus uji yang sengaja salah, lalu pastikan notebook menghasilkan pesan kesalahan yang informatif.

Refleksi: Tuliskan apa yang berubah setelah kode dimodifikasi, bukti apa yang menunjukkan keberhasilan, dan bagian mana yang masih memerlukan verifikasi eksternal.

Rubrik	Deskripsi
Skor 4	Kode berjalan dari runtime bersih, validasi lengkap, interpretasi tepat.
Skor 3	Kode berjalan dan hasil tersimpan, tetapi dokumentasi belum lengkap.
Skor 2	Kode hanya berjalan setelah intervensi atau state tertentu.
Skor 1	Kode belum berjalan dan diagnosis kesalahan belum sistematis.

5.2.2 Validasi kimia dan numerik

Validasi kimia dan numerik merupakan simpul penting dalam alur kerja notebook ilmiah karena menentukan bagaimana artefak ilmiah disiapkan, dijalankan, dan diperiksa. Pemeriksaan unsur, basis set, elektron, spin, SCF, nilai finite, koordinat, serta konsistensi jumlah atom file XYZ. Hasil yang dituju berupa laporan validasi dan jejak versi.

Prinsip kerja yang digunakan adalah memisahkan input, proses, validasi, dan ekspor. Pemisahan ini membuat kesalahan lebih mudah dilacak dan mencegah hasil lama dianggap sebagai hasil dari parameter baru. Setiap langkah perlu meninggalkan bukti yang dapat dibaca kembali: nilai parameter, versi pustaka, status proses, nama berkas, serta pesan kesalahan bila terjadi kegagalan.

Tujuan operasional: Pembaca mampu menjelaskan Validasi kimia dan numerik, menjalankan contoh minimal, memeriksa keberhasilan, dan menyimpan bukti hasil.

Komponen	Kriteria

Konsep	Pemeriksaan unsur, basis set, elektron, spin, SCF, nilai finite, koordinat, serta konsistensi jumlah atom file XYZ.
Input	Parameter dan data dinyatakan eksplisit.
Proses	Langkah dapat diulang dari runtime bersih.
Output	Nilai, berkas, atau visualisasi yang dapat diperiksa.
Batas	Interpretasi mengikuti metode dan konteks.

Landasan Teori dan Batas Penerapan

Pada bagian Validasi kimia dan numerik, perhatian tidak hanya diarahkan pada keberhasilan menjalankan kode, tetapi juga pada ketepatan asumsi, keterlacakan parameter, dan mutu keluaran. Pemeriksaan unsur, basis set, elektron, spin, SCF, nilai finite, koordinat, serta konsistensi jumlah atom file XYZ. Hasil yang dituju berupa laporan validasi dan jejak versi.

Prinsip kerja yang digunakan adalah memisahkan input, proses, validasi, dan ekspor. Pemisahan ini membuat kesalahan lebih mudah dilacak dan mencegah hasil lama dianggap sebagai hasil dari parameter baru. Setiap langkah perlu meninggalkan bukti yang dapat dibaca kembali: nilai parameter, versi pustaka, status proses, nama berkas, serta pesan kesalahan bila terjadi kegagalan.

Risiko utama pada topik ini adalah notebook hanya berhasil pada state tertentu atau hasil tidak dapat diulang. Karena itu, notebook harus memuat pemeriksaan eksplisit dan bukan hanya mengandalkan keluaran yang tampak masuk akal. Pembaca dianjurkan menjalankan notebook dari keadaan runtime bersih, membandingkan hasil dengan batas kewajaran, dan menyimpan artefak penting di lokasi persisten.

Batas penerapan: Contoh pada buku berfungsi sebagai pola kerja. Ukuran molekul, biaya komputasi, versi pustaka, dan perangkat keras dapat mengubah waktu serta strategi yang diperlukan.

Prosedur Bertahap

Pada bagian Validasi kimia dan numerik, perhatian tidak hanya diarahkan pada keberhasilan menjalankan kode, tetapi juga pada ketepatan asumsi, keterlacakan parameter, dan mutu keluaran. Pemeriksaan unsur, basis set, elektron, spin, SCF, nilai

finite, koordinat, serta konsistensi jumlah atom file XYZ. Hasil yang dituju berupa laporan validasi dan jejak versi.

571. Tuliskan tujuan operasional Validasi kimia dan numerik dan tentukan keluaran yang harus tersedia pada akhir sel.
572. Tetapkan parameter pada satu sel konfigurasi agar perubahan dapat ditelusuri dan tidak tersebar di banyak tempat.
573. Siapkan artefak ilmiah dengan input yang eksplisit; hindari nilai tersembunyi yang hanya tersimpan dalam state kernel.
574. Jalankan pemeriksaan prasyarat, termasuk keberadaan pustaka, bentuk data, tipe nilai, serta konsistensi satuan.
575. Eksekusi proses utama, tangkap status konvergensi atau keberhasilan, dan hentikan alur apabila syarat minimum tidak terpenuhi.
576. Simpan laporan validasi dan jejak versi dengan nama berkas informatif, kemudian tulis interpretasi yang membedakan fakta komputasi dan kesimpulan ilmiah.

Titik kontrol: Jangan melanjutkan ke interpretasi apabila validasi input atau status proses belum memenuhi kriteria.

Kode Dasar Siap Colab

Validasi kimia dan numerik merupakan simpul penting dalam alur kerja notebook ilmiah karena menentukan bagaimana artefak ilmiah disiapkan, dijalankan, dan diperiksa. Pemeriksaan unsur, basis set, elektron, spin, SCF, nilai finite, koordinat, serta konsistensi jumlah atom file XYZ. Hasil yang dituju berupa laporan validasi dan jejak versi.

Sel kode contoh

```
import numpy as np

def validate_chemistry(atomic_numbers, charge, spin, coords):
    nelec = int(sum(atomic_numbers) - charge)
    if spin < 0 or spin > nelec or (nelec-spin) % 2:
        raise ValueError("Elektron dan spin tidak konsisten")
    arr = np.asarray(coords, dtype=float)
    if arr.shape != (len(atomic_numbers), 3) or not np.isfinite(arr).all():
        raise ValueError("Koordinat tidak valid")
    return nelec

print(validate_chemistry([8,1,1], 0, 0, [[0,0,0],[0,0.7,0.6],[0,-0.7,0.6]]))
```

Cara menjalankan: Tempatkan kode dalam satu atau beberapa sel sesuai komentar. Untuk baris %pip atau perintah shell, gunakan runtime Google Colab.

Penjelasan Baris Kode Penting

Potongan kode	Fungsi
<code>import numpy as np</code>	Menambahkan pemeriksaan untuk mencegah hasil tidak valid diteruskan.
<code>def validate_chemistry(atomic_numbers, charge, s</code>	Mencetak status agar pengguna tidak hanya bergantung pada keluaran visual.
<code>nelec = int(sum(atomic_numbers) - charge)</code>	Mencetak status agar pengguna tidak hanya bergantung pada keluaran visual.
<code>if spin < 0 or spin > nelec or (nelec-spin)</code>	Mendefinisikan input atau parameter yang akan digunakan pada langkah berikutnya.
<code>raise ValueError("Elektron dan spin tida</code>	Menambahkan pemeriksaan untuk mencegah hasil tidak valid diteruskan.
<code>arr = np.asarray(coords, dtype=float)</code>	Menjalankan operasi utama dan menyimpan hasil pada variabel yang jelas.
<code>if arr.shape != (len(atomic_numbers), 3) or</code>	Mendefinisikan input atau parameter yang akan digunakan pada langkah berikutnya.
<code>raise ValueError("Koordinat tidak valid"</code>	Menambahkan pemeriksaan untuk mencegah hasil tidak valid diteruskan.

Nama variabel dipilih agar maknanya terbaca. Hindari singkatan yang tidak dijelaskan, nilai konstan tanpa satuan, serta penggunaan variabel dari sel lama. Komentar harus menerangkan alasan atau asumsi, bukan sekadar mengulang sintaks.

Validasi dan Interpretasi Hasil

Prinsip kerja yang digunakan adalah memisahkan input, proses, validasi, dan ekspor. Pemisahan ini membuat kesalahan lebih mudah dilacak dan mencegah hasil lama dianggap sebagai hasil dari parameter baru. Setiap langkah perlu meninggalkan bukti yang dapat dibaca kembali: nilai parameter, versi pustaka, status proses, nama berkas, serta pesan kesalahan bila terjadi kegagalan.

Aspek validasi	Kriteria lulus
Input	Tipe, bentuk, satuan, muatan, spin, dan jalur berkas dinyatakan eksplisit.
Lingkungan	Versi Python dan pustaka tercatat; backend CPU/GPU terdeteksi.
Proses	Status eksekusi atau konvergensi diperiksa sebelum hasil dipakai.
Numerik	Tidak ada NaN/inf; ukuran array dan jumlah atom konsisten.
Keluaran	Laporan validasi dan jejak versi disimpan dan dapat dibuka kembali.
Interpretasi	Klaim tidak melampaui metode, basis set, atau data yang dihitung.
Interpretasi aman: Sebutkan metode, parameter, satuan, status konvergensi, dan keterbatasan saat menuliskan kesimpulan.	

Kesalahan Umum dan Troubleshooting

Risiko utama pada topik ini adalah notebook hanya berhasil pada state tertentu atau hasil tidak dapat diulang. Karena itu, notebook harus memuat pemeriksaan eksplisit dan bukan hanya mengandalkan keluaran yang tampak masuk akal. Pembaca dianjurkan menjalankan notebook dari keadaan runtime bersih, membandingkan hasil dengan batas kewajaran, dan menyimpan artefak penting di lokasi persisten.

Gejala	Tindakan perbaikan
Sel dijalankan tidak berurutan	Restart session, lalu gunakan Run all dan periksa variabel prasyarat.
Pustaka tidak ditemukan	Jalankan kembali sel %pip, periksa nama paket, lalu restart bila dependensi inti berubah.
Input tampak benar tetapi gagal	Cetak tipe, shape, satuan, muatan, spin, dan beberapa baris pertama data.

Notebook hanya berhasil pada state tertentu atau hasil tidak dapat diulang	Tambahkan validasi sebelum proses utama dan hentikan eksekusi dengan pesan yang spesifik.
Hasil berbeda antar-runtime	Catat versi runtime/paket, random seed, parameter, dan perangkat yang digunakan.
Aturan troubleshooting: Perbaiki satu hal pada satu waktu, simpan pesan kesalahan lengkap, dan catat perubahan versi atau parameter.	

Latihan Modifikasi dan Refleksi

577. Ubah satu parameter pada contoh Validasi kimia dan numerik, jalankan ulang dari runtime bersih, dan jelaskan perubahan keluaran.
578. Tambahkan satu validasi baru yang mencegah notebook hanya berhasil pada state tertentu atau hasil tidak dapat diulang.
579. Buat berkas ringkasan TXT atau JSON yang memuat input, status proses, waktu, dan hasil utama.
580. Jelaskan bagian mana yang merupakan fakta keluaran program dan bagian mana yang merupakan interpretasi ilmiah.
581. Susun satu kasus uji yang sengaja salah, lalu pastikan notebook menghasilkan pesan kesalahan yang informatif.

Refleksi: Tuliskan apa yang berubah setelah kode dimodifikasi, bukti apa yang menunjukkan keberhasilan, dan bagian mana yang masih memerlukan verifikasi eksternal.

Rubrik	Deskripsi
Skor 4	Kode berjalan dari runtime bersih, validasi lengkap, interpretasi tepat.
Skor 3	Kode berjalan dan hasil tersimpan, tetapi dokumentasi belum lengkap.
Skor 2	Kode hanya berjalan setelah intervensi atau state tertentu.
Skor 1	Kode belum berjalan dan diagnosis kesalahan belum sistematis.

5.2.3 Dokumentasi versi dan pelaporan pengujian

Dokumentasi versi dan pelaporan pengujian merupakan simpul penting dalam alur kerja notebook ilmiah karena menentukan bagaimana skrip Python disiapkan, dijalankan, dan diperiksa. Pencatatan versi Python, pustaka, runtime, perangkat keras, waktu eksekusi, dan bagian yang benar-benar telah diuji. Hasil yang dituju berupa nilai, teks, tabel, atau grafik.

Prinsip kerja yang digunakan adalah memisahkan input, proses, validasi, dan ekspor. Pemisahan ini membuat kesalahan lebih mudah dilacak dan mencegah hasil lama dianggap sebagai hasil dari parameter baru. Setiap langkah perlu meninggalkan bukti yang dapat dibaca kembali: nilai parameter, versi pustaka, status proses, nama berkas, serta pesan kesalahan bila terjadi kegagalan.

Tujuan operasional: Pembaca mampu menjelaskan Dokumentasi versi dan pelaporan pengujian, menjalankan contoh minimal, memeriksa keberhasilan, dan menyimpan bukti hasil.

Komponen	Kriteria
Konsep	Pencatatan versi Python, pustaka, runtime, perangkat keras, waktu eksekusi, dan bagian yang benar-benar telah diuji.
Input	Parameter dan data dinyatakan eksplisit.
Proses	Langkah dapat diulang dari runtime bersih.
Output	Nilai, berkas, atau visualisasi yang dapat diperiksa.
Batas	Interpretasi mengikuti metode dan konteks.

Landasan Teori dan Batas Penerapan

Pembahasan Dokumentasi versi dan pelaporan pengujian ditempatkan dalam konteks praktik Google Colab agar pembaca dapat menghubungkan konsep, prosedur, kode, dan interpretasi secara utuh. Pencatatan versi Python, pustaka, runtime, perangkat keras, waktu eksekusi, dan bagian yang benar-benar telah diuji. Hasil yang dituju berupa nilai, teks, tabel, atau grafik.

Prinsip kerja yang digunakan adalah memisahkan input, proses, validasi, dan ekspor. Pemisahan ini membuat kesalahan lebih mudah dilacak dan mencegah hasil lama dianggap sebagai hasil dari parameter baru. Setiap langkah perlu meninggalkan bukti

yang dapat dibaca kembali: nilai parameter, versi pustaka, status proses, nama berkas, serta pesan kesalahan bila terjadi kegagalan.

Risiko utama pada topik ini adalah tipe data dan urutan operasi. Karena itu, notebook harus memuat pemeriksaan eksplisit dan bukan hanya mengandalkan keluaran yang tampak masuk akal. Pembaca dianjurkan menjalankan notebook dari keadaan runtime bersih, membandingkan hasil dengan batas kewajaran, dan menyimpan artefak penting di lokasi persisten.

Batas penerapan: Contoh pada buku berfungsi sebagai pola kerja. Ukuran molekul, biaya komputasi, versi pustaka, dan perangkat keras dapat mengubah waktu serta strategi yang diperlukan.

Prosedur Bertahap

Pembahasan Dokumentasi versi dan pelaporan pengujian ditempatkan dalam konteks praktik Google Colab agar pembaca dapat menghubungkan konsep, prosedur, kode, dan interpretasi secara utuh. Pencatatan versi Python, pustaka, runtime, perangkat keras, waktu eksekusi, dan bagian yang benar-benar telah diuji. Hasil yang dituju berupa nilai, teks, tabel, atau grafik.

582. Tuliskan tujuan operasional Dokumentasi versi dan pelaporan pengujian dan tentukan keluaran yang harus tersedia pada akhir sel.
583. Tetapkan parameter pada satu sel konfigurasi agar perubahan dapat ditelusuri dan tidak tersebar di banyak tempat.
584. Siapkan skrip Python dengan input yang eksplisit; hindari nilai tersembunyi yang hanya tersimpan dalam state kernel.
585. Jalankan pemeriksaan prasyarat, termasuk keberadaan pustaka, bentuk data, tipe nilai, serta konsistensi satuan.
586. Eksekusi proses utama, tangkap status konvergensi atau keberhasilan, dan hentikan alur apabila syarat minimum tidak terpenuhi.
587. Simpan nilai, teks, tabel, atau grafik dengan nama berkas informatif, kemudian tulis interpretasi yang membedakan fakta komputasi dan kesimpulan ilmiah.

Titik kontrol: Jangan melanjutkan ke interpretasi apabila validasi input atau status proses belum memenuhi kriteria.

Kode Dasar Siap Colab

Dokumentasi versi dan pelaporan pengujian merupakan simpul penting dalam alur kerja notebook ilmiah karena menentukan bagaimana skrip Python disiapkan, dijalankan, dan diperiksa. Pencatatan versi Python, pustaka, runtime, perangkat keras, waktu eksekusi, dan bagian yang benar-benar telah diuji. Hasil yang dituju berupa nilai, teks, tabel, atau grafik.

Sel kode contoh

```
import json, platform, sys, time
from importlib.metadata import version, PackageNotFoundError

packages = {}
for name in ["numpy", "rdkit", "pyscf", "gpu4pyscf"]:
    try: packages[name] = version(name)
    except PackageNotFoundError: packages[name] = None
report = {"timestamp": time.strftime("%Y-%m-%d %H:%M:%S"),
          "python": sys.version, "platform": platform.platform(),
          "packages": packages}
open("environment.json", "w").write(json.dumps(report, indent=2))
```

Cara menjalankan: Tempatkan kode dalam satu atau beberapa sel sesuai komentar. Untuk baris %pip atau perintah shell, gunakan runtime Google Colab.

Penjelasan Baris Kode Penting

Potongan kode	Fungsi
import json, platform, sys, time	Mendefinisikan input atau parameter yang akan digunakan pada langkah berikutnya.
from importlib.metadata import version, PackageN	Menjalankan operasi utama dan menyimpan hasil pada variabel yang jelas.
packages = {}	Mencetak status agar pengguna tidak hanya bergantung pada keluaran visual.
for name in ["numpy", "rdkit", "pyscf", "gpu4pys	Mendefinisikan input atau parameter yang akan digunakan pada langkah berikutnya.
try: packages[name] = version(name)	Mendefinisikan input atau parameter yang akan digunakan pada langkah berikutnya.
except PackageNotFoundError: packages[name]	Mencetak status agar pengguna tidak hanya bergantung pada keluaran visual.
report = {"timestamp": time.strftime("%Y-%m-%d %	Menjalankan operasi utama dan menyimpan hasil pada variabel yang jelas.
"python": sys.version, "platform": pla	Mencetak status agar pengguna tidak hanya bergantung pada keluaran visual.

Nama variabel dipilih agar maknanya terbaca. Hindari singkatan yang tidak dijelaskan, nilai konstan tanpa satuan, serta penggunaan variabel dari sel lama. Komentar harus menerangkan alasan atau asumsi, bukan sekadar mengulang sintaks.

Validasi dan Interpretasi Hasil

Prinsip kerja yang digunakan adalah memisahkan input, proses, validasi, dan ekspor. Pemisahan ini membuat kesalahan lebih mudah dilacak dan mencegah hasil lama dianggap sebagai hasil dari parameter baru. Setiap langkah perlu meninggalkan bukti yang dapat dibaca kembali: nilai parameter, versi pustaka, status proses, nama berkas, serta pesan kesalahan bila terjadi kegagalan.

Aspek validasi	Kriteria lulus
Input	Tipe, bentuk, satuan, muatan, spin, dan jalur berkas dinyatakan eksplisit.
Lingkungan	Versi Python dan pustaka tercatat; backend CPU/GPU terdeteksi.
Proses	Status eksekusi atau konvergensi diperiksa sebelum hasil dipakai.
Numerik	Tidak ada NaN/inf; ukuran array dan jumlah atom konsisten.
Keluaran	Nilai, teks, tabel, atau grafik disimpan dan dapat dibuka kembali.
Interpretasi	Klaim tidak melampaui metode, basis set, atau data yang dihitung.
Interpretasi aman: Sebutkan metode, parameter, satuan, status konvergensi, dan keterbatasan saat menuliskan kesimpulan.	

Kesalahan Umum dan Troubleshooting

Risiko utama pada topik ini adalah tipe data dan urutan operasi. Karena itu, notebook harus memuat pemeriksaan eksplisit dan bukan hanya mengandalkan keluaran yang tampak masuk akal. Pembaca dianjurkan menjalankan notebook dari keadaan runtime bersih, membandingkan hasil dengan batas kewajaran, dan menyimpan artefak penting di lokasi persisten.

Gejala	Tindakan perbaikan
Sel dijalankan tidak berurutan	Restart session, lalu gunakan Run all dan periksa variabel prasyarat.
Pustaka tidak ditemukan	Jalankan kembali sel %pip, periksa nama paket, lalu restart bila dependensi inti berubah.
Input tampak benar tetapi gagal	Cetak tipe, shape, satuan, muatan, spin, dan beberapa baris pertama data.
Tipe data dan urutan operasi	Tambahkan validasi sebelum proses utama dan hentikan eksekusi dengan pesan yang spesifik.
Hasil berbeda antar-runtime	Catat versi runtime/paket, random seed, parameter, dan perangkat yang digunakan.
Aturan troubleshooting: Perbaiki satu hal pada satu waktu, simpan pesan kesalahan lengkap, dan catat perubahan versi atau parameter.	

Latihan Modifikasi dan Refleksi

588. Ubah satu parameter pada contoh Dokumentasi versi dan pelaporan pengujian, jalankan ulang dari runtime bersih, dan jelaskan perubahan keluaran.
589. Tambahkan satu validasi baru yang mencegah tipe data dan urutan operasi.
590. Buat berkas ringkasan TXT atau JSON yang memuat input, status proses, waktu, dan hasil utama.
591. Jelaskan bagian mana yang merupakan fakta keluaran program dan bagian mana yang merupakan interpretasi ilmiah.
592. Susun satu kasus uji yang sengaja salah, lalu pastikan notebook menghasilkan pesan kesalahan yang informatif.

Refleksi: Tuliskan apa yang berubah setelah kode dimodifikasi, bukti apa yang menunjukkan keberhasilan, dan bagian mana yang masih memerlukan verifikasi eksternal.

Rubrik	Deskripsi
Skor 4	Kode berjalan dari runtime bersih, validasi lengkap, interpretasi tepat.

Skor 3	Kode berjalan dan hasil tersimpan, tetapi dokumentasi belum lengkap.
Skor 2	Kode hanya berjalan setelah intervensi atau state tertentu.
Skor 1	Kode belum berjalan dan diagnosis kesalahan belum sistematis.

5.3 Proyek Akhir: Optimasi dan Analisis Parasetamol

Proyek Akhir: Optimasi dan Analisis Parasetamol mengorganisasi tiga komponen yang saling bergantung. Pembahasan bergerak dari konsep, menuju prosedur, kemudian menghasilkan artefak yang dapat divalidasi dan disimpan.

Sub-subbab	Fokus
5.3.1 Pembuatan struktur awal parasetamol	Identifikasi senyawa, SMILES, struktur 3D, atom hidrogen, ETKDGv3, MMFF94, validasi geometri, dan ekspor XYZ.
5.3.2 Optimasi DFT B3LYP/def2-TZVP	Notebook lengkap berbasis CPU dengan opsi GPU, density fitting, strategi konvergensi, optimasi geometri, dan energi final.
5.3.3 Analisis, visualisasi, dan ekspor seluruh hasil	Koordinat optimum, energi, perubahan energi, HOMO, LUMO, celah energi, dipol, parameter geometri, tampilan 3D, TXT, XYZ, CSV, JSON, dan ZIP.
Produk belajar: Satu notebook terstruktur, satu berkas keluaran, satu catatan validasi, dan satu interpretasi singkat yang menyatakan batas hasil.	

Rencana Praktik dan Produk Belajar

593. Pelajari Pembuatan struktur awal parasetamol dan jalankan kode dasar yang menyertainya.
594. Pelajari Optimasi DFT B3LYP/def2-TZVP dan jalankan kode dasar yang menyertainya.
595. Pelajari Analisis, visualisasi, dan ekspor seluruh hasil dan jalankan kode dasar yang menyertainya.

Komponen	Kriteria
Waktu	60-120 menit, bergantung pada komputasi.
Input	Notebook baru atau salinan notebook bab.
Produk	Kode, output, berkas hasil, dan refleksi.
Uji	Restart session lalu Run all.
Dokumentasi	Versi, parameter, status, dan waktu eksekusi.
Kriteria selesai: Semua sel dapat dijalankan berurutan, pesan kesalahan ditangani, dan hasil utama tersimpan di lokasi persisten.	

5.3.1 Pembuatan struktur awal parasetamol

Pembuatan struktur awal parasetamol merupakan simpul penting dalam alur kerja notebook ilmiah karena menentukan bagaimana struktur molekul disiapkan, dijalankan, dan diperiksa. Identifikasi senyawa, SMILES, struktur 3D, atom hidrogen, ETKDGv3, MMFF94, validasi geometri, dan ekspor XYZ. Hasil yang dituju berupa konformer tiga dimensi dan koordinat XYZ.

Prinsip kerja yang digunakan adalah memisahkan input, proses, validasi, dan ekspor. Pemisahan ini membuat kesalahan lebih mudah dilacak dan mencegah hasil lama dianggap sebagai hasil dari parameter baru. Setiap langkah perlu meninggalkan bukti yang dapat dibaca kembali: nilai parameter, versi pustaka, status proses, nama berkas, serta pesan kesalahan bila terjadi kegagalan.

Tujuan operasional: Pembaca mampu menjelaskan Pembuatan struktur awal parasetamol, menjalankan contoh minimal, memeriksa keberhasilan, dan menyimpan bukti hasil.

Komponen	Kriteria
Konsep	Identifikasi senyawa, SMILES, struktur 3D, atom hidrogen, ETKDGv3, MMFF94, validasi geometri, dan ekspor XYZ.
Input	Parameter dan data dinyatakan eksplisit.

Proses	Langkah dapat diulang dari runtime bersih.
Output	Nilai, berkas, atau visualisasi yang dapat diperiksa.
Batas	Interpretasi mengikuti metode dan konteks.

Landasan Teori dan Batas Penerapan

Pada bagian Pembuatan struktur awal parasetamol, perhatian tidak hanya diarahkan pada keberhasilan menjalankan kode, tetapi juga pada ketepatan asumsi, keterlacakan parameter, dan mutu keluaran. Identifikasi senyawa, SMILES, struktur 3D, atom hidrogen, ETKDGv3, MMFF94, validasi geometri, dan ekspor XYZ. Hasil yang dituju berupa konformer tiga dimensi dan koordinat XYZ.

Prinsip kerja yang digunakan adalah memisahkan input, proses, validasi, dan ekspor. Pemisahan ini membuat kesalahan lebih mudah dilacak dan mencegah hasil lama dianggap sebagai hasil dari parameter baru. Setiap langkah perlu meninggalkan bukti yang dapat dibaca kembali: nilai parameter, versi pustaka, status proses, nama berkas, serta pesan kesalahan bila terjadi kegagalan.

Risiko utama pada topik ini adalah struktur 2D, stereokimia, muatan, dan koordinat tidak valid. Karena itu, notebook harus memuat pemeriksaan eksplisit dan bukan hanya mengandalkan keluaran yang tampak masuk akal. Pembaca dianjurkan menjalankan notebook dari keadaan runtime bersih, membandingkan hasil dengan batas kewajaran, dan menyimpan artefak penting di lokasi persisten.

Batas penerapan: Contoh pada buku berfungsi sebagai pola kerja. Ukuran molekul, biaya komputasi, versi pustaka, dan perangkat keras dapat mengubah waktu serta strategi yang diperlukan.

Prosedur Bertahap

Pada bagian Pembuatan struktur awal parasetamol, perhatian tidak hanya diarahkan pada keberhasilan menjalankan kode, tetapi juga pada ketepatan asumsi, keterlacakan parameter, dan mutu keluaran. Identifikasi senyawa, SMILES, struktur 3D, atom hidrogen, ETKDGv3, MMFF94, validasi geometri, dan ekspor XYZ. Hasil yang dituju berupa konformer tiga dimensi dan koordinat XYZ.

596. Tuliskan tujuan operasional Pembuatan struktur awal parasetamol dan tentukan keluaran yang harus tersedia pada akhir sel.
597. Tetapkan parameter pada satu sel konfigurasi agar perubahan dapat ditelusuri dan tidak tersebar di banyak tempat.

598. Siapkan struktur molekul dengan input yang eksplisit; hindari nilai tersembunyi yang hanya tersimpan dalam state kernel.
599. Jalankan pemeriksaan prasyarat, termasuk keberadaan pustaka, bentuk data, tipe nilai, serta konsistensi satuan.
600. Eksekusi proses utama, tangkap status konvergensi atau keberhasilan, dan hentikan alur apabila syarat minimum tidak terpenuhi.
601. Simpan konformer tiga dimensi dan koordinat XYZ dengan nama berkas informatif, kemudian tulis interpretasi yang membedakan fakta komputasi dan kesimpulan ilmiah.

Titik kontrol: Jangan melanjutkan ke interpretasi apabila validasi input atau status proses belum memenuhi kriteria.

Kode Dasar Siap Colab

Pembahasan Pembuatan struktur awal parasetamol ditempatkan dalam konteks praktik Google Colab agar pembaca dapat menghubungkan konsep, prosedur, kode, dan interpretasi secara utuh. Identifikasi senyawa, SMILES, struktur 3D, atom hidrogen, ETKDGV3, MMFF94, validasi geometri, dan ekspor XYZ. Hasil yang dituju berupa konformer tiga dimensi dan koordinat XYZ.

Sel kode contoh

```
from rdkit import Chem
from rdkit.Chem import AllChem

SMILES = "CC(=O)NC1=CC=C(O)C=C1"
mol = Chem.AddHs(Chem.MolFromSmiles(SMILES))
params = AllChem.ETKDGv3(); params.randomSeed = 2026
if AllChem.EmbedMolecule(mol, params) != 0:
    raise RuntimeError("Konformer parasetamol gagal dibuat")
if AllChem.MMFFHasAllMoleculeParams(mol):
    AllChem.MMFFOptimizeMolecule(mol, maxIters=1000)
else:
    AllChem.UFFOptimizeMolecule(mol, maxIters=1000)
Chem.MolToXYZFile(mol, "paracetamol_awal.xyz")
```

Cara menjalankan: Tempatkan kode dalam satu atau beberapa sel sesuai komentar. Untuk baris %pip atau perintah shell, gunakan runtime Google Colab.

Penjelasan Baris Kode Penting

Potongan kode	Fungsi

<code>from rdkit import Chem</code>	Menjalankan operasi utama dan menyimpan hasil pada variabel yang jelas.
<code>from rdkit.Chem import AllChem</code>	Menjalankan operasi utama dan menyimpan hasil pada variabel yang jelas.
<code>SMILES = "CC(=O)NC1=CC=C(O)C=C1"</code>	Mendefinisikan input atau parameter yang akan digunakan pada langkah berikutnya.
<code>mol = Chem.AddHs(Chem.MolFromSmiles(SMILES))</code>	Mencetak status agar pengguna tidak hanya bergantung pada keluaran visual.
<code>params = AllChem.ETKDGv3(); params.randomSeed =</code>	Menambahkan pemeriksaan untuk mencegah hasil tidak valid diteruskan.
<code>if AllChem.EmbedMolecule(mol, params) != 0:</code>	Menambahkan pemeriksaan untuk mencegah hasil tidak valid diteruskan.
<code>raise RuntimeError("Konformer parasetamol ga</code>	Menambahkan pemeriksaan untuk mencegah hasil tidak valid diteruskan.
<code>if AllChem.MMFFHasAllMoleculeParams(mol):</code>	Menambahkan pemeriksaan untuk mencegah hasil tidak valid diteruskan.

Nama variabel dipilih agar maknanya terbaca. Hindari singkatan yang tidak dijelaskan, nilai konstan tanpa satuan, serta penggunaan variabel dari sel lama. Komentar harus menerangkan alasan atau asumsi, bukan sekadar mengulang sintaks.

Validasi dan Interpretasi Hasil

Prinsip kerja yang digunakan adalah memisahkan input, proses, validasi, dan ekspor. Pemisahan ini membuat kesalahan lebih mudah dilacak dan mencegah hasil lama dianggap sebagai hasil dari parameter baru. Setiap langkah perlu meninggalkan bukti yang dapat dibaca kembali: nilai parameter, versi pustaka, status proses, nama berkas, serta pesan kesalahan bila terjadi kegagalan.

Aspek validasi	Kriteria lulus
Input	Tipe, bentuk, satuan, muatan, spin, dan jalur berkas dinyatakan eksplisit.
Lingkungan	Versi Python dan pustaka tercatat; backend CPU/GPU terdeteksi.
Proses	Status eksekusi atau konvergensi diperiksa sebelum hasil dipakai.
Numerik	Tidak ada NaN/inf; ukuran array dan jumlah atom konsisten.
Keluaran	Konformer tiga dimensi dan koordinat xyz disimpan dan dapat dibuka kembali.
Interpretasi	Klaim tidak melampaui metode, basis set, atau data yang dihitung.
Interpretasi aman: Sebutkan metode, parameter, satuan, status konvergensi, dan keterbatasan saat menuliskan kesimpulan.	

Kesalahan Umum dan Troubleshooting

Risiko utama pada topik ini adalah struktur 2D, stereokimia, muatan, dan koordinat tidak valid. Karena itu, notebook harus memuat pemeriksaan eksplisit dan bukan hanya mengandalkan keluaran yang tampak masuk akal. Pembaca dianjurkan menjalankan notebook dari keadaan runtime bersih, membandingkan hasil dengan batas kewajaran, dan menyimpan artefak penting di lokasi persisten.

Gejala	Tindakan perbaikan
Sel dijalankan tidak berurutan	Restart session, lalu gunakan Run all dan periksa variabel prasyarat.
Pustaka tidak ditemukan	Jalankan kembali sel %pip, periksa nama paket, lalu restart bila dependensi inti berubah.
Input tampak benar tetapi gagal	Cetak tipe, shape, satuan, muatan, spin, dan beberapa baris pertama data.

Struktur 2d, stereokimia, muatan, dan koordinat tidak valid	Tambahkan validasi sebelum proses utama dan hentikan eksekusi dengan pesan yang spesifik.
Hasil berbeda antar-runtime	Catat versi runtime/paket, random seed, parameter, dan perangkat yang digunakan.
Aturan troubleshooting: Perbaiki satu hal pada satu waktu, simpan pesan kesalahan lengkap, dan catat perubahan versi atau parameter.	

Latihan Modifikasi dan Refleksi

602. Ubah satu parameter pada contoh Pembuatan struktur awal parasetamol, jalankan ulang dari runtime bersih, dan jelaskan perubahan keluaran.
603. Tambahkan satu validasi baru yang mencegah struktur 2D, stereokimia, muatan, dan koordinat tidak valid.
604. Buat berkas ringkasan TXT atau JSON yang memuat input, status proses, waktu, dan hasil utama.
605. Jelaskan bagian mana yang merupakan fakta keluaran program dan bagian mana yang merupakan interpretasi ilmiah.
606. Susun satu kasus uji yang sengaja salah, lalu pastikan notebook menghasilkan pesan kesalahan yang informatif.

Refleksi: Tuliskan apa yang berubah setelah kode dimodifikasi, bukti apa yang menunjukkan keberhasilan, dan bagian mana yang masih memerlukan verifikasi eksternal.

Rubrik	Deskripsi
Skor 4	Kode berjalan dari runtime bersih, validasi lengkap, interpretasi tepat.
Skor 3	Kode berjalan dan hasil tersimpan, tetapi dokumentasi belum lengkap.
Skor 2	Kode hanya berjalan setelah intervensi atau state tertentu.
Skor 1	Kode belum berjalan dan diagnosis kesalahan belum sistematis.

5.3.2 Optimasi DFT B3LYP/def2-TZVP

Pada bagian Optimasi DFT B3LYP/def2-TZVP, perhatian tidak hanya diarahkan pada keberhasilan menjalankan kode, tetapi juga pada ketepatan asumsi, keterlacakan parameter, dan mutu keluaran. Notebook lengkap berbasis CPU dengan opsi GPU, density fitting, strategi konvergensi, optimasi geometri, dan energi final. Hasil yang dituju berupa sel Markdown, sel kode, output, dan metadata.

Prinsip kerja yang digunakan adalah memisahkan input, proses, validasi, dan ekspor. Pemisahan ini membuat kesalahan lebih mudah dilacak dan mencegah hasil lama dianggap sebagai hasil dari parameter baru. Setiap langkah perlu meninggalkan bukti yang dapat dibaca kembali: nilai parameter, versi pustaka, status proses, nama berkas, serta pesan kesalahan bila terjadi kegagalan.

Tujuan operasional: Pembaca mampu menjelaskan Optimasi DFT B3LYP/def2-TZVP, menjalankan contoh minimal, memeriksa keberhasilan, dan menyimpan bukti hasil.

Komponen	Kriteria
Konsep	Notebook lengkap berbasis CPU dengan opsi GPU, density fitting, strategi konvergensi, optimasi geometri, dan energi final.
Input	Parameter dan data dinyatakan eksplisit.
Proses	Langkah dapat diulang dari runtime bersih.
Output	Nilai, berkas, atau visualisasi yang dapat diperiksa.
Batas	Interpretasi mengikuti metode dan konteks.

Landasan Teori dan Batas Penerapan

Pembahasan Optimasi DFT B3LYP/def2-TZVP ditempatkan dalam konteks praktik Google Colab agar pembaca dapat menghubungkan konsep, prosedur, kode, dan interpretasi secara utuh. Notebook lengkap berbasis CPU dengan opsi GPU, density fitting, strategi konvergensi, optimasi geometri, dan energi final. Hasil yang dituju berupa sel Markdown, sel kode, output, dan metadata.

Prinsip kerja yang digunakan adalah memisahkan input, proses, validasi, dan ekspor. Pemisahan ini membuat kesalahan lebih mudah dilacak dan mencegah hasil lama dianggap sebagai hasil dari parameter baru. Setiap langkah perlu meninggalkan bukti

yang dapat dibaca kembali: nilai parameter, versi pustaka, status proses, nama berkas, serta pesan kesalahan bila terjadi kegagalan.

Risiko utama pada topik ini adalah state kernel tersembunyi dan urutan eksekusi. Karena itu, notebook harus memuat pemeriksaan eksplisit dan bukan hanya mengandalkan keluaran yang tampak masuk akal. Pembaca dianjurkan menjalankan notebook dari keadaan runtime bersih, membandingkan hasil dengan batas kewajaran, dan menyimpan artefak penting di lokasi persisten.

Batas penerapan: Contoh pada buku berfungsi sebagai pola kerja. Ukuran molekul, biaya komputasi, versi pustaka, dan perangkat keras dapat mengubah waktu serta strategi yang diperlukan.

Prosedur Bertahap

Pembahasan Optimasi DFT B3LYP/def2-TZVP ditempatkan dalam konteks praktik Google Colab agar pembaca dapat menghubungkan konsep, prosedur, kode, dan interpretasi secara utuh. Notebook lengkap berbasis CPU dengan opsi GPU, density fitting, strategi konvergensi, optimasi geometri, dan energi final. Hasil yang dituju berupa sel Markdown, sel kode, output, dan metadata.

607. Tuliskan tujuan operasional Optimasi DFT B3LYP/def2-TZVP dan tentukan keluaran yang harus tersedia pada akhir sel.
608. Tetapkan parameter pada satu sel konfigurasi agar perubahan dapat ditelusuri dan tidak tersebar di banyak tempat.
609. Siapkan notebook dengan input yang eksplisit; hindari nilai tersembunyi yang hanya tersimpan dalam state kernel.
610. Jalankan pemeriksaan prasyarat, termasuk keberadaan pustaka, bentuk data, tipe nilai, serta konsistensi satuan.
611. Eksekusi proses utama, tangkap status konvergensi atau keberhasilan, dan hentikan alur apabila syarat minimum tidak terpenuhi.
612. Simpan sel Markdown, sel kode, output, dan metadata dengan nama berkas informatif, kemudian tulis interpretasi yang membedakan fakta komputasi dan kesimpulan ilmiah.

Titik kontrol: Jangan melanjutkan ke interpretasi apabila validasi input atau status proses belum memenuhi kriteria.

Kode Dasar Siap Colab

Pada bagian Optimasi DFT B3LYP/def2-TZVP, perhatian tidak hanya diarahkan pada keberhasilan menjalankan kode, tetapi juga pada ketepatan asumsi, keterlacakan parameter, dan mutu keluaran. Notebook lengkap berbasis CPU dengan opsi GPU, density fitting, strategi konvergensi, optimasi geometri, dan energi final. Hasil yang dituju berupa sel Markdown, sel kode, output, dan metadata.

Sel kode contoh

```
from pyscf import gto, dft
from pyscf.geomopt.geometric_solver import optimize

atom_text = "\n".join(open("paracetamol_awal.xyz").read().splitlines()[2:])
mol_qm = gto.M(atom=atom_text, unit="Angstrom", basis="def2-tzvp", charge=0,
spin=0)
mf = dft.RKS(mol_qm).density_fit()
mf.xc = "b3lyp"; mf.grids.level = 4
mf.conv_tol = 1e-8; mf.max_cycle = 200
e0 = mf.kernel()
if not mf.converged: mf = mf.newton(); e0 = mf.kernel()
mol_opt = optimize(mf, maxsteps=200)
```

Cara menjalankan: Tempatkan kode dalam satu atau beberapa sel sesuai komentar. Untuk baris %pip atau perintah shell, gunakan runtime Google Colab.

Penjelasan Baris Kode Penting

Potongan kode	Fungsi
from pyscf import gto, dft	Mencetak status agar pengguna tidak hanya bergantung pada keluaran visual.
from pyscf.geomopt.geometric_solver import optim	Menjalankan operasi utama dan menyimpan hasil pada variabel yang jelas.
atom_text = "\n".join(open("paracetamol_awal.xyz	Mendefinisikan input atau parameter yang akan digunakan pada langkah berikutnya.
mol_qm = gto.M(atom=atom_text, unit="Angstrom",	Menjalankan operasi utama dan menyimpan hasil pada variabel yang jelas.
mf = dft.RKS(mol_qm).density_fit()	Mendefinisikan input atau parameter yang akan digunakan pada langkah berikutnya.
mf.xc = "b3lyp"; mf.grids.level = 4	Mencetak status agar pengguna tidak hanya bergantung pada keluaran visual.

<code>mf.conv_tol = 1e-8; mf.max_cycle = 200</code>	Mencetak status agar pengguna tidak hanya bergantung pada keluaran visual.
<code>e0 = mf.kernel()</code>	Menambahkan pemeriksaan untuk mencegah hasil tidak valid diteruskan.

Nama variabel dipilih agar maknanya terbaca. Hindari singkatan yang tidak dijelaskan, nilai konstan tanpa satuan, serta penggunaan variabel dari sel lama. Komentar harus menerangkan alasan atau asumsi, bukan sekadar mengulang sintaks.

Validasi dan Interpretasi Hasil

Prinsip kerja yang digunakan adalah memisahkan input, proses, validasi, dan ekspor. Pemisahan ini membuat kesalahan lebih mudah dilacak dan mencegah hasil lama dianggap sebagai hasil dari parameter baru. Setiap langkah perlu meninggalkan bukti yang dapat dibaca kembali: nilai parameter, versi pustaka, status proses, nama berkas, serta pesan kesalahan bila terjadi kegagalan.

Aspek validasi	Kriteria lulus
Input	Tipe, bentuk, satuan, muatan, spin, dan jalur berkas dinyatakan eksplisit.
Lingkungan	Versi Python dan pustaka tercatat; backend CPU/GPU terdeteksi.
Proses	Status eksekusi atau konvergensi diperiksa sebelum hasil dipakai.
Numerik	Tidak ada NaN/inf; ukuran array dan jumlah atom konsisten.
Keluaran	Sel markdown, sel kode, output, dan metadata disimpan dan dapat dibuka kembali.
Interpretasi	Klaim tidak melampaui metode, basis set, atau data yang dihitung.
Interpretasi aman: Sebutkan metode, parameter, satuan, status konvergensi, dan keterbatasan saat menuliskan kesimpulan.	

Kesalahan Umum dan Troubleshooting

Risiko utama pada topik ini adalah state kernel tersembunyi dan urutan eksekusi. Karena itu, notebook harus memuat pemeriksaan eksplisit dan bukan hanya mengandalkan keluaran yang tampak masuk akal. Pembaca dianjurkan menjalankan notebook dari keadaan runtime bersih, membandingkan hasil dengan batas kewajaran, dan menyimpan artefak penting di lokasi persisten.

Gejala	Tindakan perbaikan
Sel dijalankan tidak berurutan	Restart session, lalu gunakan Run all dan periksa variabel prasyarat.
Pustaka tidak ditemukan	Jalankan kembali sel %pip, periksa nama paket, lalu restart bila dependensi inti berubah.
Input tampak benar tetapi gagal	Cetak tipe, shape, satuan, muatan, spin, dan beberapa baris pertama data.
State kernel tersembunyi dan urutan eksekusi	Tambahkan validasi sebelum proses utama dan hentikan eksekusi dengan pesan yang spesifik.
Hasil berbeda antar-runtime	Catat versi runtime/paket, random seed, parameter, dan perangkat yang digunakan.
Aturan troubleshooting: Perbaiki satu hal pada satu waktu, simpan pesan kesalahan lengkap, dan catat perubahan versi atau parameter.	

Latihan Modifikasi dan Refleksi

613. Ubah satu parameter pada contoh Optimasi DFT B3LYP/def2-TZVP, jalankan ulang dari runtime bersih, dan jelaskan perubahan keluaran.
614. Tambahkan satu validasi baru yang mencegah state kernel tersembunyi dan urutan eksekusi.
615. Buat berkas ringkasan TXT atau JSON yang memuat input, status proses, waktu, dan hasil utama.
616. Jelaskan bagian mana yang merupakan fakta keluaran program dan bagian mana yang merupakan interpretasi ilmiah.
617. Susun satu kasus uji yang sengaja salah, lalu pastikan notebook menghasilkan pesan kesalahan yang informatif.

Refleksi: Tuliskan apa yang berubah setelah kode dimodifikasi, bukti apa yang menunjukkan keberhasilan, dan bagian mana yang masih memerlukan verifikasi eksternal.

Rubrik	Deskripsi
Skor 4	Kode berjalan dari runtime bersih, validasi lengkap, interpretasi tepat.
Skor 3	Kode berjalan dan hasil tersimpan, tetapi dokumentasi belum lengkap.
Skor 2	Kode hanya berjalan setelah intervensi atau state tertentu.
Skor 1	Kode belum berjalan dan diagnosis kesalahan belum sistematis.

5.3.3 Analisis, visualisasi, dan ekspor seluruh hasil

Pembahasan Analisis, visualisasi, dan ekspor seluruh hasil ditempatkan dalam konteks praktik Google Colab agar pembaca dapat menghubungkan konsep, prosedur, kode, dan interpretasi secara utuh. Koordinat optimum, energi, perubahan energi, HOMO, LUMO, celah energi, dipol, parameter geometri, tampilan 3D, TXT, XYZ, CSV, JSON, dan ZIP. Hasil yang dituju berupa energi, geometri, orbital, dan sifat molekul.

Prinsip kerja yang digunakan adalah memisahkan input, proses, validasi, dan ekspor. Pemisahan ini membuat kesalahan lebih mudah dilacak dan mencegah hasil lama dianggap sebagai hasil dari parameter baru. Setiap langkah perlu meninggalkan bukti yang dapat dibaca kembali: nilai parameter, versi pustaka, status proses, nama berkas, serta pesan kesalahan bila terjadi kegagalan.

Tujuan operasional: Pembaca mampu menjelaskan Analisis, visualisasi, dan ekspor seluruh hasil, menjalankan contoh minimal, memeriksa keberhasilan, dan menyimpan bukti hasil.

Komponen	Kriteria
Konsep	Koordinat optimum, energi, perubahan energi, HOMO, LUMO, celah energi, dipol, parameter geometri, tampilan 3D, TXT, XYZ, CSV, JSON, dan ZIP.

Input	Parameter dan data dinyatakan eksplisit.
Proses	Langkah dapat diulang dari runtime bersih.
Output	Nilai, berkas, atau visualisasi yang dapat diperiksa.
Batas	Interpretasi mengikuti metode dan konteks.

Landasan Teori dan Batas Penerapan

Pembahasan Analisis, visualisasi, dan ekspor seluruh hasil ditempatkan dalam konteks praktik Google Colab agar pembaca dapat menghubungkan konsep, prosedur, kode, dan interpretasi secara utuh. Koordinat optimum, energi, perubahan energi, HOMO, LUMO, celah energi, dipol, parameter geometri, tampilan 3D, TXT, XYZ, CSV, JSON, dan ZIP. Hasil yang dituju berupa energi, geometri, orbital, dan sifat molekul.

Prinsip kerja yang digunakan adalah memisahkan input, proses, validasi, dan ekspor. Pemisahan ini membuat kesalahan lebih mudah dilacak dan mencegah hasil lama dianggap sebagai hasil dari parameter baru. Setiap langkah perlu meninggalkan bukti yang dapat dibaca kembali: nilai parameter, versi pustaka, status proses, nama berkas, serta pesan kesalahan bila terjadi kegagalan.

Risiko utama pada topik ini adalah SCF tidak konvergen, spin keliru, atau interpretasi lintas metode. Karena itu, notebook harus memuat pemeriksaan eksplisit dan bukan hanya mengandalkan keluaran yang tampak masuk akal. Pembaca dianjurkan menjalankan notebook dari keadaan runtime bersih, membandingkan hasil dengan batas kewajaran, dan menyimpan artefak penting di lokasi persisten.

Batas penerapan: Contoh pada buku berfungsi sebagai pola kerja. Ukuran molekul, biaya komputasi, versi pustaka, dan perangkat keras dapat mengubah waktu serta strategi yang diperlukan.

Prosedur Bertahap

Pembahasan Analisis, visualisasi, dan ekspor seluruh hasil ditempatkan dalam konteks praktik Google Colab agar pembaca dapat menghubungkan konsep, prosedur, kode, dan interpretasi secara utuh. Koordinat optimum, energi, perubahan energi, HOMO, LUMO, celah energi, dipol, parameter geometri, tampilan 3D, TXT, XYZ, CSV, JSON, dan ZIP. Hasil yang dituju berupa energi, geometri, orbital, dan sifat molekul.

618. Tuliskan tujuan operasional Analisis, visualisasi, dan ekspor seluruh hasil dan tentukan keluaran yang harus tersedia pada akhir sel.

619. Tetapkan parameter pada satu sel konfigurasi agar perubahan dapat ditelusuri dan tidak tersebar di banyak tempat.
620. Siapkan perhitungan struktur elektronik dengan input yang eksplisit; hindari nilai tersembunyi yang hanya tersimpan dalam state kernel.
621. Jalankan pemeriksaan prasyarat, termasuk keberadaan pustaka, bentuk data, tipe nilai, serta konsistensi satuan.
622. Eksekusi proses utama, tangkap status konvergensi atau keberhasilan, dan hentikan alur apabila syarat minimum tidak terpenuhi.
623. Simpan energi, geometri, orbital, dan sifat molekul dengan nama berkas informatif, kemudian tulis interpretasi yang membedakan fakta komputasi dan kesimpulan ilmiah.

Titik kontrol: Jangan melanjutkan ke interpretasi apabila validasi input atau status proses belum memenuhi kriteria.

Kode Dasar Siap Colab

Pembahasan Analisis, visualisasi, dan ekspor seluruh hasil ditempatkan dalam konteks praktik Google Colab agar pembaca dapat menghubungkan konsep, prosedur, kode, dan interpretasi secara utuh. Koordinat optimum, energi, perubahan energi, HOMO, LUMO, celah energi, dipol, parameter geometri, tampilan 3D, TXT, XYZ, CSV, JSON, dan ZIP. Hasil yang dituju berupa energi, geometri, orbital, dan sifat molekul.

Sel kode contoh

```
import json, shutil
from pathlib import Path

result = {"molecule": "paracetamol", "method": "B3LYP",
          "basis": "def2-TZVP", "energy_hartree": float(e_final),
          "scf_converged": bool(mf_final.converged)}
Path("hasil.json").write_text(json.dumps(result, indent=2), encoding="utf-8")
Path("hasil.txt").write_text("\n".join(f"{k}: {v}" for k,v in result.items()),
                              encoding="utf-8")
shutil.make_archive("paracetamol_results", "zip", root_dir=".",
                   base_dir=None)
```

Cara menjalankan: Tempatkan kode dalam satu atau beberapa sel sesuai komentar. Untuk baris %pip atau perintah shell, gunakan runtime Google Colab.

Penjelasan Baris Kode Penting

Potongan kode

Fungsi

<code>import json, shutil</code>	Menjalankan operasi utama dan menyimpan hasil pada variabel yang jelas.
<code>from pathlib import Path</code>	Mendefinisikan input atau parameter yang akan digunakan pada langkah berikutnya.
<code>result = {"molecule": "paracetamol", "method": "B3"</code>	Menambahkan pemeriksaan untuk mencegah hasil tidak valid diteruskan.
<code>"basis": "def2-TZVP", "energy_hartree":</code>	Menambahkan pemeriksaan untuk mencegah hasil tidak valid diteruskan.
<code>"scf_converged": bool(mf_final.converg</code>	Menambahkan pemeriksaan untuk mencegah hasil tidak valid diteruskan.
<code>Path("hasil.json").write_text(json.dumps(result,</code>	Mencetak status agar pengguna tidak hanya bergantung pada keluaran visual.
<code>Path("hasil.txt").write_text("\n".join(f"{k}: {v</code>	Mendefinisikan input atau parameter yang akan digunakan pada langkah berikutnya.
<code>shutil.make_archive("paracetamol_results", "zip"</code>	Mendefinisikan input atau parameter yang akan digunakan pada langkah berikutnya.

Nama variabel dipilih agar maknanya terbaca. Hindari singkatan yang tidak dijelaskan, nilai konstan tanpa satuan, serta penggunaan variabel dari sel lama. Komentar harus menerangkan alasan atau asumsi, bukan sekadar mengulang sintaks.

Validasi dan Interpretasi Hasil

Prinsip kerja yang digunakan adalah memisahkan input, proses, validasi, dan ekspor. Pemisahan ini membuat kesalahan lebih mudah dilacak dan mencegah hasil lama dianggap sebagai hasil dari parameter baru. Setiap langkah perlu meninggalkan bukti yang dapat dibaca kembali: nilai parameter, versi pustaka, status proses, nama berkas, serta pesan kesalahan bila terjadi kegagalan.

Aspek validasi	Kriteria lulus
Input	Tipe, bentuk, satuan, muatan, spin, dan jalur berkas dinyatakan eksplisit.
Lingkungan	Versi Python dan pustaka tercatat; backend CPU/GPU terdeteksi.
Proses	Status eksekusi atau konvergensi diperiksa sebelum hasil dipakai.
Numerik	Tidak ada NaN/inf; ukuran array dan jumlah atom konsisten.
Keluaran	Energi, geometri, orbital, dan sifat molekul disimpan dan dapat dibuka kembali.
Interpretasi	Klaim tidak melampaui metode, basis set, atau data yang dihitung.
Interpretasi aman: Sebutkan metode, parameter, satuan, status konvergensi, dan keterbatasan saat menuliskan kesimpulan.	

Kesalahan Umum dan Troubleshooting

Risiko utama pada topik ini adalah SCF tidak konvergen, spin keliru, atau interpretasi lintas metode. Karena itu, notebook harus memuat pemeriksaan eksplisit dan bukan hanya mengandalkan keluaran yang tampak masuk akal. Pembaca dianjurkan menjalankan notebook dari keadaan runtime bersih, membandingkan hasil dengan batas kewajaran, dan menyimpan artefak penting di lokasi persisten.

Gejala	Tindakan perbaikan
Sel dijalankan tidak berurutan	Restart session, lalu gunakan Run all dan periksa variabel prasyarat.
Pustaka tidak ditemukan	Jalankan kembali sel %pip, periksa nama paket, lalu restart bila dependensi inti berubah.
Input tampak benar tetapi gagal	Cetak tipe, shape, satuan, muatan, spin, dan beberapa baris pertama data.

Scf tidak konvergen, spin keliru, atau interpretasi lintas metode	Tambahkan validasi sebelum proses utama dan hentikan eksekusi dengan pesan yang spesifik.
Hasil berbeda antar-runtime	Catat versi runtime/paket, random seed, parameter, dan perangkat yang digunakan.
Aturan troubleshooting: Perbaiki satu hal pada satu waktu, simpan pesan kesalahan lengkap, dan catat perubahan versi atau parameter.	

Latihan Modifikasi dan Refleksi

624. Ubah satu parameter pada contoh Analisis, visualisasi, dan ekspor seluruh hasil, jalankan ulang dari runtime bersih, dan jelaskan perubahan keluaran.
625. Tambahkan satu validasi baru yang mencegah SCF tidak konvergen, spin keliru, atau interpretasi lintas metode.
626. Buat berkas ringkasan TXT atau JSON yang memuat input, status proses, waktu, dan hasil utama.
627. Jelaskan bagian mana yang merupakan fakta keluaran program dan bagian mana yang merupakan interpretasi ilmiah.
628. Susun satu kasus uji yang sengaja salah, lalu pastikan notebook menghasilkan pesan kesalahan yang informatif.

Refleksi: Tuliskan apa yang berubah setelah kode dimodifikasi, bukti apa yang menunjukkan keberhasilan, dan bagian mana yang masih memerlukan verifikasi eksternal.

Rubrik	Deskripsi
Skor 4	Kode berjalan dari runtime bersih, validasi lengkap, interpretasi tepat.
Skor 3	Kode berjalan dan hasil tersimpan, tetapi dokumentasi belum lengkap.
Skor 2	Kode hanya berjalan setelah intervensi atau state tertentu.
Skor 1	Kode belum berjalan dan diagnosis kesalahan belum sistematis.

Rangkuman Bab

Bab ini membahas penggunaan GPU secara selektif, validasi notebook, dan proyek parasetamol. Fokus utama adalah kompatibilitas, fallback CPU, pelaporan pengujian, dan interpretasi yang jujur.

- Mengaktifkan dan mendeteksi GPU
- Instalasi GPU4PySCF yang kompatibel
- Perbandingan CPU, GPU, dan mode otomatis
- Validasi format dan sintaks
- Validasi kimia dan numerik
- Dokumentasi versi dan pelaporan pengujian
- Pembuatan struktur awal parasetamol
- Optimasi DFT B3LYP/def2-TZVP
- Analisis, visualisasi, dan ekspor seluruh hasil

Inti bab: Keberhasilan teknis harus selalu disertai validasi, dokumentasi, dan interpretasi yang proporsional.

Proyek Praktik Terpadu

Proyek terpadu menggabungkan seluruh keterampilan bab dalam satu notebook. Gunakan satu sel konfigurasi, fungsi modular, validasi, logging sederhana, penyimpanan hasil, dan bagian pembahasan.

629. Tentukan pertanyaan atau tujuan komputasi.
630. Siapkan input dan parameter.
631. Jalankan proses utama.
632. Validasi status dan nilai.
633. Buat satu visualisasi atau tabel.
634. Ekspor hasil dan tulis pembahasan.
635. Restart session dan Run all.

Produk akhir: Notebook .ipynb, satu folder hasil, arsip ZIP, dan ringkasan pengujian.

Pertanyaan Analitis

636. Mengapa kode yang menghasilkan angka belum tentu menghasilkan bukti ilmiah?
637. Bagaimana versi runtime dapat memengaruhi reproduksibilitas?
638. Apa perbedaan pemeriksaan sintaks, pemeriksaan numerik, dan validasi kimia?
639. Kapan fallback dapat diterima, dan bagaimana harus dilaporkan?
640. Apa batas interpretasi dari metode atau contoh pada bab ini?
641. Bagaimana merancang notebook agar mudah direview oleh orang lain?

Panduan Jawaban dan Rubrik

Komponen	Bobot dan indikator
Ketepatan konsep	25% - definisi, asumsi, dan satuan benar.
Kualitas kode	25% - modular, terbaca, dan menangani kesalahan.
Validasi	25% - input, proses, dan keluaran diperiksa.
Interpretasi	15% - klaim sesuai bukti dan keterbatasan.
Reproduksibilitas	10% - dapat dijalankan dari runtime bersih.

Panduan jawaban menilai alur penalaran, bukan hanya keluaran akhir. Perbedaan nilai kecil dapat diterima bila versi atau perangkat berbeda dan semua parameter dilaporkan.

Daftar Periksa Penguasaan Bab

Pernyataan	Cek
Saya dapat menjelaskan konsep utama	☐
Saya dapat menjalankan kode dari runtime bersih	☐
Saya memeriksa status dan nilai numerik	☐
Saya menyimpan hasil di lokasi persisten	☐
Saya dapat menjelaskan kesalahan umum	☐
Saya menuliskan batas interpretasi	☐
Saya mencatat versi dan waktu pengujian	☐
Tindak lanjut: Ulangi halaman yang belum dikuasai, lalu kerjakan proyek bab tanpa melihat kode contoh secara langsung.	

DAFTAR PUSTAKA

Google Research. (2026). Colaboratory: Frequently Asked Questions. Google Colab.

Google Research. (2026). Past Runtime Versions FAQ. Google Colab.

Jupyter Project. (2026). Jupyter documentation: Notebook formatting and conversion.

Project Jupyter. (2026). nbformat documentation: The Jupyter Notebook format and Python API.

Catatan referensi: Dokumentasi perangkat lunak bersifat dinamis. Periksa dokumentasi resmi sebelum workshop atau penerbitan ulang.

Daftar Pustaka - Bagian 2

Sun, Q., et al. (2018). PySCF: The Python-based Simulations of Chemistry Framework. Wiley Interdisciplinary Reviews: Computational Molecular Science, 8(1), e1340.

PySCF Developers. (2026). PySCF User Guide: Installation, SCF, DFT, geometry optimization, and GPU acceleration.

Li, R., Sun, Q., Zhang, X., & Chan, G. K.-L. (2025). Introducing GPU Acceleration into the Python-Based Simulations of Chemistry Framework. The Journal of Physical Chemistry A, 129(5), 1459-1468.

Wu, X., Sun, Q., Pu, Z., et al. (2025). Enhancing GPU-Acceleration in the Python-Based Simulations of Chemistry Frameworks. WIREs Computational Molecular Science, 15, e70008.

Catatan referensi: Dokumentasi perangkat lunak bersifat dinamis. Periksa dokumentasi resmi sebelum workshop atau penerbitan ulang.

Daftar Pustaka - Bagian 3

RDKit Developers. (2026). The RDKit Book and Getting Started with the RDKit in Python.

Riniker, S., & Landrum, G. A. (2015). Better informed distance geometry: Using what we know to improve conformation generation. Journal of Chemical Information and Modeling, 55, 2562-2574.

Schrodinger, LLC. (2026). 3Dmol.js and py3Dmol documentation.

Wang, L.-P., & Song, C. (2016). Geometry optimization made simple with translation and rotation coordinates. The Journal of Chemical Physics, 144, 214108.

Catatan referensi: Dokumentasi perangkat lunak bersifat dinamis. Periksa dokumentasi resmi sebelum workshop atau penerbitan ulang.

Daftar Pustaka - Bagian 4

Heras-Domingo, J., & Garay-Ruiz, D. (2024). Pythonic Chemistry: The Beginner's Guide to Digital Chemistry. *Journal of Chemical Education*, 101(11), 4883-4891.

Hrubes, J., Jaros, A., Nemirovich, T., Tepla, M., & Petrzelova, S. (2024). Integrating Computational Chemistry into Secondary School Lessons. *Journal of Chemical Education*, 101(6), 2343-2353.

Kasmui. (2026). Tutorial Google Colab untuk Kimia Komputasi Dasar. Naskah tutorial tidak diterbitkan.

Kasmui. (2026). Rancangan Buku Tutorial Menjalankan Notebook Python di Google Colab. Naskah pengembangan.

Catatan referensi: Dokumentasi perangkat lunak bersifat dinamis. Periksa dokumentasi resmi sebelum workshop atau penerbitan ulang.

Daftar Pustaka - Bagian 5

Catatan referensi: Dokumentasi perangkat lunak bersifat dinamis. Periksa dokumentasi resmi sebelum workshop atau penerbitan ulang.

LAMPIRAN A - DAFTAR PERINTAH INSTALASI DAN PEMULIHAN RUNTIME

Instalasi dasar komputasi ilmiah digunakan untuk menyiapkan atau memulihkan lingkungan Google Colab. Jalankan perintah hanya pada runtime yang sesuai, lalu catat versi paket dan lakukan restart bila dependensi inti berubah.

Perintah

```
%pip -q install numpy pandas matplotlib scipy
```

Komponen	Kriteria
Sebelum	Periksa runtime, CUDA, dan paket yang sudah ada.
Sesudah	Uji impor dan cetak versi.
Dokumentasi	Simpan pip freeze atau environment.json.
Fallback	Gunakan CPU bila backend GPU tidak kompatibel.
Catatan: Ketersediaan GPU dan versi runtime berubah. Deteksi lingkungan selalu lebih aman daripada mengasumsikan jenis perangkat.	

Daftar Perintah Instalasi dan Pemulihan Runtime: Bagian 2

Instalasi kimia struktur dan visualisasi digunakan untuk menyiapkan atau memulihkan lingkungan Google Colab. Jalankan perintah hanya pada runtime yang sesuai, lalu catat versi paket dan lakukan restart bila dependensi inti berubah.

Perintah

```
%pip -q install rdkit py3Dmol
```

Komponen	Kriteria
Sebelum	Periksa runtime, CUDA, dan paket yang sudah ada.
Sesudah	Uji impor dan cetak versi.

Dokumentasi	Simpan pip freeze atau environment.json.
Fallback	Gunakan CPU bila backend GPU tidak kompatibel.
Catatan: Ketersediaan GPU dan versi runtime berubah. Deteksi lingkungan selalu lebih aman daripada mengasumsikan jenis perangkat.	

Daftar Perintah Instalasi dan Pemulihan Runtime: Bagian 3

Instalasi kimia kuantum CPU digunakan untuk menyiapkan atau memulihkan lingkungan Google Colab. Jalankan perintah hanya pada runtime yang sesuai, lalu catat versi paket dan lakukan restart bila dependensi inti berubah.

Perintah

```
%pip -q install --prefer-binary pyscf geometric
```

Komponen	Kriteria
Sebelum	Periksa runtime, CUDA, dan paket yang sudah ada.
Sesudah	Uji impor dan cetak versi.
Dokumentasi	Simpan pip freeze atau environment.json.
Fallback	Gunakan CPU bila backend GPU tidak kompatibel.
Catatan: Ketersediaan GPU dan versi runtime berubah. Deteksi lingkungan selalu lebih aman daripada mengasumsikan jenis perangkat.	

Daftar Perintah Instalasi dan Pemulihan Runtime: Bagian 4

Instalasi validasi notebook digunakan untuk menyiapkan atau memulihkan lingkungan Google Colab. Jalankan perintah hanya pada runtime yang sesuai, lalu catat versi paket dan lakukan restart bila dependensi inti berubah.

Perintah

```
%pip -q install nbformat nbclient
```

Komponen	Kriteria
Sebelum	Periksa runtime, CUDA, dan paket yang sudah ada.
Sesudah	Uji impor dan cetak versi.
Dokumentasi	Simpan pip freeze atau environment.json.
Fallback	Gunakan CPU bila backend GPU tidak kompatibel.
Catatan: Ketersediaan GPU dan versi runtime berubah. Deteksi lingkungan selalu lebih aman daripada mengasumsikan jenis perangkat.	

Daftar Perintah Instalasi dan Pemulihan Runtime: Bagian 5

Pemeriksaan versi paket digunakan untuk menyiapkan atau memulihkan lingkungan Google Colab. Jalankan perintah hanya pada runtime yang sesuai, lalu catat versi paket dan lakukan restart bila dependensi inti berubah.

Perintah

```
python -m pip list | sed -n "1,40p"
```

Komponen	Kriteria
Sebelum	Periksa runtime, CUDA, dan paket yang sudah ada.
Sesudah	Uji impor dan cetak versi.
Dokumentasi	Simpan pip freeze atau environment.json.
Fallback	Gunakan CPU bila backend GPU tidak kompatibel.
Catatan: Ketersediaan GPU dan versi runtime berubah. Deteksi lingkungan selalu lebih aman daripada mengasumsikan jenis perangkat.	

Daftar Perintah Instalasi dan Pemulihan Runtime: Bagian 6

Pembaruan pip digunakan untuk menyiapkan atau memulihkan lingkungan Google Colab. Jalankan perintah hanya pada runtime yang sesuai, lalu catat versi paket dan lakukan restart bila dependensi ini berubah.

Perintah

```
%pip -q install --upgrade pip
```

Komponen	Kriteria
Sebelum	Periksa runtime, CUDA, dan paket yang sudah ada.
Sesudah	Uji impor dan cetak versi.
Dokumentasi	Simpan pip freeze atau environment.json.
Fallback	Gunakan CPU bila backend GPU tidak kompatibel.
Catatan: Ketersediaan GPU dan versi runtime berubah. Deteksi lingkungan selalu lebih aman daripada mengasumsikan jenis perangkat.	

Daftar Perintah Instalasi dan Pemulihan Runtime: Bagian 7

Penguncian versi digunakan untuk menyiapkan atau memulihkan lingkungan Google Colab. Jalankan perintah hanya pada runtime yang sesuai, lalu catat versi paket dan lakukan restart bila dependensi ini berubah.

Perintah

```
%pip freeze > requirements-colab.txt
```

Komponen	Kriteria
Sebelum	Periksa runtime, CUDA, dan paket yang sudah ada.
Sesudah	Uji impor dan cetak versi.

Dokumentasi	Simpan pip freeze atau environment.json.
Fallback	Gunakan CPU bila backend GPU tidak kompatibel.
Catatan: Ketersediaan GPU dan versi runtime berubah. Deteksi lingkungan selalu lebih aman daripada mengasumsikan jenis perangkat.	

Daftar Perintah Instalasi dan Pemulihan Runtime: Bagian 8

Pemulihan runtime digunakan untuk menyiapkan atau memulihkan lingkungan Google Colab. Jalankan perintah hanya pada runtime yang sesuai, lalu catat versi paket dan lakukan restart bila dependensi inti berubah.

Perintah

```
Runtime > Restart session; jalankan kembali sel instalasi.
```

Komponen	Kriteria
Sebelum	Periksa runtime, CUDA, dan paket yang sudah ada.
Sesudah	Uji impor dan cetak versi.
Dokumentasi	Simpan pip freeze atau environment.json.
Fallback	Gunakan CPU bila backend GPU tidak kompatibel.
Catatan: Ketersediaan GPU dan versi runtime berubah. Deteksi lingkungan selalu lebih aman daripada mengasumsikan jenis perangkat.	

Daftar Perintah Instalasi dan Pemulihan Runtime: Bagian 9

Deteksi GPU digunakan untuk menyiapkan atau memulihkan lingkungan Google Colab. Jalankan perintah hanya pada runtime yang sesuai, lalu catat versi paket dan lakukan restart bila dependensi inti berubah.

Perintah

```
!nvidia-smi
```

Komponen	Kriteria
Sebelum	Periksa runtime, CUDA, dan paket yang sudah ada.
Sesudah	Uji impor dan cetak versi.
Dokumentasi	Simpan pip freeze atau environment.json.
Fallback	Gunakan CPU bila backend GPU tidak kompatibel.
Catatan: Ketersediaan GPU dan versi runtime berubah. Deteksi lingkungan selalu lebih aman daripada mengasumsikan jenis perangkat.	

Daftar Perintah Instalasi dan Pemulihan Runtime: Bagian 10

GPU4PySCF CUDA 12 digunakan untuk menyiapkan atau memulihkan lingkungan Google Colab. Jalankan perintah hanya pada runtime yang sesuai, lalu catat versi paket dan lakukan restart bila dependensi inti berubah.

Perintah

```
%pip -q install gpu4pyscf-cuda12x cutensor-cu12
```

Komponen	Kriteria
Sebelum	Periksa runtime, CUDA, dan paket yang sudah ada.
Sesudah	Uji impor dan cetak versi.
Dokumentasi	Simpan pip freeze atau environment.json.
Fallback	Gunakan CPU bila backend GPU tidak kompatibel.
Catatan: Ketersediaan GPU dan versi runtime berubah. Deteksi lingkungan selalu lebih aman daripada mengasumsikan jenis perangkat.	

Daftar Perintah Instalasi dan Pemulihan Runtime: Bagian 11

GPU4PySCF CUDA 11 digunakan untuk menyiapkan atau memulihkan lingkungan Google Colab. Jalankan perintah hanya pada runtime yang sesuai, lalu catat versi paket dan lakukan restart bila dependensi inti berubah.

Perintah

```
%pip -q install gpu4pyscf-cuda11x cutensor-cu11
```

Komponen	Kriteria
Sebelum	Periksa runtime, CUDA, dan paket yang sudah ada.
Sesudah	Uji impor dan cetak versi.
Dokumentasi	Simpan pip freeze atau environment.json.
Fallback	Gunakan CPU bila backend GPU tidak kompatibel.
Catatan: Ketersediaan GPU dan versi runtime berubah. Deteksi lingkungan selalu lebih aman daripada mengasumsikan jenis perangkat.	

Daftar Perintah Instalasi dan Pemulihan Runtime: Bagian 12

Pemeriksaan impor GPU digunakan untuk menyiapkan atau memulihkan lingkungan Google Colab. Jalankan perintah hanya pada runtime yang sesuai, lalu catat versi paket dan lakukan restart bila dependensi inti berubah.

Perintah

```
import gpu4pyscf; print("GPU4PySCF tersedia")
```

Komponen	Kriteria
Sebelum	Periksa runtime, CUDA, dan paket yang sudah ada.
Sesudah	Uji impor dan cetak versi.
Dokumentasi	Simpan pip freeze atau environment.json.

Fallback

Gunakan CPU bila backend GPU tidak kompatibel.

Catatan: Ketersediaan GPU dan versi runtime berubah. Deteksi lingkungan selalu lebih aman daripada mengasumsikan jenis perangkat.

Daftar Perintah Instalasi dan Pemulihan Runtime: Bagian 13

Instalasi ulang bersih digunakan untuk menyiapkan atau memulihkan lingkungan Google Colab. Jalankan perintah hanya pada runtime yang sesuai, lalu catat versi paket dan lakukan restart bila dependensi inti berubah.

Perintah

```
%pip uninstall -y gpu4pyscf-cuda11x gpu4pyscf-cuda12x
```

Komponen	Kriteria
Sebelum	Periksa runtime, CUDA, dan paket yang sudah ada.
Sesudah	Uji impor dan cetak versi.
Dokumentasi	Simpan pip freeze atau environment.json.
Fallback	Gunakan CPU bila backend GPU tidak kompatibel.

Catatan: Ketersediaan GPU dan versi runtime berubah. Deteksi lingkungan selalu lebih aman daripada mengasumsikan jenis perangkat.

Daftar Perintah Instalasi dan Pemulihan Runtime: Bagian 14

Folder kerja digunakan untuk menyiapkan atau memulihkan lingkungan Google Colab. Jalankan perintah hanya pada runtime yang sesuai, lalu catat versi paket dan lakukan restart bila dependensi inti berubah.

Perintah

```
from pathlib import Path; Path("/content/proyek").mkdir(exist_ok=True)
```

Komponen	Kriteria

Sebelum	Periksa runtime, CUDA, dan paket yang sudah ada.
Sesudah	Uji impor dan cetak versi.
Dokumentasi	Simpan pip freeze atau environment.json.
Fallback	Gunakan CPU bila backend GPU tidak kompatibel.

Catatan: Ketersediaan GPU dan versi runtime berubah. Deteksi lingkungan selalu lebih aman daripada mengasumsikan jenis perangkat.

Daftar Perintah Instalasi dan Pemulihan Runtime: Bagian 15

Arsip lingkungan digunakan untuk menyiapkan atau memulihkan lingkungan Google Colab. Jalankan perintah hanya pada runtime yang sesuai, lalu catat versi paket dan lakukan restart bila dependensi ini berubah.

Perintah

```
!python --version && !pip freeze > environment.txt
```

Komponen	Kriteria
Sebelum	Periksa runtime, CUDA, dan paket yang sudah ada.
Sesudah	Uji impor dan cetak versi.
Dokumentasi	Simpan pip freeze atau environment.json.
Fallback	Gunakan CPU bila backend GPU tidak kompatibel.

Catatan: Ketersediaan GPU dan versi runtime berubah. Deteksi lingkungan selalu lebih aman daripada mengasumsikan jenis perangkat.

LAMPIRAN B - DAFTAR KESALAHAN UMUM DAN PEMECAHANNYA

Gejala: ModuleNotFoundError

Paket belum dipasang pada runtime aktif. Jalankan sel %pip dan periksa ejaan modul.

642. Salin pesan kesalahan lengkap dan identifikasi sel pertama yang gagal.
643. Cetak versi, current working directory, tipe, shape, dan parameter terkait.
644. Ubah satu faktor saja, lalu jalankan ulang dari titik yang jelas.
645. Setelah berhasil, restart session dan Run all untuk memastikan perbaikan tidak bergantung pada state lama.

Komponen	Kriteria
Diagnosis	Penyebab didukung bukti dari pesan atau pemeriksaan.
Perbaikan	Tidak menutupi error dengan try-except kosong.
Pengujian	Kasus benar dan kasus salah diuji.
Pelaporan	Perubahan parameter/versi dicatat.

Daftar Kesalahan Umum dan Pemecahannya: Bagian 2

Gejala: ImportError setelah upgrade

Restart session agar proses Python memuat pustaka yang baru.

646. Salin pesan kesalahan lengkap dan identifikasi sel pertama yang gagal.
647. Cetak versi, current working directory, tipe, shape, dan parameter terkait.
648. Ubah satu faktor saja, lalu jalankan ulang dari titik yang jelas.
649. Setelah berhasil, restart session dan Run all untuk memastikan perbaikan tidak bergantung pada state lama.

Komponen	Kriteria
Diagnosis	Penyebab didukung bukti dari pesan atau pemeriksaan.

Perbaikan	Tidak menutupi error dengan try-except kosong.
Pengujian	Kasus benar dan kasus salah diuji.
Pelaporan	Perubahan parameter/versi dicatat.

Daftar Kesalahan Umum dan Pemecahannya: Bagian 3

Gejala: Konflik NumPy

Catat versi, restart, dan gunakan kombinasi paket yang kompatibel; hindari upgrade acak.

650. Salin pesan kesalahan lengkap dan identifikasi sel pertama yang gagal.
651. Cetak versi, current working directory, tipe, shape, dan parameter terkait.
652. Ubah satu faktor saja, lalu jalankan ulang dari titik yang jelas.
653. Setelah berhasil, restart session dan Run all untuk memastikan perbaikan tidak bergantung pada state lama.

Komponen	Kriteria
Diagnosis	Penyebab didukung bukti dari pesan atau pemeriksaan.
Perbaikan	Tidak menutupi error dengan try-except kosong.
Pengujian	Kasus benar dan kasus salah diuji.
Pelaporan	Perubahan parameter/versi dicatat.

Daftar Kesalahan Umum dan Pemecahannya: Bagian 4

Gejala: nvidia-smi tidak ditemukan

Runtime bukan GPU atau GPU tidak tersedia. Gunakan CPU dan laporkan fallback.

654. Salin pesan kesalahan lengkap dan identifikasi sel pertama yang gagal.
655. Cetak versi, current working directory, tipe, shape, dan parameter terkait.
656. Ubah satu faktor saja, lalu jalankan ulang dari titik yang jelas.
657. Setelah berhasil, restart session dan Run all untuk memastikan perbaikan tidak bergantung pada state lama.

Komponen	Kriteria
Diagnosis	Penyebab didukung bukti dari pesan atau pemeriksaan.
Perbaikan	Tidak menutupi error dengan try-except kosong.
Pengujian	Kasus benar dan kasus salah diuji.
Pelaporan	Perubahan parameter/versi dicatat.

Daftar Kesalahan Umum dan Pemecahannya: Bagian 5

Gejala: CUDA package mismatch

Hapus paket GPU yang salah dan pasang varian sesuai CUDA runtime.

658. Salin pesan kesalahan lengkap dan identifikasi sel pertama yang gagal.
659. Cetak versi, current working directory, tipe, shape, dan parameter terkait.
660. Ubah satu faktor saja, lalu jalankan ulang dari titik yang jelas.
661. Setelah berhasil, restart session dan Run all untuk memastikan perbaikan tidak bergantung pada state lama.

Komponen	Kriteria
Diagnosis	Penyebab didukung bukti dari pesan atau pemeriksaan.
Perbaikan	Tidak menutupi error dengan try-except kosong.
Pengujian	Kasus benar dan kasus salah diuji.
Pelaporan	Perubahan parameter/versi dicatat.

Daftar Kesalahan Umum dan Pemecahannya: Bagian 6

Gejala: CuPy tidak dapat memuat library

Periksa versi CUDA, driver, dan paket CuPy/GPU4PySCF.

662. Salin pesan kesalahan lengkap dan identifikasi sel pertama yang gagal.
663. Cetak versi, current working directory, tipe, shape, dan parameter terkait.
664. Ubah satu faktor saja, lalu jalankan ulang dari titik yang jelas.
665. Setelah berhasil, restart session dan Run all untuk memastikan perbaikan tidak bergantung pada state lama.

Komponen	Kriteria
Diagnosis	Penyebab didukung bukti dari pesan atau pemeriksaan.
Perbaikan	Tidak menutupi error dengan try-except kosong.
Pengujian	Kasus benar dan kasus salah diuji.
Pelaporan	Perubahan parameter/versi dicatat.

Daftar Kesalahan Umum dan Pemecahannya: Bagian 7

Gejala: SCF not converged

Naikkan max_cycle, periksa geometri, gunakan tebakan/solver Newton, dan jangan menafsirkan energi gagal.

666. Salin pesan kesalahan lengkap dan identifikasi sel pertama yang gagal.
667. Cetak versi, current working directory, tipe, shape, dan parameter terkait.
668. Ubah satu faktor saja, lalu jalankan ulang dari titik yang jelas.
669. Setelah berhasil, restart session dan Run all untuk memastikan perbaikan tidak bergantung pada state lama.

Komponen	Kriteria
Diagnosis	Penyebab didukung bukti dari pesan atau pemeriksaan.
Perbaikan	Tidak menutupi error dengan try-except kosong.
Pengujian	Kasus benar dan kasus salah diuji.
Pelaporan	Perubahan parameter/versi dicatat.

Daftar Kesalahan Umum dan Pemecahannya: Bagian 8**Gejala:** Basis set tidak tersedia

Periksa ejaan dan dukungan unsur pada basis set.

670. Salin pesan kesalahan lengkap dan identifikasi sel pertama yang gagal.
671. Cetak versi, current working directory, tipe, shape, dan parameter terkait.
672. Ubah satu faktor saja, lalu jalankan ulang dari titik yang jelas.
673. Setelah berhasil, restart session dan Run all untuk memastikan perbaikan tidak bergantung pada state lama.

Komponen	Kriteria
Diagnosis	Penyebab didukung bukti dari pesan atau pemeriksaan.
Perbaikan	Tidak menutupi error dengan try-except kosong.
Pengujian	Kasus benar dan kasus salah diuji.
Pelaporan	Perubahan parameter/versi dicatat.

Daftar Kesalahan Umum dan Pemecahannya: Bagian 9**Gejala:** Charge-spin tidak konsisten

Hitung jumlah elektron dan pastikan (Nelec-spin) genap.

674. Salin pesan kesalahan lengkap dan identifikasi sel pertama yang gagal.
675. Cetak versi, current working directory, tipe, shape, dan parameter terkait.
676. Ubah satu faktor saja, lalu jalankan ulang dari titik yang jelas.
677. Setelah berhasil, restart session dan Run all untuk memastikan perbaikan tidak bergantung pada state lama.

Komponen	Kriteria
Diagnosis	Penyebab didukung bukti dari pesan atau pemeriksaan.
Perbaikan	Tidak menutupi error dengan try-except kosong.

Pengujian	Kasus benar dan kasus salah diuji.
Pelaporan	Perubahan parameter/versi dicatat.

Daftar Kesalahan Umum dan Pemecahannya: Bagian 10

Gejala: Geometri mengandung NaN

Hentikan proses, periksa konformer dan semua koordinat.

678. Salin pesan kesalahan lengkap dan identifikasi sel pertama yang gagal.
679. Cetak versi, current working directory, tipe, shape, dan parameter terkait.
680. Ubah satu faktor saja, lalu jalankan ulang dari titik yang jelas.
681. Setelah berhasil, restart session dan Run all untuk memastikan perbaikan tidak bergantung pada state lama.

Komponen	Kriteria
Diagnosis	Penyebab didukung bukti dari pesan atau pemeriksaan.
Perbaikan	Tidak menutupi error dengan try-except kosong.
Pengujian	Kasus benar dan kasus salah diuji.
Pelaporan	Perubahan parameter/versi dicatat.

Daftar Kesalahan Umum dan Pemecahannya: Bagian 11

Gejala: Embedding RDKit gagal

Gunakan seed lain, random coordinates, atau periksa fragmen dan stereokimia.

682. Salin pesan kesalahan lengkap dan identifikasi sel pertama yang gagal.
683. Cetak versi, current working directory, tipe, shape, dan parameter terkait.
684. Ubah satu faktor saja, lalu jalankan ulang dari titik yang jelas.
685. Setelah berhasil, restart session dan Run all untuk memastikan perbaikan tidak bergantung pada state lama.

Komponen	Kriteria

Diagnosis	Penyebab didukung bukti dari pesan atau pemeriksaan.
Perbaikan	Tidak menutupi error dengan try-except kosong.
Pengujian	Kasus benar dan kasus salah diuji.
Pelaporan	Perubahan parameter/versi dicatat.

Daftar Kesalahan Umum dan Pemecahannya: Bagian 12

Gejala: MMFF parameter tidak lengkap

Gunakan UFF sebagai fallback dan laporkan pilihan tersebut.

686. Salin pesan kesalahan lengkap dan identifikasi sel pertama yang gagal.
687. Cetak versi, current working directory, tipe, shape, dan parameter terkait.
688. Ubah satu faktor saja, lalu jalankan ulang dari titik yang jelas.
689. Setelah berhasil, restart session dan Run all untuk memastikan perbaikan tidak bergantung pada state lama.

Komponen	Kriteria
Diagnosis	Penyebab didukung bukti dari pesan atau pemeriksaan.
Perbaikan	Tidak menutupi error dengan try-except kosong.
Pengujian	Kasus benar dan kasus salah diuji.
Pelaporan	Perubahan parameter/versi dicatat.

Daftar Kesalahan Umum dan Pemecahannya: Bagian 13

Gejala: Optimasi melebihi maxsteps

Periksa geometri awal, SCF, koordinat, dan kriteria optimasi.

690. Salin pesan kesalahan lengkap dan identifikasi sel pertama yang gagal.
691. Cetak versi, current working directory, tipe, shape, dan parameter terkait.

692. Ubah satu faktor saja, lalu jalankan ulang dari titik yang jelas.
693. Setelah berhasil, restart session dan Run all untuk memastikan perbaikan tidak bergantung pada state lama.

Komponen	Kriteria
Diagnosis	Penyebab didukung bukti dari pesan atau pemeriksaan.
Perbaikan	Tidak menutupi error dengan try-except kosong.
Pengujian	Kasus benar dan kasus salah diuji.
Pelaporan	Perubahan parameter/versi dicatat.

Daftar Kesalahan Umum dan Pemecahannya: Bagian 14

Gejala: Frekuensi imajiner

Struktur mungkin bukan minimum; periksa mode dan optimasi ulang.

694. Salin pesan kesalahan lengkap dan identifikasi sel pertama yang gagal.
695. Cetak versi, current working directory, tipe, shape, dan parameter terkait.
696. Ubah satu faktor saja, lalu jalankan ulang dari titik yang jelas.
697. Setelah berhasil, restart session dan Run all untuk memastikan perbaikan tidak bergantung pada state lama.

Komponen	Kriteria
Diagnosis	Penyebab didukung bukti dari pesan atau pemeriksaan.
Perbaikan	Tidak menutupi error dengan try-except kosong.
Pengujian	Kasus benar dan kasus salah diuji.
Pelaporan	Perubahan parameter/versi dicatat.

Daftar Kesalahan Umum dan Pemecahannya: Bagian 15**Gejala:** FileNotFoundError

Gunakan pathlib, cetak cwd, dan periksa nama berkas yang diunggah.

698. Salin pesan kesalahan lengkap dan identifikasi sel pertama yang gagal.
699. Cetak versi, current working directory, tipe, shape, dan parameter terkait.
700. Ubah satu faktor saja, lalu jalankan ulang dari titik yang jelas.
701. Setelah berhasil, restart session dan Run all untuk memastikan perbaikan tidak bergantung pada state lama.

Komponen	Kriteria
Diagnosis	Penyebab didukung bukti dari pesan atau pemeriksaan.
Perbaikan	Tidak menutupi error dengan try-except kosong.
Pengujian	Kasus benar dan kasus salah diuji.
Pelaporan	Perubahan parameter/versi dicatat.

Daftar Kesalahan Umum dan Pemecahannya: Bagian 16**Gejala:** PermissionError di Drive

Pastikan Drive ter-mount dan jalur berada pada folder yang dapat ditulis.

702. Salin pesan kesalahan lengkap dan identifikasi sel pertama yang gagal.
703. Cetak versi, current working directory, tipe, shape, dan parameter terkait.
704. Ubah satu faktor saja, lalu jalankan ulang dari titik yang jelas.
705. Setelah berhasil, restart session dan Run all untuk memastikan perbaikan tidak bergantung pada state lama.

Komponen	Kriteria
Diagnosis	Penyebab didukung bukti dari pesan atau pemeriksaan.
Perbaikan	Tidak menutupi error dengan try-except kosong.

Pengujian	Kasus benar dan kasus salah diuji.
Pelaporan	Perubahan parameter/versi dicatat.

Daftar Kesalahan Umum dan Pemecahannya: Bagian 17

Gejala: Runtime disconnected

Simpan checkpoint dan berkas penting secara berkala ke Drive.

706. Salin pesan kesalahan lengkap dan identifikasi sel pertama yang gagal.
707. Cetak versi, current working directory, tipe, shape, dan parameter terkait.
708. Ubah satu faktor saja, lalu jalankan ulang dari titik yang jelas.
709. Setelah berhasil, restart session dan Run all untuk memastikan perbaikan tidak bergantung pada state lama.

Komponen	Kriteria
Diagnosis	Penyebab didukung bukti dari pesan atau pemeriksaan.
Perbaikan	Tidak menutupi error dengan try-except kosong.
Pengujian	Kasus benar dan kasus salah diuji.
Pelaporan	Perubahan parameter/versi dicatat.

Daftar Kesalahan Umum dan Pemecahannya: Bagian 18

Gejala: Notebook output lama

Clear all outputs, restart, dan Run all.

710. Salin pesan kesalahan lengkap dan identifikasi sel pertama yang gagal.
711. Cetak versi, current working directory, tipe, shape, dan parameter terkait.
712. Ubah satu faktor saja, lalu jalankan ulang dari titik yang jelas.
713. Setelah berhasil, restart session dan Run all untuk memastikan perbaikan tidak bergantung pada state lama.

Komponen	Kriteria

Diagnosis	Penyebab didukung bukti dari pesan atau pemeriksaan.
Perbaikan	Tidak menutupi error dengan try-except kosong.
Pengujian	Kasus benar dan kasus salah diuji.
Pelaporan	Perubahan parameter/versi dicatat.

Daftar Kesalahan Umum dan Pemecahannya: Bagian 19

Gejala: Cell execution order kacau

Jangan mengandalkan variabel dari sel yang belum dijalankan.

714. Salin pesan kesalahan lengkap dan identifikasi sel pertama yang gagal.
715. Cetak versi, current working directory, tipe, shape, dan parameter terkait.
716. Ubah satu faktor saja, lalu jalankan ulang dari titik yang jelas.
717. Setelah berhasil, restart session dan Run all untuk memastikan perbaikan tidak bergantung pada state lama.

Komponen	Kriteria
Diagnosis	Penyebab didukung bukti dari pesan atau pemeriksaan.
Perbaikan	Tidak menutupi error dengan try-except kosong.
Pengujian	Kasus benar dan kasus salah diuji.
Pelaporan	Perubahan parameter/versi dicatat.

Daftar Kesalahan Umum dan Pemecahannya: Bagian 20

Gejala: JSON serialization error

Ubah tipe NumPy menjadi int/float Python sebelum json.dump.

718. Salin pesan kesalahan lengkap dan identifikasi sel pertama yang gagal.
719. Cetak versi, current working directory, tipe, shape, dan parameter terkait.

720. Ubah satu faktor saja, lalu jalankan ulang dari titik yang jelas.
721. Setelah berhasil, restart session dan Run all untuk memastikan perbaikan tidak bergantung pada state lama.

Komponen	Kriteria
Diagnosis	Penyebab didukung bukti dari pesan atau pemeriksaan.
Perbaikan	Tidak menutupi error dengan try-except kosong.
Pengujian	Kasus benar dan kasus salah diuji.
Pelaporan	Perubahan parameter/versi dicatat.

Daftar Kesalahan Umum dan Pemecahannya: Bagian 21

Gejala: XYZ jumlah atom salah

Baris pertama harus sama dengan jumlah baris koordinat.

722. Salin pesan kesalahan lengkap dan identifikasi sel pertama yang gagal.
723. Cetak versi, current working directory, tipe, shape, dan parameter terkait.
724. Ubah satu faktor saja, lalu jalankan ulang dari titik yang jelas.
725. Setelah berhasil, restart session dan Run all untuk memastikan perbaikan tidak bergantung pada state lama.

Komponen	Kriteria
Diagnosis	Penyebab didukung bukti dari pesan atau pemeriksaan.
Perbaikan	Tidak menutupi error dengan try-except kosong.
Pengujian	Kasus benar dan kasus salah diuji.
Pelaporan	Perubahan parameter/versi dicatat.

Daftar Kesalahan Umum dan Pemecahannya: Bagian 22**Gejala:** py3Dmol blank

Periksa format XYZ/MOL, ukuran kanvas, dan pemanggilan `view.show()`.

726. Salin pesan kesalahan lengkap dan identifikasi sel pertama yang gagal.
727. Cetak versi, current working directory, tipe, shape, dan parameter terkait.
728. Ubah satu faktor saja, lalu jalankan ulang dari titik yang jelas.
729. Setelah berhasil, restart session dan Run all untuk memastikan perbaikan tidak bergantung pada state lama.

Komponen	Kriteria
Diagnosis	Penyebab didukung bukti dari pesan atau pemeriksaan.
Perbaikan	Tidak menutupi error dengan try-except kosong.
Pengujian	Kasus benar dan kasus salah diuji.
Pelaporan	Perubahan parameter/versi dicatat.

Daftar Kesalahan Umum dan Pemecahannya: Bagian 23**Gejala:** Energi berbeda jauh

Periksa geometri, metode, basis set, muatan, spin, dan satuan.

730. Salin pesan kesalahan lengkap dan identifikasi sel pertama yang gagal.
731. Cetak versi, current working directory, tipe, shape, dan parameter terkait.
732. Ubah satu faktor saja, lalu jalankan ulang dari titik yang jelas.
733. Setelah berhasil, restart session dan Run all untuk memastikan perbaikan tidak bergantung pada state lama.

Komponen	Kriteria
Diagnosis	Penyebab didukung bukti dari pesan atau pemeriksaan.
Perbaikan	Tidak menutupi error dengan try-except kosong.

Pengujian	Kasus benar dan kasus salah diuji.
Pelaporan	Perubahan parameter/versi dicatat.

Daftar Kesalahan Umum dan Pemecahannya: Bagian 24

Gejala: GPU aktif tetapi lambat

Ukuran sistem terlalu kecil, transfer data dominan, atau kode masih memakai CPU.

734. Salin pesan kesalahan lengkap dan identifikasi sel pertama yang gagal.
735. Cetak versi, current working directory, tipe, shape, dan parameter terkait.
736. Ubah satu faktor saja, lalu jalankan ulang dari titik yang jelas.
737. Setelah berhasil, restart session dan Run all untuk memastikan perbaikan tidak bergantung pada state lama.

Komponen	Kriteria
Diagnosis	Penyebab didukung bukti dari pesan atau pemeriksaan.
Perbaikan	Tidak menutupi error dengan try-except kosong.
Pengujian	Kasus benar dan kasus salah diuji.
Pelaporan	Perubahan parameter/versi dicatat.

Daftar Kesalahan Umum dan Pemecahannya: Bagian 25

Gejala: Arsip ZIP terlalu besar

Batasi file sementara, output verbose, dan checkpoint yang tidak diperlukan.

738. Salin pesan kesalahan lengkap dan identifikasi sel pertama yang gagal.
739. Cetak versi, current working directory, tipe, shape, dan parameter terkait.
740. Ubah satu faktor saja, lalu jalankan ulang dari titik yang jelas.
741. Setelah berhasil, restart session dan Run all untuk memastikan perbaikan tidak bergantung pada state lama.

Komponen	Kriteria

Diagnosis	Penyebab didukung bukti dari pesan atau pemeriksaan.
Perbaikan	Tidak menutupi error dengan try-except kosong.
Pengujian	Kasus benar dan kasus salah diuji.
Pelaporan	Perubahan parameter/versi dicatat.

LAMPIRAN C - FORMAT HASIL: XYZ, CSV, JSON, TXT, MOLDEN, DAN ZIP

Format TXT - Teks ringkasan yang mudah dibaca manusia; gunakan encoding UTF-8 dan kunci: nilai.

```
# Contoh penamaan
filename = "proyek_txt."
print("Gunakan nama, metode, dan tanggal pada metadata.")
```

Komponen	Kriteria
Keterbacaan	Dapat dibuka dengan alat yang sesuai.
Konsistensi	Jumlah record, atom, atau kolom benar.
Satuan	Dinyatakan pada header, komentar, atau metadata.
Provenance	Metode, basis, versi, dan tanggal tercatat.

Format Hasil: XYZ, CSV, JSON, TXT, Molden, dan ZIP: Bagian 2

Format CSV - Data tabular; simpan header, satuan pada nama kolom, dan indeks hanya bila diperlukan.

```
atom,x_A,y_A,z_A
O,0.000000,0.000000,0.000000
H,0.000000,0.757000,0.587000
H,0.000000,-0.757000,0.587000
```

Komponen	Kriteria
Keterbacaan	Dapat dibuka dengan alat yang sesuai.
Konsistensi	Jumlah record, atom, atau kolom benar.
Satuan	Dinyatakan pada header, komentar, atau metadata.
Provenance	Metode, basis, versi, dan tanggal tercatat.

Format Hasil: XYZ, CSV, JSON, TXT, Molden, dan ZIP: Bagian 3

Format JSON - Metadata terstruktur; konversi tipe NumPy dan gunakan indent untuk keterbacaan.

```
{  
  "molecule": "H2O",  
  "method": "B3LYP",  
  "basis": "def2-TZVP",  
  "converged": true,  
  "energy_hartree": -76.4  
}
```

Komponen	Kriteria
Keterbacaan	Dapat dibuka dengan alat yang sesuai.
Konsistensi	Jumlah record, atom, atau kolom benar.
Satuan	Dinyatakan pada header, komentar, atau metadata.
Provenance	Metode, basis, versi, dan tanggal tercatat.

Format Hasil: XYZ, CSV, JSON, TXT, Molden, dan ZIP: Bagian 4

Format XYZ - Jumlah atom, baris komentar, lalu simbol dan x y z dalam Angstrom.

```
3  
H2O B3LYP/def2-TZVP  
O 0.000000 0.000000 0.000000  
H 0.000000 0.757000 0.587000  
H 0.000000 -0.757000 0.587000
```

Komponen	Kriteria
Keterbacaan	Dapat dibuka dengan alat yang sesuai.
Konsistensi	Jumlah record, atom, atau kolom benar.
Satuan	Dinyatakan pada header, komentar, atau metadata.
Provenance	Metode, basis, versi, dan tanggal tercatat.

Format Hasil: XYZ, CSV, JSON, TXT, Molden, dan ZIP: Bagian 5

Format MOL/SDF - Menyimpan konektivitas, koordinat, dan dapat memuat banyak record pada SDF.

```
# Contoh penamaan
filename = "proyek_mol_sdf."
print("Gunakan nama, metode, dan tanggal pada metadata.")
```

Komponen	Kriteria
Keterbacaan	Dapat dibuka dengan alat yang sesuai.
Konsistensi	Jumlah record, atom, atau kolom benar.
Satuan	Dinyatakan pada header, komentar, atau metadata.
Provenance	Metode, basis, versi, dan tanggal tercatat.

Format Hasil: XYZ, CSV, JSON, TXT, Molden, dan ZIP: Bagian 6

Format Molden - Format visualisasi orbital; pastikan program pembaca kompatibel.

```
# Contoh penamaan
filename = "proyek_molden."
print("Gunakan nama, metode, dan tanggal pada metadata.")
```

Komponen	Kriteria
Keterbacaan	Dapat dibuka dengan alat yang sesuai.
Konsistensi	Jumlah record, atom, atau kolom benar.
Satuan	Dinyatakan pada header, komentar, atau metadata.
Provenance	Metode, basis, versi, dan tanggal tercatat.

Format Hasil: XYZ, CSV, JSON, TXT, Molden, dan ZIP: Bagian 7

Format PNG - Grafik raster; gunakan dpi dan bbox_inches yang memadai.

```
# Contoh penamaan
filename = "proyek_png."
print("Gunakan nama, metode, dan tanggal pada metadata.")
```

Komponen	Kriteria
Keterbacaan	Dapat dibuka dengan alat yang sesuai.
Konsistensi	Jumlah record, atom, atau kolom benar.
Satuan	Dinyatakan pada header, komentar, atau metadata.
Provenance	Metode, basis, versi, dan tanggal tercatat.

Format Hasil: XYZ, CSV, JSON, TXT, Molden, dan ZIP: Bagian 8

Format SVG - Grafik vektor; cocok untuk publikasi dan penyuntingan.

```
# Contoh penamaan
filename = "proyek_svg."
print("Gunakan nama, metode, dan tanggal pada metadata.")
```

Komponen	Kriteria
Keterbacaan	Dapat dibuka dengan alat yang sesuai.
Konsistensi	Jumlah record, atom, atau kolom benar.
Satuan	Dinyatakan pada header, komentar, atau metadata.
Provenance	Metode, basis, versi, dan tanggal tercatat.

Format Hasil: XYZ, CSV, JSON, TXT, Molden, dan ZIP: Bagian 9

Format LOG - Jejak proses, parameter, warning, dan status konvergensi.

```
# Contoh penamaan
filename = "proyek_log."
print("Gunakan nama, metode, dan tanggal pada metadata.")
```

Komponen	Kriteria

Keterbacaan	Dapat dibuka dengan alat yang sesuai.
Konsistensi	Jumlah record, atom, atau kolom benar.
Satuan	Dinyatakan pada header, komentar, atau metadata.
Provenance	Metode, basis, versi, dan tanggal tercatat.

Format Hasil: XYZ, CSV, JSON, TXT, Molden, dan ZIP: Bagian 10

Format IPYNB - Dokumen JSON yang berisi sel, output, dan metadata notebook.

```
# Contoh penamaan
filename = "proyek_ipynb."
print("Gunakan nama, metode, dan tanggal pada metadata.")
```

Komponen	Kriteria
Keterbacaan	Dapat dibuka dengan alat yang sesuai.
Konsistensi	Jumlah record, atom, atau kolom benar.
Satuan	Dinyatakan pada header, komentar, atau metadata.
Provenance	Metode, basis, versi, dan tanggal tercatat.

Format Hasil: XYZ, CSV, JSON, TXT, Molden, dan ZIP: Bagian 11

Format ZIP - Arsip pengiriman; sertakan README, parameter, hasil, dan checksum bila perlu.

```
# Contoh penamaan
filename = "proyek_zip."
print("Gunakan nama, metode, dan tanggal pada metadata.")
```

Komponen	Kriteria
Keterbacaan	Dapat dibuka dengan alat yang sesuai.
Konsistensi	Jumlah record, atom, atau kolom benar.

Satuan	Dinyatakan pada header, komentar, atau metadata.
Provenance	Metode, basis, versi, dan tanggal tercatat.

Format Hasil: XYZ, CSV, JSON, TXT, Molden, dan ZIP: Bagian 12

Format README - Menjelaskan cara menjalankan, struktur folder, versi, dan status pengujian.

```
# Contoh penamaan
filename = "proyek_readme."
print("Gunakan nama, metode, dan tanggal pada metadata.")
```

Komponen	Kriteria
Keterbacaan	Dapat dibuka dengan alat yang sesuai.
Konsistensi	Jumlah record, atom, atau kolom benar.
Satuan	Dinyatakan pada header, komentar, atau metadata.
Provenance	Metode, basis, versi, dan tanggal tercatat.

Format Hasil: XYZ, CSV, JSON, TXT, Molden, dan ZIP: Bagian 13

Format requirements.txt - Daftar versi paket untuk reproduksibilitas.

```
# Contoh penamaan
filename = "proyek_requirements.txt."
print("Gunakan nama, metode, dan tanggal pada metadata.")
```

Komponen	Kriteria
Keterbacaan	Dapat dibuka dengan alat yang sesuai.
Konsistensi	Jumlah record, atom, atau kolom benar.
Satuan	Dinyatakan pada header, komentar, atau metadata.

Provenance	Metode, basis, versi, dan tanggal tercatat.

Format Hasil: XYZ, CSV, JSON, TXT, Molden, dan ZIP: Bagian 14

Format environment.json - Ringkasan runtime, perangkat, pustaka, dan waktu.

```
# Contoh penamaan
filename = "proyek_environment.json."
print("Gunakan nama, metode, dan tanggal pada metadata.")
```

Komponen	Kriteria
Keterbacaan	Dapat dibuka dengan alat yang sesuai.
Konsistensi	Jumlah record, atom, atau kolom benar.
Satuan	Dinyatakan pada header, komentar, atau metadata.
Provenance	Metode, basis, versi, dan tanggal tercatat.

Format Hasil: XYZ, CSV, JSON, TXT, Molden, dan ZIP: Bagian 15

Format manifest.csv - Daftar seluruh berkas hasil, ukuran, fungsi, dan checksum.

```
# Contoh penamaan
filename = "proyek_manifest.csv."
print("Gunakan nama, metode, dan tanggal pada metadata.")
```

Komponen	Kriteria
Keterbacaan	Dapat dibuka dengan alat yang sesuai.
Konsistensi	Jumlah record, atom, atau kolom benar.
Satuan	Dinyatakan pada header, komentar, atau metadata.
Provenance	Metode, basis, versi, dan tanggal tercatat.

LAMPIRAN D - LIMA NOTEBOOK LENGKAP SIAP GOOGLE COLAB

Notebook: 01-dasar-google-colab-python.ipynb

Notebook ini dirancang sebagai dokumen .ipynb yang sah, terdiri atas sel Markdown dan sel kode. Struktur halaman lampiran merepresentasikan urutan sel yang harus dijalankan dari atas ke bawah.

Komponen	Kriteria
Bagian	Judul dan tujuan
Instalasi	Paket dan versi
Konfigurasi	Parameter yang dapat diubah
Perhitungan	Fungsi dan proses utama
Validasi	Format, sintaks, kimia, numerik
Ekspor	TXT, CSV, JSON, XYZ, ZIP

Lima Notebook Lengkap Siap Google Colab: Bagian 2

Notebook: 01-dasar-google-colab-python.ipynb

Sel inti: Python sebagai bahasa komputasi ilmiah. Kode berikut merupakan pola utama yang digunakan dalam notebook.

```
import sys
import math

print("Versi Python:", sys.version.split()[0])
energi_hartree = -1.117349
energi_ev = energi_hartree * 27.211386245988
print(f"Energi = {energi_ev:.6f} eV")
```

Validasi sel: Cetak status, periksa nilai finite, dan jangan meneruskan hasil bila syarat minimum gagal.

Lima Notebook Lengkap Siap Google Colab: Bagian 3

Notebook: 01-dasar-google-colab-python.ipynb

Sel inti: NumPy, Pandas, dan Matplotlib. Kode berikut merupakan pola utama yang digunakan dalam notebook.

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

x = np.linspace(0.4, 2.0, 17)
y = (x - 0.74)**2 - 1.1
df = pd.DataFrame({"jarak_A": x, "energi_rel": y})
print(df.head())

plt.plot(x, y, marker="o")
plt.xlabel("Jarak H-H (Å)")
plt.ylabel("Energi relatif")
plt.savefig("kurva_energi.png", dpi=200, bbox_inches="tight")
```

Validasi sel: Cetak status, periksa nilai finite, dan jangan meneruskan hasil bila syarat minimum gagal.

Lima Notebook Lengkap Siap Google Colab: Bagian 4

Notebook: 01-dasar-google-colab-python.ipynb

Sel inti: Mengunggah dan mengunduh berkas. Kode berikut merupakan pola utama yang digunakan dalam notebook.

```
from pathlib import Path
from google.colab import files

unggah = files.upload()
for nama in unggahan:
    print("Berkas diterima:", nama)

Path("hasil.txt").write_text("Contoh hasil Colab", encoding="utf-8")
files.download("hasil.txt")
```

Validasi sel: Cetak status, periksa nilai finite, dan jangan meneruskan hasil bila syarat minimum gagal.

Lima Notebook Lengkap Siap Google Colab: Bagian 5

Notebook: 01-dasar-google-colab-python.ipynb

Sel pemeriksaan lingkungan

```
import sys, platform, json, time
from importlib.metadata import version, PackageNotFoundError

def package_version(name):
    try: return version(name)
    except PackageNotFoundError: return None

env = {"python":sys.version, "platform":platform.platform(),
       "timestamp":time.strftime("%Y-%m-%d %H:%M:%S"),
       "packages":{p:package_version(p) for p in
["numpy","rdkit","pyscf","nbformat"]}}
print(json.dumps(env, indent=2))
```

Lima Notebook Lengkap Siap Google Colab: Bagian 6

Notebook: 01-dasar-google-colab-python.ipynb

Sel validasi notebook

```
import ast, nbformat
nb = nbformat.read("01-dasar-google-colab-python.ipynb", as_version=4)
nbformat.validate(nb)
for cell in nb.cells:
    if cell.cell_type == "code" and not cell.source.lstrip().startswith(("%",
"!")):
        ast.parse(cell.source)
print("Format dan sintaks dasar valid")
```

Lima Notebook Lengkap Siap Google Colab: Bagian 7

Notebook: 01-dasar-google-colab-python.ipynb

Validasi notebook	Kriteria
Input	Tipe, bentuk, satuan, muatan, spin, dan jalur berkas dinyatakan eksplisit.
Lingkungan	Versi Python dan pustaka tercatat; backend CPU/GPU terdeteksi.

Proses	Status eksekusi atau konvergensi diperiksa sebelum hasil dipakai.
Numerik	Tidak ada NaN/inf; ukuran array dan jumlah atom konsisten.
Keluaran	Nilai, teks, tabel, atau grafik disimpan dan dapat dibuka kembali.
Interpretasi	Klaim tidak melampaui metode, basis set, atau data yang dihitung.
Status pengujian: Laporkan sel yang benar-benar dieksekusi, perangkat yang dipakai, waktu, dan setiap fallback.	

Lima Notebook Lengkap Siap Google Colab: Bagian 8

Notebook: 01-dasar-google-colab-python.ipynb

Sel ekspor dan arsip

```
from pathlib import Path
import json, shutil

Path("hasil").mkdir(exist_ok=True)
Path("hasil/README.txt").write_text("Cara menjalankan dan status pengujian",
encoding="utf-8")
Path("hasil/metadata.json").write_text(json.dumps({"notebook": "01-dasar-
google-colab-python.ipynb"}, indent=2), encoding="utf-8")
shutil.make_archive("hasil_notebook", "zip", root_dir="hasil")
```

Lima Notebook Lengkap Siap Google Colab: Bagian 9

Notebook: 01-dasar-google-colab-python.ipynb

Kesalahan yang mungkin muncul	Respons awal
ModuleNotFoundError	Paket belum dipasang pada runtime aktif. Jalankan sel %pip dan periksa ejaan modul.
ImportError setelah upgrade	Restart session agar proses Python memuat pustaka yang baru.

Konflik NumPy	Catat versi, restart, dan gunakan kombinasi paket yang kompatibel; hindari upgrade acak.
nvidia-smi tidak ditemukan	Runtime bukan GPU atau GPU tidak tersedia. Gunakan CPU dan laporkan fallback.
CUDA package mismatch	Hapus paket GPU yang salah dan pasang varian sesuai CUDA runtime.
CuPy tidak dapat memuat library	Periksa versi CUDA, driver, dan paket CuPy/GPU4PySCF.
SCF not converged	Naikkan max_cycle, periksa geometri, gunakan tebakan/solver Newton, dan jangan menafsirkan energi gagal.
Basis set tidak tersedia	Periksa ejaan dan dukungan unsur pada basis set.

Lima Notebook Lengkap Siap Google Colab: Bagian 10

Notebook: 01-dasar-google-colab-python.ipynb

742. Buat notebook baru dengan nama yang ditentukan.
743. Salin sel sesuai urutan lampiran.
744. Restart session dan Run all.
745. Simpan output penting dan arsip hasil.
746. Tulis laporan singkat tentang versi, waktu, status, dan keterbatasan.

Kriteria notebook lengkap: File dapat dibuka di Colab, lulus nbformat.validate, sel kode dapat dikompilasi sesuai konteks Colab, dan keluaran utama dapat diekspor.

Lima Notebook Lengkap Siap Google Colab: Bagian 11

Notebook: 02-notebook-ilmiah-reproduktif.ipynb

Notebook ini dirancang sebagai dokumen .ipynb yang sah, terdiri atas sel Markdown dan sel kode. Struktur halaman lampiran merepresentasikan urutan sel yang harus dijalankan dari atas ke bawah.

Komponen	Kriteria
Bagian	Judul dan tujuan
Instalasi	Paket dan versi
Konfigurasi	Parameter yang dapat diubah
Perhitungan	Fungsi dan proses utama
Validasi	Format, sintaks, kimia, numerik
Ekspor	TXT, CSV, JSON, XYZ, ZIP

Lima Notebook Lengkap Siap Google Colab: Bagian 12

Notebook: 02-notebook-ilmiah-reproduktif.ipynb

Sel inti: Judul, identitas, tujuan, dan teori singkat. Kode berikut merupakan pola utama yang digunakan dalam notebook.

```
# Judul Notebook

**Penulis:** KASMUI
**Tujuan:** Menjalankan perhitungan yang dapat direproduksi.
**Metode:** B3LYP/def2-TZVP
**Input utama:** muatan, spin, dan koordinat molekul.

> Jalankan notebook dari atas ke bawah setelah Restart session.
```

Validasi sel: Cetak status, periksa nilai finite, dan jangan meneruskan hasil bila syarat minimum gagal.

Lima Notebook Lengkap Siap Google Colab: Bagian 13

Notebook: 02-notebook-ilmiah-reproduktif.ipynb

Sel inti: Konfigurasi parameter dan fungsi modular. Kode berikut merupakan pola utama yang digunakan dalam notebook.

```
from dataclasses import dataclass

@dataclass(frozen=True)
class Config:
```

```
method: str = "b3lyp"
basis: str = "def2-tzvp"
charge: int = 0
spin: int = 0
max_cycle: int = 200

CFG = Config()
print(CFG)
```

Validasi sel: Cetak status, periksa nilai finite, dan jangan meneruskan hasil bila syarat minimum gagal.

Lima Notebook Lengkap Siap Google Colab: Bagian 14

Notebook: 02-notebook-ilmiah-reproduktif.ipynb

Sel inti: Validasi format dan sintaks. Kode berikut merupakan pola utama yang digunakan dalam notebook.

```
import ast, nbformat

nb = nbformat.v4.new_notebook()
nb.cells = [
    nbformat.v4.new_markdown_cell("# Notebook tervalidasi"),
    nbformat.v4.new_code_cell("x = 2\nprint(x**2)"),
]
nbformat.validate(nb)
for cell in nb.cells:
    if cell.cell_type == "code":
        ast.parse(cell.source)
nbformat.write(nb, "notebook_valid.ipynb")
```

Validasi sel: Cetak status, periksa nilai finite, dan jangan meneruskan hasil bila syarat minimum gagal.

Lima Notebook Lengkap Siap Google Colab: Bagian 15

Notebook: 02-notebook-ilmiah-reproduktif.ipynb

Sel pemeriksaan lingkungan

```
import sys, platform, json, time
from importlib.metadata import version, PackageNotFoundError

def package_version(name):
    try: return version(name)
    except PackageNotFoundError: return None
```



```
env = {"python":sys.version, "platform":platform.platform(),
      "timestamp":time.strftime("%Y-%m-%d %H:%M:%S"),
      "packages":{p:package_version(p) for p in
["numpy","rdkit","pyscf","nbformat"]}}
print(json.dumps(env, indent=2))
```

Lima Notebook Lengkap Siap Google Colab: Bagian 16

Notebook: 02-notebook-ilmiah-reproduktif.ipynb

Sel validasi notebook

```
import ast, nbformat
nb = nbformat.read("02-notebook-ilmiah-reproduktif.ipynb", as_version=4)
nbformat.validate(nb)
for cell in nb.cells:
    if cell.cell_type == "code" and not cell.source.lstrip().startswith(("%",
"!")):
        ast.parse(cell.source)
print("Format dan sintaks dasar valid")
```

Lima Notebook Lengkap Siap Google Colab: Bagian 17

Notebook: 02-notebook-ilmiah-reproduktif.ipynb

Validasi notebook	Kriteria
Input	Tipe, bentuk, satuan, muatan, spin, dan jalur berkas dinyatakan eksplisit.
Lingkungan	Versi Python dan pustaka tercatat; backend CPU/GPU terdeteksi.
Proses	Status eksekusi atau konvergensi diperiksa sebelum hasil dipakai.
Numerik	Tidak ada NaN/inf; ukuran array dan jumlah atom konsisten.
Keluaran	Laporan validasi dan jejak versi disimpan dan dapat dibuka kembali.
Interpretasi	Klaim tidak melampaui metode, basis set, atau data yang dihitung.

Status pengujian: Laporkan sel yang benar-benar dieksekusi, perangkat yang dipakai, waktu, dan setiap fallback.

Lima Notebook Lengkap Siap Google Colab: Bagian 18

Notebook: 02-notebook-ilmiah-reproduktif.ipynb

Sel ekspor dan arsip

```
from pathlib import Path
import json, shutil

Path("hasil").mkdir(exist_ok=True)
Path("hasil/README.txt").write_text("Cara menjalankan dan status pengujian",
encoding="utf-8")
Path("hasil/metadata.json").write_text(json.dumps({"notebook": "02-notebook-
ilmiah-reproduktif.ipynb"}, indent=2), encoding="utf-8")
shutil.make_archive("hasil_notebook", "zip", root_dir="hasil")
```

Lima Notebook Lengkap Siap Google Colab: Bagian 19

Notebook: 02-notebook-ilmiah-reproduktif.ipynb

Kesalahan yang mungkin muncul	Respons awal
ModuleNotFoundError	Paket belum dipasang pada runtime aktif. Jalankan sel %pip dan periksa ejaan modul.
ImportError setelah upgrade	Restart session agar proses Python memuat pustaka yang baru.
Konflik NumPy	Catat versi, restart, dan gunakan kombinasi paket yang kompatibel; hindari upgrade acak.
nvidia-smi tidak ditemukan	Runtime bukan GPU atau GPU tidak tersedia. Gunakan CPU dan laporkan fallback.
CUDA package mismatch	Hapus paket GPU yang salah dan pasang varian sesuai CUDA runtime.

CuPy tidak dapat memuat library	Periksa versi CUDA, driver, dan paket CuPy/GPU4PySCF.
SCF not converged	Naikkan max_cycle, periksa geometri, gunakan tebakan/solver Newton, dan jangan menafsirkan energi gagal.
Basis set tidak tersedia	Periksa ejaan dan dukungan unsur pada basis set.

Lima Notebook Lengkap Siap Google Colab: Bagian 20

Notebook: 02-notebook-ilmiah-reproduktif.ipynb

747. Buat notebook baru dengan nama yang ditentukan.
748. Salin sel sesuai urutan lampiran.
749. Restart session dan Run all.
750. Simpan output penting dan arsip hasil.
751. Tulis laporan singkat tentang versi, waktu, status, dan keterbatasan.

Kriteria notebook lengkap: File dapat dibuka di Colab, lulus nbformat.validate, sel kode dapat dikompilasi sesuai konteks Colab, dan keluaran utama dapat diekspor.

Lima Notebook Lengkap Siap Google Colab: Bagian 21

Notebook: 03-rdkit-visualisasi-molekul-3d.ipynb

Notebook ini dirancang sebagai dokumen .ipynb yang sah, terdiri atas sel Markdown dan sel kode. Struktur halaman lampiran merepresentasikan urutan sel yang harus dijalankan dari atas ke bawah.

Komponen	Kriteria
Bagian	Judul dan tujuan
Instalasi	Paket dan versi
Konfigurasi	Parameter yang dapat diubah
Perhitungan	Fungsi dan proses utama

Validasi	Format, sintaks, kimia, numerik
Ekspor	TXT, CSV, JSON, XYZ, ZIP

Lima Notebook Lengkap Siap Google Colab: Bagian 22

Notebook: 03-rdkit-visualisasi-molekul-3d.ipynb

Sel inti: Mengubah SMILES menjadi molekul tiga dimensi. Kode berikut merupakan pola utama yang digunakan dalam notebook.

```
from rdkit import Chem
from rdkit.Chem import AllChem

mol = Chem.AddHs(Chem.MolFromSmiles("CCO"))
params = AllChem.ETKDGv3()
params.randomSeed = 2026
status = AllChem.EmbedMolecule(mol, params)
if status != 0:
    raise RuntimeError("Embedding 3D gagal")
print("Konformer 3D tersedia:", mol.GetNumConformers())
```

Validasi sel: Cetak status, periksa nilai finite, dan jangan meneruskan hasil bila syarat minimum gagal.

Lima Notebook Lengkap Siap Google Colab: Bagian 23

Notebook: 03-rdkit-visualisasi-molekul-3d.ipynb

Sel inti: ETKDGv3, MMFF94, dan UFF. Kode berikut merupakan pola utama yang digunakan dalam notebook.

```
from rdkit import Chem
from rdkit.Chem import AllChem

mol = Chem.AddHs(Chem.MolFromSmiles("CC(=O)NC1=CC=C(O)C=C1O"))
params = AllChem.ETKDGv3(); params.randomSeed = 2026
if AllChem.EmbedMolecule(mol, params) != 0:
    raise RuntimeError("ETKDGv3 gagal")
if AllChem.MMFFHasAllMoleculeParams(mol):
    status = AllChem.MMFFOptimizeMolecule(mol, maxIters=500)
else:
    status = AllChem.UFFOptimizeMolecule(mol, maxIters=500)
print("Status optimasi force field:", status)
```

Validasi sel: Cetak status, periksa nilai finite, dan jangan meneruskan hasil bila syarat minimum gagal.

Lima Notebook Lengkap Siap Google Colab: Bagian 24

Notebook: 03-rdkit-visualisasi-molekul-3d.ipynb

Sel inti: Menampilkan struktur awal. Kode berikut merupakan pola utama yang digunakan dalam notebook.

```
import py3Dmol

xyz_text = open("molekul_awal.xyz", encoding="utf-8").read()
view = py3Dmol.view(width=700, height=450)
view.addModel(xyz_text, "xyz")
view.setStyle({"stick": {}, "sphere": {"scale": 0.30}})
view.zoomTo()
view.show()
```

Validasi sel: Cetak status, periksa nilai finite, dan jangan meneruskan hasil bila syarat minimum gagal.

Lima Notebook Lengkap Siap Google Colab: Bagian 25

Notebook: 03-rdkit-visualisasi-molekul-3d.ipynb

Sel pemeriksaan lingkungan

```
import sys, platform, json, time
from importlib.metadata import version, PackageNotFoundError

def package_version(name):
    try: return version(name)
    except PackageNotFoundError: return None

env = {"python":sys.version, "platform":platform.platform(),
       "timestamp":time.strftime("%Y-%m-%d %H:%M:%S"),
       "packages":{p:package_version(p) for p in
["numpy","rdkit","pyscf","nbformat"]}}
print(json.dumps(env, indent=2))
```

Lima Notebook Lengkap Siap Google Colab: Bagian 26

Notebook: 03-rdkit-visualisasi-molekul-3d.ipynb

Sel validasi notebook

```
import ast, nbformat
nb = nbformat.read("03-rdkit-visualisasi-molekul-3d.ipynb", as_version=4)
nbformat.validate(nb)
for cell in nb.cells:
    if cell.cell_type == "code" and not cell.source.lstrip().startswith(("%",
"!")):
        ast.parse(cell.source)
print("Format dan sintaks dasar valid")
```

Lima Notebook Lengkap Siap Google Colab: Bagian 27

Notebook: 03-rdkit-visualisasi-molekul-3d.ipynb

Validasi notebook	Kriteria
Input	Tipe, bentuk, satuan, muatan, spin, dan jalur berkas dinyatakan eksplisit.
Lingkungan	Versi Python dan pustaka tercatat; backend CPU/GPU terdeteksi.
Proses	Status eksekusi atau konvergensi diperiksa sebelum hasil dipakai.
Numerik	Tidak ada NaN/inf; ukuran array dan jumlah atom konsisten.
Keluaran	Konformer tiga dimensi dan koordinat xyz disimpan dan dapat dibuka kembali.
Interpretasi	Klaim tidak melampaui metode, basis set, atau data yang dihitung.

Status pengujian: Laporkan sel yang benar-benar dieksekusi, perangkat yang dipakai, waktu, dan setiap fallback.

Lima Notebook Lengkap Siap Google Colab: Bagian 28

Notebook: 03-rdkit-visualisasi-molekul-3d.ipynb

Sel ekspor dan arsip

```
from pathlib import Path
import json, shutil
```

```
Path("hasil").mkdir(exist_ok=True)
Path("hasil/README.txt").write_text("Cara menjalankan dan status pengujian",
encoding="utf-8")
Path("hasil/metadata.json").write_text(json.dumps({"notebook": "03-rdkit-
visualisasi-molekul-3d.ipynb"}, indent=2), encoding="utf-8")
shutil.make_archive("hasil_notebook", "zip", root_dir="hasil")
```

Lima Notebook Lengkap Siap Google Colab: Bagian 29

Notebook: 03-rdkit-visualisasi-molekul-3d.ipynb

Kesalahan yang mungkin muncul	Respons awal
ModuleNotFoundError	Paket belum dipasang pada runtime aktif. Jalankan sel %pip dan periksa ejaan modul.
ImportError setelah upgrade	Restart session agar proses Python memuat pustaka yang baru.
Konflik NumPy	Catat versi, restart, dan gunakan kombinasi paket yang kompatibel; hindari upgrade acak.
nvidia-smi tidak ditemukan	Runtime bukan GPU atau GPU tidak tersedia. Gunakan CPU dan laporkan fallback.
CUDA package mismatch	Hapus paket GPU yang salah dan pasang varian sesuai CUDA runtime.
CuPy tidak dapat memuat library	Periksa versi CUDA, driver, dan paket CuPy/GPU4PySCF.
SCF not converged	Naikkan max_cycle, periksa geometri, gunakan tebakan/solver Newton, dan jangan menafsirkan energi gagal.
Basis set tidak tersedia	Periksa ejaan dan dukungan unsur pada basis set.

Lima Notebook Lengkap Siap Google Colab: Bagian 30

Notebook: 03-rdkit-visualisasi-molekul-3d.ipynb

752. Buat notebook baru dengan nama yang ditentukan.
753. Salin sel sesuai urutan lampiran.
754. Restart session dan Run all.
755. Simpan output penting dan arsip hasil.
756. Tulis laporan singkat tentang versi, waktu, status, dan keterbatasan.

Kriteria notebook lengkap: File dapat dibuka di Colab, lulus nbformat.validate, sel kode dapat dikompilasi sesuai konteks Colab, dan keluaran utama dapat diekspor.

Lima Notebook Lengkap Siap Google Colab: Bagian 31

Notebook: 04-optimasi-h2o-b3lyp-def2-tzvp.ipynb

Notebook ini dirancang sebagai dokumen .ipynb yang sah, terdiri atas sel Markdown dan sel kode. Struktur halaman lampiran merepresentasikan urutan sel yang harus dijalankan dari atas ke bawah.

Komponen	Kriteria
Bagian	Judul dan tujuan
Instalasi	Paket dan versi
Konfigurasi	Parameter yang dapat diubah
Perhitungan	Fungsi dan proses utama
Validasi	Format, sintaks, kimia, numerik
Ekspor	TXT, CSV, JSON, XYZ, ZIP

Lima Notebook Lengkap Siap Google Colab: Bagian 32

Notebook: 04-optimasi-h2o-b3lyp-def2-tzvp.ipynb

Sel inti: Instalasi, struktur awal, dan energi awal. Kode berikut merupakan pola utama yang digunakan dalam notebook.

```
%pip -q install --prefer-binary pyscf geometric py3Dmol

from pyscf import gto, dft

mol_h2o = gto.M(
    atom="O 0 0 0; H 0 0.757 0.587; H 0 -0.757 0.587",
```



```
unit="Angstrom", basis="def2-tzvp", charge=0, spin=0
)
mf0 = dft.RKS(mol_h2o); mf0.xc = "b3lyp"
e0 = mf0.kernel()
print("Energi awal:", e0, "konvergen:", mf0.converged)
```

Validasi sel: Cetak status, periksa nilai finite, dan jangan meneruskan hasil bila syarat minimum gagal.

Lima Notebook Lengkap Siap Google Colab: Bagian 33

Notebook: 04-optimasi-h2o-b3lyp-def2-tzvp.ipynb

Sel inti: Optimasi B3LYP/def2-TZVP. Kode berikut merupakan pola utama yang digunakan dalam notebook.

```
from pyscf import dft
from pyscf.geomopt.geometric_solver import optimize

mf_opt = dft.RKS(mol_h2o).density_fit()
mf_opt.xc = "b3lyp"; mf_opt.grids.level = 4
mf_opt.conv_tol = 1e-9; mf_opt.max_cycle = 200
mol_opt = optimize(mf_opt, maxsteps=100)

mf_final = dft.RKS(mol_opt).density_fit()
mf_final.xc = "b3lyp"
e_final = mf_final.kernel()
print("Energi optimum:", e_final)
```

Validasi sel: Cetak status, periksa nilai finite, dan jangan meneruskan hasil bila syarat minimum gagal.

Lima Notebook Lengkap Siap Google Colab: Bagian 34

Notebook: 04-optimasi-h2o-b3lyp-def2-tzvp.ipynb

Sel inti: HOMO, LUMO, celah energi, dan momen dipol. Kode berikut merupakan pola utama yang digunakan dalam notebook.

```
import numpy as np

occ = np.asarray(mf_final.mo_occ)
eps = np.asarray(mf_final.mo_energy)
homo = eps[np.where(occ > 0)[0][-1]]
lumo = eps[np.where(occ == 0)[0][0]]
print("HOMO (eV):", homo * 27.211386245988)
print("LUMO (eV):", lumo * 27.211386245988)
```

```
print("Gap (eV):", (lumo-homo) * 27.211386245988)
print("Dipol (Debye):", mf_final.dip_moment(unit="Debye"))
```

Validasi sel: Cetak status, periksa nilai finite, dan jangan meneruskan hasil bila syarat minimum gagal.

Lima Notebook Lengkap Siap Google Colab: Bagian 35

Notebook: 04-optimasi-h2o-b3lyp-def2-tzvp.ipynb

Sel pemeriksaan lingkungan

```
import sys, platform, json, time
from importlib.metadata import version, PackageNotFoundError

def package_version(name):
    try: return version(name)
    except PackageNotFoundError: return None

env = {"python":sys.version, "platform":platform.platform(),
       "timestamp":time.strftime("%Y-%m-%d %H:%M:%S"),
       "packages":{p:package_version(p) for p in
["numpy","rdkit","pyscf","nbformat"]}}
print(json.dumps(env, indent=2))
```

Lima Notebook Lengkap Siap Google Colab: Bagian 36

Notebook: 04-optimasi-h2o-b3lyp-def2-tzvp.ipynb

Sel validasi notebook

```
import ast, nbformat
nb = nbformat.read("04-optimasi-h2o-b3lyp-def2-tzvp.ipynb", as_version=4)
nbformat.validate(nb)
for cell in nb.cells:
    if cell.cell_type == "code" and not cell.source.lstrip().startswith(("%",
"!")):
        ast.parse(cell.source)
print("Format dan sintaks dasar valid")
```

Lima Notebook Lengkap Siap Google Colab: Bagian 37

Notebook: 04-optimasi-h2o-b3lyp-def2-tzvp.ipynb

Validasi notebook

Kriteria

Input	Tipe, bentuk, satuan, muatan, spin, dan jalur berkas dinyatakan eksplisit.
Lingkungan	Versi Python dan pustaka tercatat; backend CPU/GPU terdeteksi.
Proses	Status eksekusi atau konvergensi diperiksa sebelum hasil dipakai.
Numerik	Tidak ada NaN/inf; ukuran array dan jumlah atom konsisten.
Keluaran	Energi, geometri, orbital, dan sifat molekul disimpan dan dapat dibuka kembali.
Interpretasi	Klaim tidak melampaui metode, basis set, atau data yang dihitung.

Status pengujian: Laporkan sel yang benar-benar dieksekusi, perangkat yang dipakai, waktu, dan setiap fallback.

Lima Notebook Lengkap Siap Google Colab: Bagian 38

Notebook: 04-optimasi-h2o-b3lyp-def2-tzvp.ipynb

Sel ekspor dan arsip

```
from pathlib import Path
import json, shutil

Path("hasil").mkdir(exist_ok=True)
Path("hasil/README.txt").write_text("Cara menjalankan dan status pengujian",
encoding="utf-8")
Path("hasil/metadata.json").write_text(json.dumps({"notebook": "04-optimasi-
h2o-b3lyp-def2-tzvp.ipynb"}, indent=2), encoding="utf-8")
shutil.make_archive("hasil_notebook", "zip", root_dir="hasil")
```

Lima Notebook Lengkap Siap Google Colab: Bagian 39

Notebook: 04-optimasi-h2o-b3lyp-def2-tzvp.ipynb

Kesalahan yang mungkin muncul

Respons awal

ModuleNotFoundError	Paket belum dipasang pada runtime aktif. Jalankan sel %pip dan periksa ejaan modul.
ImportError setelah upgrade	Restart session agar proses Python memuat pustaka yang baru.
Konflik NumPy	Catat versi, restart, dan gunakan kombinasi paket yang kompatibel; hindari upgrade acak.
nvidia-smi tidak ditemukan	Runtime bukan GPU atau GPU tidak tersedia. Gunakan CPU dan laporkan fallback.
CUDA package mismatch	Hapus paket GPU yang salah dan pasang varian sesuai CUDA runtime.
CuPy tidak dapat memuat library	Periksa versi CUDA, driver, dan paket CuPy/GPU4PySCF.
SCF not converged	Naikkan max_cycle, periksa geometri, gunakan tebakan/solver Newton, dan jangan menafsirkan energi gagal.
Basis set tidak tersedia	Periksa ejaan dan dukungan unsur pada basis set.

Lima Notebook Lengkap Siap Google Colab: Bagian 40

Notebook: 04-optimasi-h2o-b3lyp-def2-tzvp.ipynb

757. Buat notebook baru dengan nama yang ditentukan.
758. Salin sel sesuai urutan lampiran.
759. Restart session dan Run all.
760. Simpan output penting dan arsip hasil.
761. Tulis laporan singkat tentang versi, waktu, status, dan keterbatasan.

Kriteria notebook lengkap: File dapat dibuka di Colab, lulus nbformat.validate, sel kode dapat dikompilasi sesuai konteks Colab, dan keluaran utama dapat diekspor.

Lima Notebook Lengkap Siap Google Colab: Bagian 41

Notebook: 05-optimasi-paracetamol-colab.ipynb

Notebook ini dirancang sebagai dokumen .ipynb yang sah, terdiri atas sel Markdown dan sel kode. Struktur halaman lampiran merepresentasikan urutan sel yang harus dijalankan dari atas ke bawah.

Komponen	Kriteria
Bagian	Judul dan tujuan
Instalasi	Paket dan versi
Konfigurasi	Parameter yang dapat diubah
Perhitungan	Fungsi dan proses utama
Validasi	Format, sintaks, kimia, numerik
Ekspor	TXT, CSV, JSON, XYZ, ZIP

Lima Notebook Lengkap Siap Google Colab: Bagian 42

Notebook: 05-optimasi-paracetamol-colab.ipynb

Sel inti: Pembuatan struktur awal parasetamol. Kode berikut merupakan pola utama yang digunakan dalam notebook.

```
from rdkit import Chem
from rdkit.Chem import AllChem

SMILES = "CC(=O)NC1=CC=C(O)C=C1"
mol = Chem.AddHs(Chem.MolFromSmiles(SMILES))
params = AllChem.ETKDGv3(); params.randomSeed = 2026
if AllChem.EmbedMolecule(mol, params) != 0:
    raise RuntimeError("Konformer parasetamol gagal dibuat")
if AllChem.MMFFHasAllMoleculeParams(mol):
    AllChem.MMFFOptimizeMolecule(mol, maxIters=1000)
else:
    AllChem.UFFOptimizeMolecule(mol, maxIters=1000)
Chem.MolToXYZFile(mol, "paracetamol_awal.xyz")
```

Validasi sel: Cetak status, periksa nilai finite, dan jangan meneruskan hasil bila syarat minimum gagal.

Lima Notebook Lengkap Siap Google Colab: Bagian 43

Notebook: 05-optimasi-paracetamol-colab.ipynb

Sel inti: Optimasi DFT B3LYP/def2-TZVP. Kode berikut merupakan pola utama yang digunakan dalam notebook.

```
from pyscf import gto, dft
from pyscf.geomopt.geometric_solver import optimize

atom_text = "\n".join(open("paracetamol_awal.xyz").read().splitlines()[2:])
mol_qm = gto.M(atom=atom_text, unit="Angstrom", basis="def2-tzvp", charge=0,
spin=0)
mf = dft.RKS(mol_qm).density_fit()
mf.xc = "b3lyp"; mf.grids.level = 4
mf.conv_tol = 1e-8; mf.max_cycle = 200
e0 = mf.kernel()
if not mf.converged: mf = mf.newton(); e0 = mf.kernel()
mol_opt = optimize(mf, maxsteps=200)
```

Validasi sel: Cetak status, periksa nilai finite, dan jangan meneruskan hasil bila syarat minimum gagal.

Lima Notebook Lengkap Siap Google Colab: Bagian 44

Notebook: 05-optimasi-paracetamol-colab.ipynb

Sel inti: Analisis, visualisasi, dan ekspor seluruh hasil. Kode berikut merupakan pola utama yang digunakan dalam notebook.

```
import json, shutil
from pathlib import Path

result = {"molecule": "paracetamol", "method": "B3LYP",
          "basis": "def2-TZVP", "energy_hartree": float(e_final),
          "scf_converged": bool(mf_final.converged)}
Path("hasil.json").write_text(json.dumps(result, indent=2), encoding="utf-8")
Path("hasil.txt").write_text("\n".join(f"{k}: {v}" for k,v in result.items()),
encoding="utf-8")
shutil.make_archive("paracetamol_results", "zip", root_dir=".",
                    base_dir=None)
```

Validasi sel: Cetak status, periksa nilai finite, dan jangan meneruskan hasil bila syarat minimum gagal.

Lima Notebook Lengkap Siap Google Colab: Bagian 45

Notebook: 05-optimasi-paracetamol-colab.ipynb

Sel pemeriksaan lingkungan

```
import sys, platform, json, time
from importlib.metadata import version, PackageNotFoundError

def package_version(name):
    try: return version(name)
    except PackageNotFoundError: return None

env = {"python":sys.version, "platform":platform.platform(),
       "timestamp":time.strftime("%Y-%m-%d %H:%M:%S"),
       "packages":{p:package_version(p) for p in
["numpy","rdkit","pyscf","nbformat"]}}
print(json.dumps(env, indent=2))
```

Lima Notebook Lengkap Siap Google Colab: Bagian 46

Notebook: 05-optimasi-paracetamol-colab.ipynb

Sel validasi notebook

```
import ast, nbformat
nb = nbformat.read("05-optimasi-paracetamol-colab.ipynb", as_version=4)
nbformat.validate(nb)
for cell in nb.cells:
    if cell.cell_type == "code" and not cell.source.lstrip().startswith(("%",
"!")):
        ast.parse(cell.source)
print("Format dan sintaks dasar valid")
```

Lima Notebook Lengkap Siap Google Colab: Bagian 47

Notebook: 05-optimasi-paracetamol-colab.ipynb

Validasi notebook	Kriteria
Input	Tipe, bentuk, satuan, muatan, spin, dan jalur berkas dinyatakan eksplisit.

Lingkungan	Versi Python dan pustaka tercatat; backend CPU/GPU terdeteksi.
Proses	Status eksekusi atau konvergensi diperiksa sebelum hasil dipakai.
Numerik	Tidak ada NaN/inf; ukuran array dan jumlah atom konsisten.
Keluaran	Hasil yang dapat diperiksa dan disimpan disimpan dan dapat dibuka kembali.
Interpretasi	Klaim tidak melampaui metode, basis set, atau data yang dihitung.

Status pengujian: Laporkan sel yang benar-benar dieksekusi, perangkat yang dipakai, waktu, dan setiap fallback.

Lima Notebook Lengkap Siap Google Colab: Bagian 48

Notebook: 05-optimasi-paracetamol-colab.ipynb

Sel ekspor dan arsip

```
from pathlib import Path
import json, shutil

Path("hasil").mkdir(exist_ok=True)
Path("hasil/README.txt").write_text("Cara menjalankan dan status pengujian",
encoding="utf-8")
Path("hasil/metadata.json").write_text(json.dumps({"notebook": "05-optimasi-
paracetamol-colab.ipynb"}, indent=2), encoding="utf-8")
shutil.make_archive("hasil_notebook", "zip", root_dir="hasil")
```

Lima Notebook Lengkap Siap Google Colab: Bagian 49

Notebook: 05-optimasi-paracetamol-colab.ipynb

Kesalahan yang mungkin muncul	Respons awal
ModuleNotFoundError	Paket belum dipasang pada runtime aktif. Jalankan sel %pip dan periksa ejaan modul.

ImportError setelah upgrade	Restart session agar proses Python memuat pustaka yang baru.
Konflik NumPy	Catat versi, restart, dan gunakan kombinasi paket yang kompatibel; hindari upgrade acak.
nvidia-smi tidak ditemukan	Runtime bukan GPU atau GPU tidak tersedia. Gunakan CPU dan laporkan fallback.
CUDA package mismatch	Hapus paket GPU yang salah dan pasang varian sesuai CUDA runtime.
CuPy tidak dapat memuat library	Periksa versi CUDA, driver, dan paket CuPy/GPU4PySCF.
SCF not converged	Naikkan max_cycle, periksa geometri, gunakan tebakan/solver Newton, dan jangan menafsirkan energi gagal.
Basis set tidak tersedia	Periksa ejaan dan dukungan unsur pada basis set.

Lima Notebook Lengkap Siap Google Colab: Bagian 50

Notebook: 05-optimasi-paracetamol-colab.ipynb

762. Buat notebook baru dengan nama yang ditentukan.
763. Salin sel sesuai urutan lampiran.
764. Restart session dan Run all.
765. Simpan output penting dan arsip hasil.
766. Tulis laporan singkat tentang versi, waktu, status, dan keterbatasan.

Kriteria notebook lengkap: File dapat dibuka di Colab, lulus nbformat.validate, sel kode dapat dikompilasi sesuai konteks Colab, dan keluaran utama dapat diekspor.

LAMPIRAN E - 60 RESEP NOTEBOOK PYTHON SIAP GOOGLE COLAB

Lampiran ini menyediakan enam puluh resep notebook yang dapat digunakan sebagai unit latihan mandiri. Setiap resep memisahkan tujuan, input, proses, validasi, keluaran, dan pengembangan lebih lanjut. Kode ditulis agar mudah dipindahkan ke Google Colab, tetapi pembaca tetap harus menjalankannya dari runtime bersih, memeriksa versi pustaka, serta menyesuaikan jalur berkas dan parameter dengan proyek yang sedang dikerjakan.

Resep disusun dari keterampilan Python dasar menuju pengolahan data, visualisasi, manajemen berkas, dan kimia komputasi. Sebagian kode menghasilkan berkas pada direktori /content. Karena penyimpanan runtime bersifat sementara, hasil penting harus diunduh atau dipindahkan ke Google Drive sebelum sesi berakhir. Untuk perhitungan kimia, muatan, spin, koordinat, metode, basis set, dan status konvergensi harus selalu dicatat bersama hasil.

E.1 Python Dasar dan Lingkungan Runtime

E.01 Mendeteksi Versi Python dan Sistem Runtime

Resep ini membahas pemeriksaan versi Python, platform, jumlah inti CPU, direktori kerja, dan variabel lingkungan. Tujuannya bukan sekadar memperoleh keluaran yang tampak benar, tetapi membangun alur notebook yang dapat dijalankan ulang dan diperiksa. Pembaca perlu menyatakan input secara eksplisit, memisahkan konfigurasi dari proses utama, mencetak status penting, dan menyimpan artefak yang diperlukan. Hasil utama yang ditargetkan adalah ringkasan lingkungan yang dapat disalin ke laporan pengujian. Setiap angka harus disertai satuan dan konteks, sedangkan setiap berkas perlu diberi nama yang menjelaskan molekul, metode, atau tahap proses.

Tujuan, Input, dan Keluaran

Input resep meliputi parameter yang relevan dengan pemeriksaan versi Python, platform, jumlah inti CPU, direktori kerja, dan variabel lingkungan. Sebelum eksekusi, periksa tipe data, bentuk array, jalur berkas, dan ketersediaan pustaka. Keluaran harus dibedakan menjadi keluaran langsung program, hasil validasi, dan interpretasi. Keluaran langsung dapat berupa nilai, tabel, gambar, atau berkas; hasil validasi menyatakan apakah syarat minimum terpenuhi; interpretasi menjelaskan makna hasil tanpa melampaui metode yang digunakan.

Kode Siap Google Colab

```
import os, platform, sys
from pathlib import Path

info = {
    "python": sys.version.split()[0],
```

```
"platform": platform.platform(),  
"processor": platform.processor() or "tidak terdeteksi",  
"cpu_count": os.cpu_count(),  
"working_directory": str(Path.cwd()),  
}  
for k, v in info.items():  
    print(f"{k:20s}: {v}")
```

Cara Membaca dan Memvalidasi Hasil

Setelah sel dijalankan, baca pesan dari atas ke bawah dan pastikan tidak ada pengecualian yang terlewat. Verifikasi bahwa keluaran sesuai dengan ringkasan lingkungan yang dapat disalin ke laporan pengujian. Untuk keluaran numerik, periksa bilangan hingga, satuan, tanda, skala, dan batas kewajaran. Untuk berkas, periksa keberadaan, ukuran, encoding, dan kemampuan untuk dibuka kembali. Untuk grafik atau visualisasi, pastikan label sumbu, satuan, legenda, dan judul menjelaskan konteks data.

Risiko utama pada resep ini adalah Mengandalkan nama perangkat saja tidak cukup karena image runtime dan versi Python dapat berubah tanpa mengubah nama notebook. Oleh karena itu, status keberhasilan tidak boleh disimpulkan hanya dari munculnya output. Jalankan kembali notebook dari sesi bersih, bandingkan hasil dengan perhitungan sederhana atau data acuan ketika tersedia, dan catat versi lingkungan. Bila hasil berubah, identifikasi apakah perubahan berasal dari input, urutan eksekusi, versi pustaka, perangkat, atau algoritma.

Pengembangan dan Latihan

767. Tambahkan informasi RAM, waktu eksekusi, dan nama akun proyek tanpa menyimpan data pribadi yang tidak diperlukan.
768. Tambahkan satu pemeriksaan yang sengaja menghentikan proses ketika input tidak memenuhi syarat, lalu uji pesan kesalahannya.
769. Simpan parameter, status, waktu eksekusi, dan hasil utama ke JSON atau CSV; kemudian baca kembali berkas tersebut dalam runtime baru.
770. Tuliskan dua paragraf interpretasi: paragraf pertama hanya memuat fakta keluaran program, sedangkan paragraf kedua memuat makna dan keterbatasannya.

E.02 Mengelola Variabel dan Satuan Secara Eksplisit

Resep ini membahas penamaan variabel, konversi satuan, konstanta, dan format keluaran numerik. Tujuannya bukan sekadar memperoleh keluaran yang tampak benar, tetapi membangun alur notebook yang dapat dijalankan ulang dan diperiksa. Pembaca perlu menyatakan input secara eksplisit, memisahkan konfigurasi dari proses utama, mencetak status penting, dan menyimpan artefak yang diperlukan. Hasil utama yang ditargetkan adalah nilai panjang dalam beberapa satuan dengan jejak konversi yang jelas. Setiap angka harus disertai satuan dan konteks, sedangkan setiap berkas perlu diberi nama yang menjelaskan molekul, metode, atau tahap proses.

Tujuan, Input, dan Keluaran

Input resep meliputi parameter yang relevan dengan penamaan variabel, konversi satuan, konstanta, dan format keluaran numerik. Sebelum eksekusi, periksa tipe data, bentuk array, jalur berkas, dan ketersediaan pustaka. Keluaran harus dibedakan menjadi keluaran langsung program, hasil validasi, dan interpretasi. Keluaran langsung dapat berupa nilai, tabel, gambar, atau berkas; hasil validasi menyatakan apakah syarat minimum terpenuhi; interpretasi menjelaskan makna hasil tanpa melampaui metode yang digunakan.

Kode Siap Google Colab

```
ANGSTROM_KE_NM = 0.1
HARTREE_KE_EV = 27.211386245988

panjang_angstrom = 0.9572
energi_hartree = -76.3681

panjang_nm = panjang_angstrom * ANGSTROM_KE_NM
energi_ev = energi_hartree * HARTREE_KE_EV

print(f"Panjang : {panjang_angstrom:.4f} Å = {panjang_nm:.5f} nm")
print(f"Energi : {energi_hartree:.6f} Eh = {energi_ev:.6f} eV")
```

Cara Membaca dan Memvalidasi Hasil

Setelah sel dijalankan, baca pesan dari atas ke bawah dan pastikan tidak ada pengecualian yang terlewat. Verifikasi bahwa keluaran sesuai dengan nilai panjang dalam beberapa satuan dengan jejak konversi yang jelas. Untuk keluaran numerik, periksa bilangan hingga, satuan, tanda, skala, dan batas kewajaran. Untuk berkas, periksa keberadaan, ukuran, encoding, dan kemampuan untuk dibuka kembali. Untuk grafik atau visualisasi, pastikan label sumbu, satuan, legenda, dan judul menjelaskan konteks data.

Risiko utama pada resep ini adalah Konstanta tanpa nama dan angka yang tidak disertai satuan membuat notebook sulit diaudit. Oleh karena itu, status keberhasilan tidak boleh disimpulkan hanya dari munculnya output. Jalankan kembali notebook dari sesi bersih, bandingkan hasil dengan perhitungan sederhana atau data acuan ketika tersedia, dan catat versi lingkungan. Bila hasil berubah, identifikasi apakah perubahan berasal dari input, urutan eksekusi, versi pustaka, perangkat, atau algoritma.

Pengembangan dan Latihan

771. Tambahkan konversi energi ke kJ mol⁻¹ dan sertakan pemeriksaan bahwa input merupakan bilangan hingga.
772. Tambahkan satu pemeriksaan yang sengaja menghentikan proses ketika input tidak memenuhi syarat, lalu uji pesan kesalahannya.
773. Simpan parameter, status, waktu eksekusi, dan hasil utama ke JSON atau CSV; kemudian baca kembali berkas tersebut dalam runtime baru.

774. Tuliskan dua paragraf interpretasi: paragraf pertama hanya memuat fakta keluaran program, sedangkan paragraf kedua memuat makna dan keterbatasannya.

E.03 Menyusun List, Tuple, dan Dictionary untuk Parameter

Resep ini membahas struktur data Python untuk menyimpan daftar metode, koordinat, dan konfigurasi perhitungan. Tujuannya bukan sekadar memperoleh keluaran yang tampak benar, tetapi membangun alur notebook yang dapat dijalankan ulang dan diperiksa. Pembaca perlu menyatakan input secara eksplisit, memisahkan konfigurasi dari proses utama, mencetak status penting, dan menyimpan artefak yang diperlukan. Hasil utama yang ditargetkan adalah objek konfigurasi yang dapat dibaca manusia dan dipakai ulang. Setiap angka harus disertai satuan dan konteks, sedangkan setiap berkas perlu diberi nama yang menjelaskan molekul, metode, atau tahap proses.

Tujuan, Input, dan Keluaran

Input resep meliputi parameter yang relevan dengan struktur data Python untuk menyimpan daftar metode, koordinat, dan konfigurasi perhitungan. Sebelum eksekusi, periksa tipe data, bentuk array, jalur berkas, dan ketersediaan pustaka. Keluaran harus dibedakan menjadi keluaran langsung program, hasil validasi, dan interpretasi. Keluaran langsung dapat berupa nilai, tabel, gambar, atau berkas; hasil validasi menyatakan apakah syarat minimum terpenuhi; interpretasi menjelaskan makna hasil tanpa melampaui metode yang digunakan.

Kode Siap Google Colab

```
metode = ["RHF", "B3LYP", "PBE0"]
basis_set = ("sto-3g", "6-31g", "def2-svp")
konfigurasi = {
    "molekul": "H2O",
    "charge": 0,
    "spin": 0,
    "metode": metode[1],
    "basis": basis_set[2],
}

for k, v in konfigurasi.items():
    print(f"{k:10s}: {v}")
```

Cara Membaca dan Memvalidasi Hasil

Setelah sel dijalankan, baca pesan dari atas ke bawah dan pastikan tidak ada pengecualian yang terlewat. Verifikasi bahwa keluaran sesuai dengan objek konfigurasi yang dapat dibaca manusia dan dipakai ulang. Untuk keluaran numerik, periksa bilangan hingga, satuan, tanda, skala, dan batas kewajaran. Untuk berkas, periksa keberadaan, ukuran, encoding, dan kemampuan untuk dibuka kembali. Untuk grafik atau visualisasi, pastikan label sumbu, satuan, legenda, dan judul menjelaskan konteks data.

Risiko utama pada resep ini adalah Mengubah elemen struktur data tanpa memahami apakah objek bersifat mutable dapat menimbulkan state tersembunyi. Oleh karena itu, status keberhasilan tidak boleh disimpulkan hanya dari munculnya output. Jalankan kembali notebook dari sesi bersih, bandingkan hasil dengan perhitungan sederhana atau data acuan ketika tersedia, dan catat versi lingkungan. Bila hasil berubah, identifikasi apakah perubahan berasal dari input, urutan eksekusi, versi pustaka, perangkat, atau algoritma.

Pengembangan dan Latihan

775. Buat dua konfigurasi molekul berbeda dan bandingkan tanpa menimpa konfigurasi awal.
776. Tambahkan satu pemeriksaan yang sengaja menghentikan proses ketika input tidak memenuhi syarat, lalu uji pesan kesalahannya.
777. Simpan parameter, status, waktu eksekusi, dan hasil utama ke JSON atau CSV; kemudian baca kembali berkas tersebut dalam runtime baru.
778. Tuliskan dua paragraf interpretasi: paragraf pertama hanya memuat fakta keluaran program, sedangkan paragraf kedua memuat makna dan keterbatasannya.

E.04 Membuat Percabangan untuk Validasi Input

Resep ini membahas if, elif, else, operator logika, dan validasi rentang. Tujuannya bukan sekadar memperoleh keluaran yang tampak benar, tetapi membangun alur notebook yang dapat dijalankan ulang dan diperiksa. Pembaca perlu menyatakan input secara eksplisit, memisahkan konfigurasi dari proses utama, mencetak status penting, dan menyimpan artefak yang diperlukan. Hasil utama yang ditargetkan adalah pesan validasi yang menghentikan input tidak masuk akal. Setiap angka harus disertai satuan dan konteks, sedangkan setiap berkas perlu diberi nama yang menjelaskan molekul, metode, atau tahap proses.

Tujuan, Input, dan Keluaran

Input resep meliputi parameter yang relevan dengan if, elif, else, operator logika, dan validasi rentang. Sebelum eksekusi, periksa tipe data, bentuk array, jalur berkas, dan ketersediaan pustaka. Keluaran harus dibedakan menjadi keluaran langsung program, hasil validasi, dan interpretasi. Keluaran langsung dapat berupa nilai, tabel, gambar, atau berkas; hasil validasi menyatakan apakah syarat minimum terpenuhi; interpretasi menjelaskan makna hasil tanpa melampaui metode yang digunakan.

Kode Siap Google Colab

```
def validasi_molekul(charge, spin, jumlah_elektron):
    if not isinstance(charge, int) or not isinstance(spin, int):
        raise TypeError("charge dan spin harus bertipe integer")
    if spin < 0:
        raise ValueError("spin tidak boleh negatif")
    elektron_efektif = jumlah_elektron - charge
```

```
if (elektron_efektif - spin) % 2 != 0:  
    raise ValueError("Kombinasi jumlah elektron dan spin tidak konsisten")  
return True  
  
print(validasi_molekul(charge=0, spin=0, jumlah_elektron=10))
```

Cara Membaca dan Memvalidasi Hasil

Setelah sel dijalankan, baca pesan dari atas ke bawah dan pastikan tidak ada pengecualian yang terlewat. Verifikasi bahwa keluaran sesuai dengan pesan validasi yang menghentikan input tidak masuk akal. Untuk keluaran numerik, periksa bilangan hingga, satuan, tanda, skala, dan batas kewajaran. Untuk berkas, periksa keberadaan, ukuran, encoding, dan kemampuan untuk dibuka kembali. Untuk grafik atau visualisasi, pastikan label sumbu, satuan, legenda, dan judul menjelaskan konteks data.

Risiko utama pada resep ini adalah Validasi yang hanya mencetak peringatan tetapi tetap melanjutkan perhitungan dapat menghasilkan keluaran yang tampak sah. Oleh karena itu, status keberhasilan tidak boleh disimpulkan hanya dari munculnya output. Jalankan kembali notebook dari sesi bersih, bandingkan hasil dengan perhitungan sederhana atau data acuan ketika tersedia, dan catat versi lingkungan. Bila hasil berubah, identifikasi apakah perubahan berasal dari input, urutan eksekusi, versi pustaka, perangkat, atau algoritma.

Pengembangan dan Latihan

779. Uji fungsi dengan tiga input sengaja salah dan catat pesan kesalahannya.
780. Tambahkan satu pemeriksaan yang sengaja menghentikan proses ketika input tidak memenuhi syarat, lalu uji pesan kesalahannya.
781. Simpan parameter, status, waktu eksekusi, dan hasil utama ke JSON atau CSV; kemudian baca kembali berkas tersebut dalam runtime baru.
782. Tuliskan dua paragraf interpretasi: paragraf pertama hanya memuat fakta keluaran program, sedangkan paragraf kedua memuat makna dan keterbatasannya.

E.05 Menggunakan Perulangan dengan Status Proses

Resep ini membahas for, enumerate, akumulasi hasil, dan pelaporan kemajuan. Tujuannya bukan sekadar memperoleh keluaran yang tampak benar, tetapi membangun alur notebook yang dapat dijalankan ulang dan diperiksa. Pembaca perlu menyatakan input secara eksplisit, memisahkan konfigurasi dari proses utama, mencetak status penting, dan menyimpan artefak yang diperlukan. Hasil utama yang ditargetkan adalah daftar hasil terstruktur beserta indeks dan status. Setiap angka harus disertai satuan dan konteks, sedangkan setiap berkas perlu diberi nama yang menjelaskan molekul, metode, atau tahap proses.

Tujuan, Input, dan Keluaran

Input resep meliputi parameter yang relevan dengan for, enumerate, akumulasi hasil, dan pelaporan kemajuan. Sebelum eksekusi, periksa tipe data, bentuk array, jalur berkas, dan ketersediaan pustaka. Keluaran harus dibedakan menjadi keluaran langsung program, hasil validasi, dan interpretasi. Keluaran langsung dapat berupa nilai, tabel, gambar, atau berkas; hasil validasi menyatakan apakah syarat minimum terpenuhi; interpretasi menjelaskan makna hasil tanpa melampaui metode yang digunakan.

Kode Siap Google Colab

```
jarak = [0.60, 0.70, 0.74, 0.80, 1.00]
hasil = []
for i, r in enumerate(jarak, start=1):
    nilai = (r - 0.74) ** 2
    hasil.append({"titik": i, "jarak": r, "nilai": nilai})
    print(f"Titik {i:02d}/{len(jarak)} | r={r:.2f} | nilai={nilai:.6f}")
```

Cara Membaca dan Memvalidasi Hasil

Setelah sel dijalankan, baca pesan dari atas ke bawah dan pastikan tidak ada pengecualian yang terlewat. Verifikasi bahwa keluaran sesuai dengan daftar hasil terstruktur beserta indeks dan status. Untuk keluaran numerik, periksa bilangan hingga, satuan, tanda, skala, dan batas kewajaran. Untuk berkas, periksa keberadaan, ukuran, encoding, dan kemampuan untuk dibuka kembali. Untuk grafik atau visualisasi, pastikan label sumbu, satuan, legenda, dan judul menjelaskan konteks data.

Risiko utama pada resep ini adalah Perulangan yang menimpa variabel hasil pada setiap iterasi menyebabkan data sebelumnya hilang. Oleh karena itu, status keberhasilan tidak boleh disimpulkan hanya dari munculnya output. Jalankan kembali notebook dari sesi bersih, bandingkan hasil dengan perhitungan sederhana atau data acuan ketika tersedia, dan catat versi lingkungan. Bila hasil berubah, identifikasi apakah perubahan berasal dari input, urutan eksekusi, versi pustaka, perangkat, atau algoritma.

Pengembangan dan Latihan

783. Tambahkan penentuan titik dengan nilai minimum dan simpan seluruh hasil ke CSV.
784. Tambahkan satu pemeriksaan yang sengaja menghentikan proses ketika input tidak memenuhi syarat, lalu uji pesan kesalahannya.
785. Simpan parameter, status, waktu eksekusi, dan hasil utama ke JSON atau CSV; kemudian baca kembali berkas tersebut dalam runtime baru.
786. Tuliskan dua paragraf interpretasi: paragraf pertama hanya memuat fakta keluaran program, sedangkan paragraf kedua memuat makna dan keterbatasannya.

E.06 Membuat Fungsi dengan Type Hint dan Docstring

Resep ini membahas fungsi modular, parameter default, type hint, docstring, dan nilai balik. Tujuannya bukan sekadar memperoleh keluaran yang tampak benar, tetapi membangun alur notebook yang dapat dijalankan ulang dan diperiksa. Pembaca perlu menyatakan input secara eksplisit, memisahkan konfigurasi dari proses utama, mencetak status penting, dan menyimpan artefak yang diperlukan. Hasil utama yang ditargetkan adalah fungsi konversi yang terdokumentasi dan mudah diuji. Setiap angka harus disertai satuan dan konteks, sedangkan setiap berkas perlu diberi nama yang menjelaskan molekul, metode, atau tahap proses.

Tujuan, Input, dan Keluaran

Input resep meliputi parameter yang relevan dengan fungsi modular, parameter default, type hint, docstring, dan nilai balik. Sebelum eksekusi, periksa tipe data, bentuk array, jalur berkas, dan ketersediaan pustaka. Keluaran harus dibedakan menjadi keluaran langsung program, hasil validasi, dan interpretasi. Keluaran langsung dapat berupa nilai, tabel, gambar, atau berkas; hasil validasi menyatakan apakah syarat minimum terpenuhi; interpretasi menjelaskan makna hasil tanpa melampaui metode yang digunakan.

Kode Siap Google Colab

```
def hartree_ke_ev(energi: float, faktor: float = 27.211386245988) -> float:
    """Mengubah energi dari Hartree menjadi elektronvolt."""
    if not isinstance(energi, (int, float)):
        raise TypeError("energi harus numerik")
    return float(energi) * faktor

nilai = hartree_ke_ev(-1.117349)
print(f"Energi = {nilai:.8f} eV")
```

Cara Membaca dan Memvalidasi Hasil

Setelah sel dijalankan, baca pesan dari atas ke bawah dan pastikan tidak ada pengecualian yang terlewat. Verifikasi bahwa keluaran sesuai dengan fungsi konversi yang terdokumentasi dan mudah diuji. Untuk keluaran numerik, periksa bilangan hingga, satuan, tanda, skala, dan batas kewajaran. Untuk berkas, periksa keberadaan, ukuran, encoding, dan kemampuan untuk dibuka kembali. Untuk grafik atau visualisasi, pastikan label sumbu, satuan, legenda, dan judul menjelaskan konteks data.

Risiko utama pada resep ini adalah Fungsi yang bergantung pada variabel global sulit diuji dan mudah berubah akibat eksekusi sel lain. Oleh karena itu, status keberhasilan tidak boleh disimpulkan hanya dari munculnya output. Jalankan kembali notebook dari sesi bersih, bandingkan hasil dengan perhitungan sederhana atau data acuan ketika tersedia, dan catat versi lingkungan. Bila hasil berubah, identifikasi apakah perubahan berasal dari input, urutan eksekusi, versi pustaka, perangkat, atau algoritma.

Pengembangan dan Latihan

787. Buat unit test sederhana untuk nilai nol, nilai negatif, dan input bertipe string.
788. Tambahkan satu pemeriksaan yang sengaja menghentikan proses ketika input tidak memenuhi syarat, lalu uji pesan kesalahannya.
789. Simpan parameter, status, waktu eksekusi, dan hasil utama ke JSON atau CSV; kemudian baca kembali berkas tersebut dalam runtime baru.
790. Tuliskan dua paragraf interpretasi: paragraf pertama hanya memuat fakta keluaran program, sedangkan paragraf kedua memuat makna dan keterbatasannya.

E.07 Menangani Kesalahan dengan try-except yang Informatif

Resep ini membahas penangkapan pengecualian spesifik, finally, dan pesan diagnosis. Tujuannya bukan sekadar memperoleh keluaran yang tampak benar, tetapi membangun alur notebook yang dapat dijalankan ulang dan diperiksa. Pembaca perlu menyatakan input secara eksplisit, memisahkan konfigurasi dari proses utama, mencetak status penting, dan menyimpan artefak yang diperlukan. Hasil utama yang ditargetkan adalah alur yang gagal dengan aman tanpa menyembunyikan penyebab. Setiap angka harus disertai satuan dan konteks, sedangkan setiap berkas perlu diberi nama yang menjelaskan molekul, metode, atau tahap proses.

Tujuan, Input, dan Keluaran

Input resep meliputi parameter yang relevan dengan penangkapan pengecualian spesifik, finally, dan pesan diagnosis. Sebelum eksekusi, periksa tipe data, bentuk array, jalur berkas, dan ketersediaan pustaka. Keluaran harus dibedakan menjadi keluaran langsung program, hasil validasi, dan interpretasi. Keluaran langsung dapat berupa nilai, tabel, gambar, atau berkas; hasil validasi menyatakan apakah syarat minimum terpenuhi; interpretasi menjelaskan makna hasil tanpa melampaui metode yang digunakan.

Kode Siap Google Colab

```
from pathlib import Path

nama = Path("data_yang_belum_ada.csv")
try:
    teks = nama.read_text(encoding="utf-8")
except FileNotFoundError as exc:
    print("Berkas tidak ditemukan:", exc.filename)
except UnicodeDecodeError:
    print("Encoding berkas tidak sesuai UTF-8")
else:
    print("Jumlah karakter:", len(teks))
finally:
    print("Pemeriksaan berkas selesai")
```

Cara Membaca dan Memvalidasi Hasil

Setelah sel dijalankan, baca pesan dari atas ke bawah dan pastikan tidak ada pengecualian yang terlewat. Verifikasi bahwa keluaran sesuai dengan alur yang gagal dengan aman tanpa menyembunyikan penyebab. Untuk keluaran numerik, periksa bilangan hingga, satuan, tanda, skala, dan batas kewajaran. Untuk berkas, periksa keberadaan, ukuran, encoding, dan kemampuan untuk dibuka kembali. Untuk grafik atau visualisasi, pastikan label sumbu, satuan, legenda, dan judul menjelaskan konteks data.

Risiko utama pada resep ini adalah Menggunakan except tanpa menyebut jenis kesalahan dapat menutupi bug pemrograman yang sebenarnya. Oleh karena itu, status keberhasilan tidak boleh disimpulkan hanya dari munculnya output. Jalankan kembali notebook dari sesi bersih, bandingkan hasil dengan perhitungan sederhana atau data acuan ketika tersedia, dan catat versi lingkungan. Bila hasil berubah, identifikasi apakah perubahan berasal dari input, urutan eksekusi, versi pustaka, perangkat, atau algoritma.

Pengembangan dan Latihan

791. Tambahkan penanganan `PermissionError` dan simpan catatan kesalahan ke berkas log.
792. Tambahkan satu pemeriksaan yang sengaja menghentikan proses ketika input tidak memenuhi syarat, lalu uji pesan kesalahannya.
793. Simpan parameter, status, waktu eksekusi, dan hasil utama ke JSON atau CSV; kemudian baca kembali berkas tersebut dalam runtime baru.
794. Tuliskan dua paragraf interpretasi: paragraf pertama hanya memuat fakta keluaran program, sedangkan paragraf kedua memuat makna dan keterbatasannya.

E.08 Membaca dan Menulis Teks UTF-8

Resep ini membahas Pathlib, encoding, pemisahan baris, dan pemeriksaan keberadaan berkas. Tujuannya bukan sekadar memperoleh keluaran yang tampak benar, tetapi membangun alur notebook yang dapat dijalankan ulang dan diperiksa. Pembaca perlu menyatakan input secara eksplisit, memisahkan konfigurasi dari proses utama, mencetak status penting, dan menyimpan artefak yang diperlukan. Hasil utama yang ditargetkan adalah berkas teks yang dapat dibuka kembali tanpa karakter rusak. Setiap angka harus disertai satuan dan konteks, sedangkan setiap berkas perlu diberi nama yang menjelaskan molekul, metode, atau tahap proses.

Tujuan, Input, dan Keluaran

Input resep meliputi parameter yang relevan dengan Pathlib, encoding, pemisahan baris, dan pemeriksaan keberadaan berkas. Sebelum eksekusi, periksa tipe data, bentuk array, jalur berkas, dan ketersediaan pustaka. Keluaran harus dibedakan menjadi keluaran langsung program, hasil validasi, dan interpretasi. Keluaran langsung dapat berupa nilai, tabel, gambar, atau berkas; hasil validasi menyatakan apakah syarat minimum terpenuhi; interpretasi menjelaskan makna hasil tanpa melampaui metode yang digunakan.

Kode Siap Google Colab

```
from pathlib import Path

path = Path("catatan_runtime.txt")
baris = [
    "Proyek: Tutorial Google Colab",
    "Satuan panjang: Å",
    "Status: berhasil",
]
path.write_text("\n".join(baris) + "\n", encoding="utf-8")

assert path.exists()
print(path.read_text(encoding="utf-8"))
```

Cara Membaca dan Memvalidasi Hasil

Setelah sel dijalankan, baca pesan dari atas ke bawah dan pastikan tidak ada pengecualian yang terlewat. Verifikasi bahwa keluaran sesuai dengan berkas teks yang dapat dibuka kembali tanpa karakter rusak. Untuk keluaran numerik, periksa bilangan hingga, satuan, tanda, skala, dan batas kewajaran. Untuk berkas, periksa keberadaan, ukuran, encoding, dan kemampuan untuk dibuka kembali. Untuk grafik atau visualisasi, pastikan label sumbu, satuan, legenda, dan judul menjelaskan konteks data.

Risiko utama pada resep ini adalah Mengandalkan encoding bawaan sistem dapat menghasilkan karakter rusak ketika berkas dipindahkan. Oleh karena itu, status keberhasilan tidak boleh disimpulkan hanya dari munculnya output. Jalankan kembali notebook dari sesi bersih, bandingkan hasil dengan perhitungan sederhana atau data acuan ketika tersedia, dan catat versi lingkungan. Bila hasil berubah, identifikasi apakah perubahan berasal dari input, urutan eksekusi, versi pustaka, perangkat, atau algoritma.

Pengembangan dan Latihan

795. Tambahkan tanggal, versi Python, dan checksum sederhana pada berkas yang sama.
796. Tambahkan satu pemeriksaan yang sengaja menghentikan proses ketika input tidak memenuhi syarat, lalu uji pesan kesalahannya.
797. Simpan parameter, status, waktu eksekusi, dan hasil utama ke JSON atau CSV; kemudian baca kembali berkas tersebut dalam runtime baru.
798. Tuliskan dua paragraf interpretasi: paragraf pertama hanya memuat fakta keluaran program, sedangkan paragraf kedua memuat makna dan keterbatasannya.

E.09 Menyimpan Konfigurasi sebagai JSON

Resep ini membahas serialisasi dictionary, indentasi, dan pembacaan ulang. Tujuannya bukan sekadar memperoleh keluaran yang tampak benar, tetapi membangun alur notebook yang dapat dijalankan ulang dan diperiksa. Pembaca perlu menyatakan input secara eksplisit, memisahkan konfigurasi dari proses utama, mencetak status penting, dan

menyimpan artefak yang diperlukan. Hasil utama yang ditargetkan adalah berkas konfigurasi yang mudah diperiksa dan direproduksi. Setiap angka harus disertai satuan dan konteks, sedangkan setiap berkas perlu diberi nama yang menjelaskan molekul, metode, atau tahap proses.

Tujuan, Input, dan Keluaran

Input resep meliputi parameter yang relevan dengan serialisasi dictionary, indentasi, dan pembacaan ulang. Sebelum eksekusi, periksa tipe data, bentuk array, jalur berkas, dan ketersediaan pustaka. Keluaran harus dibedakan menjadi keluaran langsung program, hasil validasi, dan interpretasi. Keluaran langsung dapat berupa nilai, tabel, gambar, atau berkas; hasil validasi menyatakan apakah syarat minimum terpenuhi; interpretasi menjelaskan makna hasil tanpa melampaui metode yang digunakan.

Kode Siap Google Colab

```
import json
from pathlib import Path

config = {
    "molecule": "H2O",
    "method": "B3LYP",
    "basis": "def2-SVP",
    "charge": 0,
    "spin": 0,
}
path = Path("config_h2o.json")
path.write_text(json.dumps(config, indent=2), encoding="utf-8")

loaded = json.loads(path.read_text(encoding="utf-8"))
assert loaded == config
print(loaded)
```

Cara Membaca dan Memvalidasi Hasil

Setelah sel dijalankan, baca pesan dari atas ke bawah dan pastikan tidak ada pengecualian yang terlewat. Verifikasi bahwa keluaran sesuai dengan berkas konfigurasi yang mudah diperiksa dan direproduksi. Untuk keluaran numerik, periksa bilangan hingga, satuan, tanda, skala, dan batas kewajaran. Untuk berkas, periksa keberadaan, ukuran, encoding, dan kemampuan untuk dibuka kembali. Untuk grafik atau visualisasi, pastikan label sumbu, satuan, legenda, dan judul menjelaskan konteks data.

Risiko utama pada resep ini adalah Objek NumPy dan tipe khusus lain tidak selalu dapat diserialisasi langsung ke JSON. Oleh karena itu, status keberhasilan tidak boleh disimpulkan hanya dari munculnya output. Jalankan kembali notebook dari sesi bersih, bandingkan hasil dengan perhitungan sederhana atau data acuan ketika tersedia, dan catat versi lingkungan. Bila hasil berubah, identifikasi apakah perubahan berasal dari input, urutan eksekusi, versi pustaka, perangkat, atau algoritma.

Pengembangan dan Latihan

799. Tambahkan parameter grid, toleransi SCF, dan waktu pembuatan konfigurasi.
800. Tambahkan satu pemeriksaan yang sengaja menghentikan proses ketika input tidak memenuhi syarat, lalu uji pesan kesalahannya.
801. Simpan parameter, status, waktu eksekusi, dan hasil utama ke JSON atau CSV; kemudian baca kembali berkas tersebut dalam runtime baru.
802. Tuliskan dua paragraf interpretasi: paragraf pertama hanya memuat fakta keluaran program, sedangkan paragraf kedua memuat makna dan keterbatasannya.

E.10 Mengelola Jalur Berkas dengan pathlib

Resep ini membahas jalur relatif, direktori proyek, pembuatan folder, dan enumerasi berkas. Tujuannya bukan sekadar memperoleh keluaran yang tampak benar, tetapi membangun alur notebook yang dapat dijalankan ulang dan diperiksa. Pembaca perlu menyatakan input secara eksplisit, memisahkan konfigurasi dari proses utama, mencetak status penting, dan menyimpan artefak yang diperlukan. Hasil utama yang ditargetkan adalah struktur folder proyek yang portabel. Setiap angka harus disertai satuan dan konteks, sedangkan setiap berkas perlu diberi nama yang menjelaskan molekul, metode, atau tahap proses.

Tujuan, Input, dan Keluaran

Input resep meliputi parameter yang relevan dengan jalur relatif, direktori proyek, pembuatan folder, dan enumerasi berkas. Sebelum eksekusi, periksa tipe data, bentuk array, jalur berkas, dan ketersediaan pustaka. Keluaran harus dibedakan menjadi keluaran langsung program, hasil validasi, dan interpretasi. Keluaran langsung dapat berupa nilai, tabel, gambar, atau berkas; hasil validasi menyatakan apakah syarat minimum terpenuhi; interpretasi menjelaskan makna hasil tanpa melampaui metode yang digunakan.

Kode Siap Google Colab

```
from pathlib import Path

root = Path("proyek_colab")
folder = {
    "input": root / "input",
    "output": root / "output",
    "figures": root / "figures",
    "logs": root / "logs",
}
for path in folder.values():
    path.mkdir(parents=True, exist_ok=True)

for nama, path in folder.items():
    print(f"{nama:8s}: {path.resolve()}")
```

Cara Membaca dan Memvalidasi Hasil

Setelah sel dijalankan, baca pesan dari atas ke bawah dan pastikan tidak ada pengecualian yang terlewat. Verifikasi bahwa keluaran sesuai dengan struktur folder proyek yang portabel. Untuk keluaran numerik, periksa bilangan hingga, satuan, tanda, skala, dan batas kewajaran. Untuk berkas, periksa keberadaan, ukuran, encoding, dan kemampuan untuk dibuka kembali. Untuk grafik atau visualisasi, pastikan label sumbu, satuan, legenda, dan judul menjelaskan konteks data.

Risiko utama pada resep ini adalah Menggabungkan jalur menggunakan string manual rentan terhadap kesalahan separator dan nama folder. Oleh karena itu, status keberhasilan tidak boleh disimpulkan hanya dari munculnya output. Jalankan kembali notebook dari sesi bersih, bandingkan hasil dengan perhitungan sederhana atau data acuan ketika tersedia, dan catat versi lingkungan. Bila hasil berubah, identifikasi apakah perubahan berasal dari input, urutan eksekusi, versi pustaka, perangkat, atau algoritma.

Pengembangan dan Latihan

803. Buat fungsi yang memastikan seluruh folder ada sebelum notebook menjalankan perhitungan.
804. Tambahkan satu pemeriksaan yang sengaja menghentikan proses ketika input tidak memenuhi syarat, lalu uji pesan kesalahannya.
805. Simpan parameter, status, waktu eksekusi, dan hasil utama ke JSON atau CSV; kemudian baca kembali berkas tersebut dalam runtime baru.
806. Tuliskan dua paragraf interpretasi: paragraf pertama hanya memuat fakta keluaran program, sedangkan paragraf kedua memuat makna dan keterbatasannya.

E.11 Membuat Log Eksekusi Sederhana

Resep ini membahas logging, level pesan, format waktu, dan berkas log. Tujuannya bukan sekadar memperoleh keluaran yang tampak benar, tetapi membangun alur notebook yang dapat dijalankan ulang dan diperiksa. Pembaca perlu menyatakan input secara eksplisit, memisahkan konfigurasi dari proses utama, mencetak status penting, dan menyimpan artefak yang diperlukan. Hasil utama yang ditargetkan adalah catatan proses yang tetap tersedia setelah sel selesai. Setiap angka harus disertai satuan dan konteks, sedangkan setiap berkas perlu diberi nama yang menjelaskan molekul, metode, atau tahap proses.

Tujuan, Input, dan Keluaran

Input resep meliputi parameter yang relevan dengan logging, level pesan, format waktu, dan berkas log. Sebelum eksekusi, periksa tipe data, bentuk array, jalur berkas, dan ketersediaan pustaka. Keluaran harus dibedakan menjadi keluaran langsung program, hasil validasi, dan interpretasi. Keluaran langsung dapat berupa nilai, tabel, gambar, atau berkas; hasil validasi menyatakan apakah syarat minimum terpenuhi; interpretasi menjelaskan makna hasil tanpa melampaui metode yang digunakan.

Kode Siap Google Colab

```
import logging

logging.basicConfig(
    filename="notebook.log",
    level=logging.INFO,
    format="%(asctime)s | %(levelname)s | %(message)s",
    force=True,
)
logging.info("Notebook dimulai")
logging.info("Parameter berhasil divalidasi")
logging.warning("Contoh peringatan: basis set minimal")
print("Log tersimpan pada notebook.log")
```

Cara Membaca dan Memvalidasi Hasil

Setelah sel dijalankan, baca pesan dari atas ke bawah dan pastikan tidak ada pengecualian yang terlewat. Verifikasi bahwa keluaran sesuai dengan catatan proses yang tetap tersedia setelah sel selesai. Untuk keluaran numerik, periksa bilangan hingga, satuan, tanda, skala, dan batas kewajaran. Untuk berkas, periksa keberadaan, ukuran, encoding, dan kemampuan untuk dibuka kembali. Untuk grafik atau visualisasi, pastikan label sumbu, satuan, legenda, dan judul menjelaskan konteks data.

Risiko utama pada resep ini adalah Mencetak semua informasi ke layar membuat keluaran panjang dan menyulitkan penelusuran kejadian. Oleh karena itu, status keberhasilan tidak boleh disimpulkan hanya dari munculnya output. Jalankan kembali notebook dari sesi bersih, bandingkan hasil dengan perhitungan sederhana atau data acuan ketika tersedia, dan catat versi lingkungan. Bila hasil berubah, identifikasi apakah perubahan berasal dari input, urutan eksekusi, versi pustaka, perangkat, atau algoritma.

Pengembangan dan Latihan

807. Tambahkan handler layar untuk pesan WARNING ke atas dan buka lima baris terakhir log.
808. Tambahkan satu pemeriksaan yang sengaja menghentikan proses ketika input tidak memenuhi syarat, lalu uji pesan kesalahannya.
809. Simpan parameter, status, waktu eksekusi, dan hasil utama ke JSON atau CSV; kemudian baca kembali berkas tersebut dalam runtime baru.
810. Tuliskan dua paragraf interpretasi: paragraf pertama hanya memuat fakta keluaran program, sedangkan paragraf kedua memuat makna dan keterbatasannya.

E.12 Mengukur Waktu Eksekusi Secara Konsisten

Resep ini membahas `perf_counter`, fungsi pembungkus, dan perbandingan waktu. Tujuannya bukan sekadar memperoleh keluaran yang tampak benar, tetapi membangun alur notebook yang dapat dijalankan ulang dan diperiksa. Pembaca perlu menyatakan input secara eksplisit, memisahkan konfigurasi dari proses utama, mencetak status

penting, dan menyimpan artefak yang diperlukan. Hasil utama yang ditargetkan adalah durasi proses dengan satuan dan konteks yang jelas. Setiap angka harus disertai satuan dan konteks, sedangkan setiap berkas perlu diberi nama yang menjelaskan molekul, metode, atau tahap proses.

Tujuan, Input, dan Keluaran

Input resep meliputi parameter yang relevan dengan `perf_counter`, fungsi pembungkus, dan perbandingan waktu. Sebelum eksekusi, periksa tipe data, bentuk array, jalur berkas, dan ketersediaan pustaka. Keluaran harus dibedakan menjadi keluaran langsung program, hasil validasi, dan interpretasi. Keluaran langsung dapat berupa nilai, tabel, gambar, atau berkas; hasil validasi menyatakan apakah syarat minimum terpenuhi; interpretasi menjelaskan makna hasil tanpa melampaui metode yang digunakan.

Kode Siap Google Colab

```
from time import perf_counter

def ukur_waktu(fungsi, *args, **kwargs):
    mulai = perf_counter()
    hasil = fungsi(*args, **kwargs)
    durasi = perf_counter() - mulai
    return hasil, durasi

hasil, detik = ukur_waktu(sum, range(1_000_000))
print("Hasil :", hasil)
print(f"Durasi : {detik:.6f} s")
```

Cara Membaca dan Memvalidasi Hasil

Setelah sel dijalankan, baca pesan dari atas ke bawah dan pastikan tidak ada pengecualian yang terlewat. Verifikasi bahwa keluaran sesuai dengan durasi proses dengan satuan dan konteks yang jelas. Untuk keluaran numerik, periksa bilangan hingga, satuan, tanda, skala, dan batas kewajaran. Untuk berkas, periksa keberadaan, ukuran, encoding, dan kemampuan untuk dibuka kembali. Untuk grafik atau visualisasi, pastikan label sumbu, satuan, legenda, dan judul menjelaskan konteks data.

Risiko utama pada resep ini adalah Perbandingan waktu hanya bermakna jika input, runtime, dan kondisi perangkat dicatat. Oleh karena itu, status keberhasilan tidak boleh disimpulkan hanya dari munculnya output. Jalankan kembali notebook dari sesi bersih, bandingkan hasil dengan perhitungan sederhana atau data acuan ketika tersedia, dan catat versi lingkungan. Bila hasil berubah, identifikasi apakah perubahan berasal dari input, urutan eksekusi, versi pustaka, perangkat, atau algoritma.

Pengembangan dan Latihan

811. Jalankan fungsi lima kali, hitung median, dan jelaskan variasi yang muncul.

812. Tambahkan satu pemeriksaan yang sengaja menghentikan proses ketika input tidak memenuhi syarat, lalu uji pesan kesalahannya.
813. Simpan parameter, status, waktu eksekusi, dan hasil utama ke JSON atau CSV; kemudian baca kembali berkas tersebut dalam runtime baru.
814. Tuliskan dua paragraf interpretasi: paragraf pertama hanya memuat fakta keluaran program, sedangkan paragraf kedua memuat makna dan keterbatasannya.

E.2 NumPy dan Komputasi Numerik

E.13 Membuat Array dan Memeriksa Shape

Resep ini membahas array, dtype, ndim, shape, dan validasi bilangan hingga. Tujuannya bukan sekadar memperoleh keluaran yang tampak benar, tetapi membangun alur notebook yang dapat dijalankan ulang dan diperiksa. Pembaca perlu menyatakan input secara eksplisit, memisahkan konfigurasi dari proses utama, mencetak status penting, dan menyimpan artefak yang diperlukan. Hasil utama yang ditargetkan adalah array numerik yang siap diproses. Setiap angka harus disertai satuan dan konteks, sedangkan setiap berkas perlu diberi nama yang menjelaskan molekul, metode, atau tahap proses.

Tujuan, Input, dan Keluaran

Input resep meliputi parameter yang relevan dengan array, dtype, ndim, shape, dan validasi bilangan hingga. Sebelum eksekusi, periksa tipe data, bentuk array, jalur berkas, dan ketersediaan pustaka. Keluaran harus dibedakan menjadi keluaran langsung program, hasil validasi, dan interpretasi. Keluaran langsung dapat berupa nilai, tabel, gambar, atau berkas; hasil validasi menyatakan apakah syarat minimum terpenuhi; interpretasi menjelaskan makna hasil tanpa melampaui metode yang digunakan.

Kode Siap Google Colab

```
import numpy as np

koordinat = np.array([
    [0.0000, 0.0000, 0.0000],
    [0.0000, 0.7570, 0.5860],
    [0.0000, -0.7570, 0.5860],
], dtype=float)

assert koordinat.shape == (3, 3)
assert np.isfinite(koordinat).all()
print("shape:", koordinat.shape)
print(koordinat)
```

Cara Membaca dan Memvalidasi Hasil

Setelah sel dijalankan, baca pesan dari atas ke bawah dan pastikan tidak ada pengecualian yang terlewat. Verifikasi bahwa keluaran sesuai dengan array numerik yang siap diproses. Untuk keluaran numerik, periksa bilangan hingga, satuan, tanda, skala, dan batas

kewajaran. Untuk berkas, periksa keberadaan, ukuran, encoding, dan kemampuan untuk dibuka kembali. Untuk grafik atau visualisasi, pastikan label sumbu, satuan, legenda, dan judul menjelaskan konteks data.

Risiko utama pada resep ini adalah Array dengan dtype object dapat menyimpan campuran nilai dan menyebabkan operasi numerik tidak konsisten. Oleh karena itu, status keberhasilan tidak boleh disimpulkan hanya dari munculnya output. Jalankan kembali notebook dari sesi bersih, bandingkan hasil dengan perhitungan sederhana atau data acuan ketika tersedia, dan catat versi lingkungan. Bila hasil berubah, identifikasi apakah perubahan berasal dari input, urutan eksekusi, versi pustaka, perangkat, atau algoritma.

Pengembangan dan Latihan

815. Tambahkan satu atom dan pastikan pemeriksaan shape diperbarui dengan benar.
816. Tambahkan satu pemeriksaan yang sengaja menghentikan proses ketika input tidak memenuhi syarat, lalu uji pesan kesalahannya.
817. Simpan parameter, status, waktu eksekusi, dan hasil utama ke JSON atau CSV; kemudian baca kembali berkas tersebut dalam runtime baru.
818. Tuliskan dua paragraf interpretasi: paragraf pertama hanya memuat fakta keluaran program, sedangkan paragraf kedua memuat makna dan keterbatasannya.

E.14 Menghitung Jarak Antaratom dengan Vektor

Resep ini membahas pengurangan vektor, norma Euclidean, dan indeks atom. Tujuannya bukan sekadar memperoleh keluaran yang tampak benar, tetapi membangun alur notebook yang dapat dijalankan ulang dan diperiksa. Pembaca perlu menyatakan input secara eksplisit, memisahkan konfigurasi dari proses utama, mencetak status penting, dan menyimpan artefak yang diperlukan. Hasil utama yang ditargetkan adalah panjang ikatan dari koordinat Cartesian. Setiap angka harus disertai satuan dan konteks, sedangkan setiap berkas perlu diberi nama yang menjelaskan molekul, metode, atau tahap proses.

Tujuan, Input, dan Keluaran

Input resep meliputi parameter yang relevan dengan pengurangan vektor, norma Euclidean, dan indeks atom. Sebelum eksekusi, periksa tipe data, bentuk array, jalur berkas, dan ketersediaan pustaka. Keluaran harus dibedakan menjadi keluaran langsung program, hasil validasi, dan interpretasi. Keluaran langsung dapat berupa nilai, tabel, gambar, atau berkas; hasil validasi menyatakan apakah syarat minimum terpenuhi; interpretasi menjelaskan makna hasil tanpa melampaui metode yang digunakan.

Kode Siap Google Colab

```
import numpy as np

xyz = np.array([[0,0,0], [0,0,0.9572]], dtype=float)
vektor = xyz[1] - xyz[0]
jarak = np.linalg.norm(vektor)
```

```
print(f"Jarak atom 0-1 = {jarak:.6f} Å")  
assert jarak > 0
```

Cara Membaca dan Memvalidasi Hasil

Setelah sel dijalankan, baca pesan dari atas ke bawah dan pastikan tidak ada pengecualian yang terlewat. Verifikasi bahwa keluaran sesuai dengan panjang ikatan dari koordinat Cartesian. Untuk keluaran numerik, periksa bilangan hingga, satuan, tanda, skala, dan batas kewajaran. Untuk berkas, periksa keberadaan, ukuran, encoding, dan kemampuan untuk dibuka kembali. Untuk grafik atau visualisasi, pastikan label sumbu, satuan, legenda, dan judul menjelaskan konteks data.

Risiko utama pada resep ini adalah Indeks atom yang salah tetap dapat menghasilkan angka sehingga pemeriksaan makna kimia tetap diperlukan. Oleh karena itu, status keberhasilan tidak boleh disimpulkan hanya dari munculnya output. Jalankan kembali notebook dari sesi bersih, bandingkan hasil dengan perhitungan sederhana atau data acuan ketika tersedia, dan catat versi lingkungan. Bila hasil berubah, identifikasi apakah perubahan berasal dari input, urutan eksekusi, versi pustaka, perangkat, atau algoritma.

Pengembangan dan Latihan

819. Kembangkan fungsi jarak(*i*, *j*) dan uji pada tiga pasang atom.
820. Tambahkan satu pemeriksaan yang sengaja menghentikan proses ketika input tidak memenuhi syarat, lalu uji pesan kesalahannya.
821. Simpan parameter, status, waktu eksekusi, dan hasil utama ke JSON atau CSV; kemudian baca kembali berkas tersebut dalam runtime baru.
822. Tuliskan dua paragraf interpretasi: paragraf pertama hanya memuat fakta keluaran program, sedangkan paragraf kedua memuat makna dan keterbatasannya.

E.15 Menghitung Sudut Ikatan

Resep ini membahas dot product, normalisasi vektor, arccos, dan konversi derajat. Tujuannya bukan sekadar memperoleh keluaran yang tampak benar, tetapi membangun alur notebook yang dapat dijalankan ulang dan diperiksa. Pembaca perlu menyatakan input secara eksplisit, memisahkan konfigurasi dari proses utama, mencetak status penting, dan menyimpan artefak yang diperlukan. Hasil utama yang ditargetkan adalah sudut tiga atom yang stabil secara numerik. Setiap angka harus disertai satuan dan konteks, sedangkan setiap berkas perlu diberi nama yang menjelaskan molekul, metode, atau tahap proses.

Tujuan, Input, dan Keluaran

Input resep meliputi parameter yang relevan dengan dot product, normalisasi vektor, arccos, dan konversi derajat. Sebelum eksekusi, periksa tipe data, bentuk array, jalur berkas, dan ketersediaan pustaka. Keluaran harus dibedakan menjadi keluaran langsung program, hasil validasi, dan interpretasi. Keluaran langsung dapat berupa nilai, tabel,

gambar, atau berkas; hasil validasi menyatakan apakah syarat minimum terpenuhi; interpretasi menjelaskan makna hasil tanpa melampaui metode yang digunakan.

Kode Siap Google Colab

```
import numpy as np

def sudut(a, b, c):
    v1 = a - b
    v2 = c - b
    cos_theta = np.dot(v1, v2) / (np.linalg.norm(v1)*np.linalg.norm(v2))
    theta = np.degrees(np.arccos(np.clip(cos_theta, -1.0, 1.0)))
    return theta

O = np.array([0.0, 0.0, 0.0])
H1 = np.array([0.0, 0.757, 0.586])
H2 = np.array([0.0, -0.757, 0.586])
print(f"Sudut H-O-H = {sudut(H1,O,H2):.3f}°")
```

Cara Membaca dan Memvalidasi Hasil

Setelah sel dijalankan, baca pesan dari atas ke bawah dan pastikan tidak ada pengecualian yang terlewat. Verifikasi bahwa keluaran sesuai dengan sudut tiga atom yang stabil secara numerik. Untuk keluaran numerik, periksa bilangan hingga, satuan, tanda, skala, dan batas kewajaran. Untuk berkas, periksa keberadaan, ukuran, encoding, dan kemampuan untuk dibuka kembali. Untuk grafik atau visualisasi, pastikan label sumbu, satuan, legenda, dan judul menjelaskan konteks data.

Risiko utama pada resep ini adalah Tanpa `np.clip`, galat pembulatan dapat membuat nilai `cos` sedikit di luar rentang `arccos`. Oleh karena itu, status keberhasilan tidak boleh disimpulkan hanya dari munculnya output. Jalankan kembali notebook dari sesi bersih, bandingkan hasil dengan perhitungan sederhana atau data acuan ketika tersedia, dan catat versi lingkungan. Bila hasil berubah, identifikasi apakah perubahan berasal dari input, urutan eksekusi, versi pustaka, perangkat, atau algoritma.

Pengembangan dan Latihan

823. Tambahkan pemeriksaan untuk vektor nol dan berikan pesan kesalahan yang jelas.
824. Tambahkan satu pemeriksaan yang sengaja menghentikan proses ketika input tidak memenuhi syarat, lalu uji pesan kesalahannya.
825. Simpan parameter, status, waktu eksekusi, dan hasil utama ke JSON atau CSV; kemudian baca kembali berkas tersebut dalam runtime baru.
826. Tuliskan dua paragraf interpretasi: paragraf pertama hanya memuat fakta keluaran program, sedangkan paragraf kedua memuat makna dan keterbatasannya.

E.16 Menggunakan Masking untuk Menyaring Data

Resep ini membahas boolean mask, isfinite, dan pemilihan data valid. Tujuannya bukan sekadar memperoleh keluaran yang tampak benar, tetapi membangun alur notebook yang dapat dijalankan ulang dan diperiksa. Pembaca perlu menyatakan input secara eksplisit, memisahkan konfigurasi dari proses utama, mencetak status penting, dan menyimpan artefak yang diperlukan. Hasil utama yang ditargetkan adalah subset data tanpa NaN dan inf. Setiap angka harus disertai satuan dan konteks, sedangkan setiap berkas perlu diberi nama yang menjelaskan molekul, metode, atau tahap proses.

Tujuan, Input, dan Keluaran

Input resep meliputi parameter yang relevan dengan boolean mask, isfinite, dan pemilihan data valid. Sebelum eksekusi, periksa tipe data, bentuk array, jalur berkas, dan ketersediaan pustaka. Keluaran harus dibedakan menjadi keluaran langsung program, hasil validasi, dan interpretasi. Keluaran langsung dapat berupa nilai, tabel, gambar, atau berkas; hasil validasi menyatakan apakah syarat minimum terpenuhi; interpretasi menjelaskan makna hasil tanpa melampaui metode yang digunakan.

Kode Siap Google Colab

```
import numpy as np

energi = np.array([-1.0, -1.1, np.nan, -1.08, np.inf])
mask = np.isfinite(energi)
valid = energi[mask]
print("Jumlah valid:", valid.size)
print("Energi valid:", valid)
assert valid.size > 0
```

Cara Membaca dan Memvalidasi Hasil

Setelah sel dijalankan, baca pesan dari atas ke bawah dan pastikan tidak ada pengecualian yang terlewat. Verifikasi bahwa keluaran sesuai dengan subset data tanpa NaN dan inf. Untuk keluaran numerik, periksa bilangan hingga, satuan, tanda, skala, dan batas kewajaran. Untuk berkas, periksa keberadaan, ukuran, encoding, dan kemampuan untuk dibuka kembali. Untuk grafik atau visualisasi, pastikan label sumbu, satuan, legenda, dan judul menjelaskan konteks data.

Risiko utama pada resep ini adalah Menghapus nilai tidak valid tanpa mencatat indeksnya dapat menyembunyikan titik perhitungan yang gagal. Oleh karena itu, status keberhasilan tidak boleh disimpulkan hanya dari munculnya output. Jalankan kembali notebook dari sesi bersih, bandingkan hasil dengan perhitungan sederhana atau data acuan ketika tersedia, dan catat versi lingkungan. Bila hasil berubah, identifikasi apakah perubahan berasal dari input, urutan eksekusi, versi pustaka, perangkat, atau algoritma.

Pengembangan dan Latihan

827. Simpan indeks titik yang gagal dan buat ringkasan alasan kegagalan.
828. Tambahkan satu pemeriksaan yang sengaja menghentikan proses ketika input tidak memenuhi syarat, lalu uji pesan kesalahannya.
829. Simpan parameter, status, waktu eksekusi, dan hasil utama ke JSON atau CSV; kemudian baca kembali berkas tersebut dalam runtime baru.
830. Tuliskan dua paragraf interpretasi: paragraf pertama hanya memuat fakta keluaran program, sedangkan paragraf kedua memuat makna dan keterbatasannya.

E.17 Melakukan Reshape dan Transpose

Resep ini membahas reshape, transpose, flatten, dan konsistensi jumlah elemen. Tujuannya bukan sekadar memperoleh keluaran yang tampak benar, tetapi membangun alur notebook yang dapat dijalankan ulang dan diperiksa. Pembaca perlu menyatakan input secara eksplisit, memisahkan konfigurasi dari proses utama, mencetak status penting, dan menyimpan artefak yang diperlukan. Hasil utama yang ditargetkan adalah matriks dengan orientasi yang sesuai untuk analisis. Setiap angka harus disertai satuan dan konteks, sedangkan setiap berkas perlu diberi nama yang menjelaskan molekul, metode, atau tahap proses.

Tujuan, Input, dan Keluaran

Input resep meliputi parameter yang relevan dengan reshape, transpose, flatten, dan konsistensi jumlah elemen. Sebelum eksekusi, periksa tipe data, bentuk array, jalur berkas, dan ketersediaan pustaka. Keluaran harus dibedakan menjadi keluaran langsung program, hasil validasi, dan interpretasi. Keluaran langsung dapat berupa nilai, tabel, gambar, atau berkas; hasil validasi menyatakan apakah syarat minimum terpenuhi; interpretasi menjelaskan makna hasil tanpa melampaui metode yang digunakan.

Kode Siap Google Colab

```
import numpy as np

data = np.arange(12, dtype=float)
matriks = data.reshape(4, 3)
transpose = matriks.T
print("Matriks:\n", matriks)
print("Transpose:\n", transpose)
assert transpose.shape == (3, 4)
```

Cara Membaca dan Memvalidasi Hasil

Setelah sel dijalankan, baca pesan dari atas ke bawah dan pastikan tidak ada pengecualian yang terlewat. Verifikasi bahwa keluaran sesuai dengan matriks dengan orientasi yang sesuai untuk analisis. Untuk keluaran numerik, periksa bilangan hingga, satuan, tanda, skala, dan batas kewajaran. Untuk berkas, periksa keberadaan, ukuran, encoding, dan

kemampuan untuk dibuka kembali. Untuk grafik atau visualisasi, pastikan label sumbu, satuan, legenda, dan judul menjelaskan konteks data.

Risiko utama pada resep ini adalah Reshape tidak mengubah urutan data secara konseptual; salah membaca sumbu dapat merusak interpretasi. Oleh karena itu, status keberhasilan tidak boleh disimpulkan hanya dari munculnya output. Jalankan kembali notebook dari sesi bersih, bandingkan hasil dengan perhitungan sederhana atau data acuan ketika tersedia, dan catat versi lingkungan. Bila hasil berubah, identifikasi apakah perubahan berasal dari input, urutan eksekusi, versi pustaka, perangkat, atau algoritma.

Pengembangan dan Latihan

831. Ubah menjadi bentuk 2 x 2 x 3 dan jelaskan arti setiap sumbu.
832. Tambahkan satu pemeriksaan yang sengaja menghentikan proses ketika input tidak memenuhi syarat, lalu uji pesan kesalahannya.
833. Simpan parameter, status, waktu eksekusi, dan hasil utama ke JSON atau CSV; kemudian baca kembali berkas tersebut dalam runtime baru.
834. Tuliskan dua paragraf interpretasi: paragraf pertama hanya memuat fakta keluaran program, sedangkan paragraf kedua memuat makna dan keterbatasannya.

E.18 Memahami Broadcasting

Resep ini membahas operasi array dengan skalar dan vektor tanpa loop eksplisit. Tujuannya bukan sekadar memperoleh keluaran yang tampak benar, tetapi membangun alur notebook yang dapat dijalankan ulang dan diperiksa. Pembaca perlu menyatakan input secara eksplisit, memisahkan konfigurasi dari proses utama, mencetak status penting, dan menyimpan artefak yang diperlukan. Hasil utama yang ditargetkan adalah koordinat yang ditranslasi terhadap pusat tertentu. Setiap angka harus disertai satuan dan konteks, sedangkan setiap berkas perlu diberi nama yang menjelaskan molekul, metode, atau tahap proses.

Tujuan, Input, dan Keluaran

Input resep meliputi parameter yang relevan dengan operasi array dengan skalar dan vektor tanpa loop eksplisit. Sebelum eksekusi, periksa tipe data, bentuk array, jalur berkas, dan ketersediaan pustaka. Keluaran harus dibedakan menjadi keluaran langsung program, hasil validasi, dan interpretasi. Keluaran langsung dapat berupa nilai, tabel, gambar, atau berkas; hasil validasi menyatakan apakah syarat minimum terpenuhi; interpretasi menjelaskan makna hasil tanpa melampaui metode yang digunakan.

Kode Siap Google Colab

```
import numpy as np

xyz = np.array([[1,0,0], [0,1,0], [0,0,1]], dtype=float)
pusat = xyz.mean(axis=0)
xyz_terpusat = xyz - pusat
```



```
print("Pusat awal:", pusat)
print("Pusat baru:", xyz_terpusat.mean(axis=0))
```

Cara Membaca dan Memvalidasi Hasil

Setelah sel dijalankan, baca pesan dari atas ke bawah dan pastikan tidak ada pengecualian yang terlewat. Verifikasi bahwa keluaran sesuai dengan koordinat yang ditranslasi terhadap pusat tertentu. Untuk keluaran numerik, periksa bilangan hingga, satuan, tanda, skala, dan batas kewajaran. Untuk berkas, periksa keberadaan, ukuran, encoding, dan kemampuan untuk dibuka kembali. Untuk grafik atau visualisasi, pastikan label sumbu, satuan, legenda, dan judul menjelaskan konteks data.

Risiko utama pada resep ini adalah Broadcasting yang berhasil secara sintaks belum tentu sesuai dengan sumbu yang dimaksud. Oleh karena itu, status keberhasilan tidak boleh disimpulkan hanya dari munculnya output. Jalankan kembali notebook dari sesi bersih, bandingkan hasil dengan perhitungan sederhana atau data acuan ketika tersedia, dan catat versi lingkungan. Bila hasil berubah, identifikasi apakah perubahan berasal dari input, urutan eksekusi, versi pustaka, perangkat, atau algoritma.

Pengembangan dan Latihan

835. Tambahkan assert pada shape sebelum dan setelah translasi.
836. Tambahkan satu pemeriksaan yang sengaja menghentikan proses ketika input tidak memenuhi syarat, lalu uji pesan kesalahannya.
837. Simpan parameter, status, waktu eksekusi, dan hasil utama ke JSON atau CSV; kemudian baca kembali berkas tersebut dalam runtime baru.
838. Tuliskan dua paragraf interpretasi: paragraf pertama hanya memuat fakta keluaran program, sedangkan paragraf kedua memuat makna dan keterbatasannya.

E.19 Menggunakan Aljabar Linear Dasar

Resep ini membahas matriks, determinan, eigenvalue, dan pemeriksaan simetri. Tujuannya bukan sekadar memperoleh keluaran yang tampak benar, tetapi membangun alur notebook yang dapat dijalankan ulang dan diperiksa. Pembaca perlu menyatakan input secara eksplisit, memisahkan konfigurasi dari proses utama, mencetak status penting, dan menyimpan artefak yang diperlukan. Hasil utama yang ditargetkan adalah nilai eigen dari matriks simetris sederhana. Setiap angka harus disertai satuan dan konteks, sedangkan setiap berkas perlu diberi nama yang menjelaskan molekul, metode, atau tahap proses.

Tujuan, Input, dan Keluaran

Input resep meliputi parameter yang relevan dengan matriks, determinan, eigenvalue, dan pemeriksaan simetri. Sebelum eksekusi, periksa tipe data, bentuk array, jalur berkas, dan ketersediaan pustaka. Keluaran harus dibedakan menjadi keluaran langsung program, hasil validasi, dan interpretasi. Keluaran langsung dapat berupa nilai, tabel, gambar, atau

berkas; hasil validasi menyatakan apakah syarat minimum terpenuhi; interpretasi menjelaskan makna hasil tanpa melampaui metode yang digunakan.

Kode Siap Google Colab

```
import numpy as np

A = np.array([[2.0, 0.5], [0.5, 1.0]])
assert np.allclose(A, A.T)
nilai, vektor = np.linalg.eigh(A)
print("Eigenvalue:", nilai)
print("Ortonormal:", np.allclose(vektor.T @ vektor, np.eye(2)))
```

Cara Membaca dan Memvalidasi Hasil

Setelah sel dijalankan, baca pesan dari atas ke bawah dan pastikan tidak ada pengecualian yang terlewat. Verifikasi bahwa keluaran sesuai dengan nilai eigen dari matriks simetris sederhana. Untuk keluaran numerik, periksa bilangan hingga, satuan, tanda, skala, dan batas kewajaran. Untuk berkas, periksa keberadaan, ukuran, encoding, dan kemampuan untuk dibuka kembali. Untuk grafik atau visualisasi, pastikan label sumbu, satuan, legenda, dan judul menjelaskan konteks data.

Risiko utama pada resep ini adalah `np.linalg.eig` dan `np.linalg.eigh` memiliki asumsi berbeda; gunakan versi Hermitian untuk matriks simetris. Oleh karena itu, status keberhasilan tidak boleh disimpulkan hanya dari munculnya output. Jalankan kembali notebook dari sesi bersih, bandingkan hasil dengan perhitungan sederhana atau data acuan ketika tersedia, dan catat versi lingkungan. Bila hasil berubah, identifikasi apakah perubahan berasal dari input, urutan eksekusi, versi pustaka, perangkat, atau algoritma.

Pengembangan dan Latihan

839. Rekonstruksi A dari eigenvalue dan eigenvector, lalu hitung galat maksimum.
840. Tambahkan satu pemeriksaan yang sengaja menghentikan proses ketika input tidak memenuhi syarat, lalu uji pesan kesalahannya.
841. Simpan parameter, status, waktu eksekusi, dan hasil utama ke JSON atau CSV; kemudian baca kembali berkas tersebut dalam runtime baru.
842. Tuliskan dua paragraf interpretasi: paragraf pertama hanya memuat fakta keluaran program, sedangkan paragraf kedua memuat makna dan keterbatasannya.

E.20 Melakukan Interpolasi Satu Dimensi

Resep ini membahas interpolasi linear, domain, dan pemeriksaan urutan x. Tujuannya bukan sekadar memperoleh keluaran yang tampak benar, tetapi membangun alur notebook yang dapat dijalankan ulang dan diperiksa. Pembaca perlu menyatakan input secara eksplisit, memisahkan konfigurasi dari proses utama, mencetak status penting, dan menyimpan artefak yang diperlukan. Hasil utama yang ditargetkan adalah perkiraan nilai

di antara titik data. Setiap angka harus disertai satuan dan konteks, sedangkan setiap berkas perlu diberi nama yang menjelaskan molekul, metode, atau tahap proses.

Tujuan, Input, dan Keluaran

Input resep meliputi parameter yang relevan dengan interpolasi linear, domain, dan pemeriksaan urutan x. Sebelum eksekusi, periksa tipe data, bentuk array, jalur berkas, dan ketersediaan pustaka. Keluaran harus dibedakan menjadi keluaran langsung program, hasil validasi, dan interpretasi. Keluaran langsung dapat berupa nilai, tabel, gambar, atau berkas; hasil validasi menyatakan apakah syarat minimum terpenuhi; interpretasi menjelaskan makna hasil tanpa melampaui metode yang digunakan.

Kode Siap Google Colab

```
import numpy as np

x = np.array([0.4, 0.6, 0.8, 1.0])
y = np.array([-0.7, -1.0, -1.1, -0.9])
x_baru = np.linspace(x.min(), x.max(), 13)
y_baru = np.interp(x_baru, x, y)
print(np.column_stack([x_baru, y_baru]))
```

Cara Membaca dan Memvalidasi Hasil

Setelah sel dijalankan, baca pesan dari atas ke bawah dan pastikan tidak ada pengecualian yang terlewat. Verifikasi bahwa keluaran sesuai dengan perkiraan nilai di antara titik data. Untuk keluaran numerik, periksa bilangan hingga, satuan, tanda, skala, dan batas kewajaran. Untuk berkas, periksa keberadaan, ukuran, encoding, dan kemampuan untuk dibuka kembali. Untuk grafik atau visualisasi, pastikan label sumbu, satuan, legenda, dan judul menjelaskan konteks data.

Risiko utama pada resep ini adalah Interpolasi bukan ekstrapolasi dan tidak menggantikan model fisik di luar domain data. Oleh karena itu, status keberhasilan tidak boleh disimpulkan hanya dari munculnya output. Jalankan kembali notebook dari sesi bersih, bandingkan hasil dengan perhitungan sederhana atau data acuan ketika tersedia, dan catat versi lingkungan. Bila hasil berubah, identifikasi apakah perubahan berasal dari input, urutan eksekusi, versi pustaka, perangkat, atau algoritma.

Pengembangan dan Latihan

843. Bandingkan titik minimum data asli dan interpolasi, lalu jelaskan keterbatasannya.
844. Tambahkan satu pemeriksaan yang sengaja menghentikan proses ketika input tidak memenuhi syarat, lalu uji pesan kesalahannya.
845. Simpan parameter, status, waktu eksekusi, dan hasil utama ke JSON atau CSV; kemudian baca kembali berkas tersebut dalam runtime baru.

846. Tuliskan dua paragraf interpretasi: paragraf pertama hanya memuat fakta keluaran program, sedangkan paragraf kedua memuat makna dan keterbatasannya.

E.21 Menghitung Turunan Numerik

Resep ini membahas gradient, jarak grid, dan sensitivitas terhadap interval. Tujuannya bukan sekadar memperoleh keluaran yang tampak benar, tetapi membangun alur notebook yang dapat dijalankan ulang dan diperiksa. Pembaca perlu menyatakan input secara eksplisit, memisahkan konfigurasi dari proses utama, mencetak status penting, dan menyimpan artefak yang diperlukan. Hasil utama yang ditargetkan adalah perkiraan gradien kurva energi. Setiap angka harus disertai satuan dan konteks, sedangkan setiap berkas perlu diberi nama yang menjelaskan molekul, metode, atau tahap proses.

Tujuan, Input, dan Keluaran

Input resep meliputi parameter yang relevan dengan gradient, jarak grid, dan sensitivitas terhadap interval. Sebelum eksekusi, periksa tipe data, bentuk array, jalur berkas, dan ketersediaan pustaka. Keluaran harus dibedakan menjadi keluaran langsung program, hasil validasi, dan interpretasi. Keluaran langsung dapat berupa nilai, tabel, gambar, atau berkas; hasil validasi menyatakan apakah syarat minimum terpenuhi; interpretasi menjelaskan makna hasil tanpa melampaui metode yang digunakan.

Kode Siap Google Colab

```
import numpy as np

r = np.linspace(0.5, 1.5, 21)
E = (r - 0.74)**2 - 1.1
dE_dr = np.gradient(E, r)
idx = np.argmin(np.abs(dE_dr))
print(f"Gradien mendekati nol pada r={r[idx]:.4f}")
```

Cara Membaca dan Memvalidasi Hasil

Setelah sel dijalankan, baca pesan dari atas ke bawah dan pastikan tidak ada pengecualian yang terlewat. Verifikasi bahwa keluaran sesuai dengan perkiraan gradien kurva energi. Untuk keluaran numerik, periksa bilangan hingga, satuan, tanda, skala, dan batas kewajaran. Untuk berkas, periksa keberadaan, ukuran, encoding, dan kemampuan untuk dibuka kembali. Untuk grafik atau visualisasi, pastikan label sumbu, satuan, legenda, dan judul menjelaskan konteks data.

Risiko utama pada resep ini adalah Turunan numerik menjadi bising bila data tidak halus atau interval tidak seragam tanpa penanganan tepat. Oleh karena itu, status keberhasilan tidak boleh disimpulkan hanya dari munculnya output. Jalankan kembali notebook dari sesi bersih, bandingkan hasil dengan perhitungan sederhana atau data acuan ketika tersedia, dan catat versi lingkungan. Bila hasil berubah, identifikasi apakah perubahan berasal dari input, urutan eksekusi, versi pustaka, perangkat, atau algoritma.

Pengembangan dan Latihan

847. Ulangi dengan 11 dan 101 titik, kemudian bandingkan posisi gradien nol.
848. Tambahkan satu pemeriksaan yang sengaja menghentikan proses ketika input tidak memenuhi syarat, lalu uji pesan kesalahannya.
849. Simpan parameter, status, waktu eksekusi, dan hasil utama ke JSON atau CSV; kemudian baca kembali berkas tersebut dalam runtime baru.
850. Tuliskan dua paragraf interpretasi: paragraf pertama hanya memuat fakta keluaran program, sedangkan paragraf kedua memuat makna dan keterbatasannya.

E.22 Menghitung Integral Numerik

Resep ini membahas trapezoid, grid, dan makna luas di bawah kurva. Tujuannya bukan sekadar memperoleh keluaran yang tampak benar, tetapi membangun alur notebook yang dapat dijalankan ulang dan diperiksa. Pembaca perlu menyatakan input secara eksplisit, memisahkan konfigurasi dari proses utama, mencetak status penting, dan menyimpan artefak yang diperlukan. Hasil utama yang ditargetkan adalah nilai integral diskret dengan satuan yang dinyatakan. Setiap angka harus disertai satuan dan konteks, sedangkan setiap berkas perlu diberi nama yang menjelaskan molekul, metode, atau tahap proses.

Tujuan, Input, dan Keluaran

Input resep meliputi parameter yang relevan dengan trapezoid, grid, dan makna luas di bawah kurva. Sebelum eksekusi, periksa tipe data, bentuk array, jalur berkas, dan ketersediaan pustaka. Keluaran harus dibedakan menjadi keluaran langsung program, hasil validasi, dan interpretasi. Keluaran langsung dapat berupa nilai, tabel, gambar, atau berkas; hasil validasi menyatakan apakah syarat minimum terpenuhi; interpretasi menjelaskan makna hasil tanpa melampaui metode yang digunakan.

Kode Siap Google Colab

```
import numpy as np

x = np.linspace(0, np.pi, 1001)
y = np.sin(x)
luas = np.trapezoid(y, x)
print(f"Integral sin(x) 0..pi = {luas:.10f}")
```

Cara Membaca dan Memvalidasi Hasil

Setelah sel dijalankan, baca pesan dari atas ke bawah dan pastikan tidak ada pengecualian yang terlewat. Verifikasi bahwa keluaran sesuai dengan nilai integral diskret dengan satuan yang dinyatakan. Untuk keluaran numerik, periksa bilangan hingga, satuan, tanda, skala, dan batas kewajaran. Untuk berkas, periksa keberadaan, ukuran, encoding, dan kemampuan untuk dibuka kembali. Untuk grafik atau visualisasi, pastikan label sumbu, satuan, legenda, dan judul menjelaskan konteks data.

Risiko utama pada resep ini adalah Nilai integral hanya bermakna jika sumbu, satuan, dan domain dinyatakan. Oleh karena itu, status keberhasilan tidak boleh disimpulkan hanya dari munculnya output. Jalankan kembali notebook dari sesi bersih, bandingkan hasil dengan perhitungan sederhana atau data acuan ketika tersedia, dan catat versi lingkungan. Bila hasil berubah, identifikasi apakah perubahan berasal dari input, urutan eksekusi, versi pustaka, perangkat, atau algoritma.

Pengembangan dan Latihan

851. Kurangi jumlah titik dan analisis perubahan galat terhadap nilai teoritis 2.
852. Tambahkan satu pemeriksaan yang sengaja menghentikan proses ketika input tidak memenuhi syarat, lalu uji pesan kesalahannya.
853. Simpan parameter, status, waktu eksekusi, dan hasil utama ke JSON atau CSV; kemudian baca kembali berkas tersebut dalam runtime baru.
854. Tuliskan dua paragraf interpretasi: paragraf pertama hanya memuat fakta keluaran program, sedangkan paragraf kedua memuat makna dan keterbatasannya.

E.23 Menggunakan Random Seed untuk Reprodusibilitas

Resep ini membahas generator acak modern, seed, dan sampel terkontrol. Tujuannya bukan sekadar memperoleh keluaran yang tampak benar, tetapi membangun alur notebook yang dapat dijalankan ulang dan diperiksa. Pembaca perlu menyatakan input secara eksplisit, memisahkan konfigurasi dari proses utama, mencetak status penting, dan menyimpan artefak yang diperlukan. Hasil utama yang ditargetkan adalah data simulasi yang dapat diulang. Setiap angka harus disertai satuan dan konteks, sedangkan setiap berkas perlu diberi nama yang menjelaskan molekul, metode, atau tahap proses.

Tujuan, Input, dan Keluaran

Input resep meliputi parameter yang relevan dengan generator acak modern, seed, dan sampel terkontrol. Sebelum eksekusi, periksa tipe data, bentuk array, jalur berkas, dan ketersediaan pustaka. Keluaran harus dibedakan menjadi keluaran langsung program, hasil validasi, dan interpretasi. Keluaran langsung dapat berupa nilai, tabel, gambar, atau berkas; hasil validasi menyatakan apakah syarat minimum terpenuhi; interpretasi menjelaskan makna hasil tanpa melampaui metode yang digunakan.

Kode Siap Google Colab

```
import numpy as np

rng = np.random.default_rng(seed=2026)
sampel = rng.normal(loc=0.0, scale=1.0, size=10)
print(sampel)
print("Rata-rata:", sampel.mean())
```

Cara Membaca dan Memvalidasi Hasil

Setelah sel dijalankan, baca pesan dari atas ke bawah dan pastikan tidak ada pengecualian yang terlewat. Verifikasi bahwa keluaran sesuai dengan data simulasi yang dapat diulang. Untuk keluaran numerik, periksa bilangan hingga, satuan, tanda, skala, dan batas kewajaran. Untuk berkas, periksa keberadaan, ukuran, encoding, dan kemampuan untuk dibuka kembali. Untuk grafik atau visualisasi, pastikan label sumbu, satuan, legenda, dan judul menjelaskan konteks data.

Risiko utama pada resep ini adalah Seed membuat urutan dapat diulang, tetapi tidak menjamin hasil identik antaralgoritma atau versi pustaka yang berbeda. Oleh karena itu, status keberhasilan tidak boleh disimpulkan hanya dari munculnya output. Jalankan kembali notebook dari sesi bersih, bandingkan hasil dengan perhitungan sederhana atau data acuan ketika tersedia, dan catat versi lingkungan. Bila hasil berubah, identifikasi apakah perubahan berasal dari input, urutan eksekusi, versi pustaka, perangkat, atau algoritma.

Pengembangan dan Latihan

855. Buat dua generator dengan seed sama dan buktikan kesamaan keluaran.
856. Tambahkan satu pemeriksaan yang sengaja menghentikan proses ketika input tidak memenuhi syarat, lalu uji pesan kesalahannya.
857. Simpan parameter, status, waktu eksekusi, dan hasil utama ke JSON atau CSV; kemudian baca kembali berkas tersebut dalam runtime baru.
858. Tuliskan dua paragraf interpretasi: paragraf pertama hanya memuat fakta keluaran program, sedangkan paragraf kedua memuat makna dan keterbatasannya.

E.24 Menyimpan dan Membaca Array

Resep ini membahas save, load, savetxt, delimiter, dan validasi pembacaan ulang. Tujuannya bukan sekadar memperoleh keluaran yang tampak benar, tetapi membangun alur notebook yang dapat dijalankan ulang dan diperiksa. Pembaca perlu menyatakan input secara eksplisit, memisahkan konfigurasi dari proses utama, mencetak status penting, dan menyimpan artefak yang diperlukan. Hasil utama yang ditargetkan adalah berkas NPY dan CSV dengan data identik. Setiap angka harus disertai satuan dan konteks, sedangkan setiap berkas perlu diberi nama yang menjelaskan molekul, metode, atau tahap proses.

Tujuan, Input, dan Keluaran

Input resep meliputi parameter yang relevan dengan save, load, savetxt, delimiter, dan validasi pembacaan ulang. Sebelum eksekusi, periksa tipe data, bentuk array, jalur berkas, dan ketersediaan pustaka. Keluaran harus dibedakan menjadi keluaran langsung program, hasil validasi, dan interpretasi. Keluaran langsung dapat berupa nilai, tabel, gambar, atau berkas; hasil validasi menyatakan apakah syarat minimum terpenuhi; interpretasi menjelaskan makna hasil tanpa melampaui metode yang digunakan.

Kode Siap Google Colab

```
import numpy as np

A = np.arange(12, dtype=float).reshape(4,3)
np.save("array.npy", A)
np.savetxt("array.csv", A, delimiter=",", fmt="%.8f")

B = np.load("array.npy")
C = np.loadtxt("array.csv", delimiter=",")
print(np.allclose(A, B), np.allclose(A, C))
```

Cara Membaca dan Memvalidasi Hasil

Setelah sel dijalankan, baca pesan dari atas ke bawah dan pastikan tidak ada pengecualian yang terlewat. Verifikasi bahwa keluaran sesuai dengan berkas NPY dan CSV dengan data identik. Untuk keluaran numerik, periksa bilangan hingga, satuan, tanda, skala, dan batas kewajaran. Untuk berkas, periksa keberadaan, ukuran, encoding, dan kemampuan untuk dibuka kembali. Untuk grafik atau visualisasi, pastikan label sumbu, satuan, legenda, dan judul menjelaskan konteks data.

Risiko utama pada resep ini adalah CSV mudah dibaca lintas aplikasi tetapi tidak menyimpan dtype dan metadata sebaik NPY. Oleh karena itu, status keberhasilan tidak boleh disimpulkan hanya dari munculnya output. Jalankan kembali notebook dari sesi bersih, bandingkan hasil dengan perhitungan sederhana atau data acuan ketika tersedia, dan catat versi lingkungan. Bila hasil berubah, identifikasi apakah perubahan berasal dari input, urutan eksekusi, versi pustaka, perangkat, atau algoritma.

Pengembangan dan Latihan

859. Tambahkan nama kolom menggunakan Pandas dan simpan metadata shape dalam JSON.
860. Tambahkan satu pemeriksaan yang sengaja menghentikan proses ketika input tidak memenuhi syarat, lalu uji pesan kesalahannya.
861. Simpan parameter, status, waktu eksekusi, dan hasil utama ke JSON atau CSV; kemudian baca kembali berkas tersebut dalam runtime baru.
862. Tuliskan dua paragraf interpretasi: paragraf pertama hanya memuat fakta keluaran program, sedangkan paragraf kedua memuat makna dan keterbatasannya.

E.3 Pandas dan Pengelolaan Data

E.25 Membuat DataFrame dari Dictionary

Resep ini membahas kolom, indeks, dtype, dan ringkasan struktur. Tujuannya bukan sekadar memperoleh keluaran yang tampak benar, tetapi membangun alur notebook yang dapat dijalankan ulang dan diperiksa. Pembaca perlu menyatakan input secara eksplisit, memisahkan konfigurasi dari proses utama, mencetak status penting, dan menyimpan

artefak yang diperlukan. Hasil utama yang ditargetkan adalah tabel hasil perhitungan yang mudah disaring. Setiap angka harus disertai satuan dan konteks, sedangkan setiap berkas perlu diberi nama yang menjelaskan molekul, metode, atau tahap proses.

Tujuan, Input, dan Keluaran

Input resep meliputi parameter yang relevan dengan kolom, indeks, dtype, dan ringkasan struktur. Sebelum eksekusi, periksa tipe data, bentuk array, jalur berkas, dan ketersediaan pustaka. Keluaran harus dibedakan menjadi keluaran langsung program, hasil validasi, dan interpretasi. Keluaran langsung dapat berupa nilai, tabel, gambar, atau berkas; hasil validasi menyatakan apakah syarat minimum terpenuhi; interpretasi menjelaskan makna hasil tanpa melampaui metode yang digunakan.

Kode Siap Google Colab

```
import pandas as pd

data = {
    "jarak_A": [0.6, 0.7, 0.74, 0.8, 1.0],
    "energi_Eh": [-1.05, -1.10, -1.117, -1.11, -1.00],
    "konvergen": [True, True, True, True, True],
}
df = pd.DataFrame(data)
print(df)
print(df.dtypes)
```

Cara Membaca dan Memvalidasi Hasil

Setelah sel dijalankan, baca pesan dari atas ke bawah dan pastikan tidak ada pengecualian yang terlewat. Verifikasi bahwa keluaran sesuai dengan tabel hasil perhitungan yang mudah disaring. Untuk keluaran numerik, periksa bilangan hingga, satuan, tanda, skala, dan batas kewajaran. Untuk berkas, periksa keberadaan, ukuran, encoding, dan kemampuan untuk dibuka kembali. Untuk grafik atau visualisasi, pastikan label sumbu, satuan, legenda, dan judul menjelaskan konteks data.

Risiko utama pada resep ini adalah Kolom numerik dapat terbaca sebagai teks bila ada simbol atau pemisah desimal yang tidak konsisten. Oleh karena itu, status keberhasilan tidak boleh disimpulkan hanya dari munculnya output. Jalankan kembali notebook dari sesi bersih, bandingkan hasil dengan perhitungan sederhana atau data acuan ketika tersedia, dan catat versi lingkungan. Bila hasil berubah, identifikasi apakah perubahan berasal dari input, urutan eksekusi, versi pustaka, perangkat, atau algoritma.

Pengembangan dan Latihan

863. Tambahkan kolom energi eV dan urutkan tabel berdasarkan energi.
864. Tambahkan satu pemeriksaan yang sengaja menghentikan proses ketika input tidak memenuhi syarat, lalu uji pesan kesalahannya.

865. Simpan parameter, status, waktu eksekusi, dan hasil utama ke JSON atau CSV; kemudian baca kembali berkas tersebut dalam runtime baru.
866. Tuliskan dua paragraf interpretasi: paragraf pertama hanya memuat fakta keluaran program, sedangkan paragraf kedua memuat makna dan keterbatasannya.

E.26 Membaca CSV dengan Validasi Kolom

Resep ini membahas `read_csv`, `required columns`, dan konversi tipe. Tujuannya bukan sekadar memperoleh keluaran yang tampak benar, tetapi membangun alur notebook yang dapat dijalankan ulang dan diperiksa. Pembaca perlu menyatakan input secara eksplisit, memisahkan konfigurasi dari proses utama, mencetak status penting, dan menyimpan artefak yang diperlukan. Hasil utama yang ditargetkan adalah `DataFrame` yang memenuhi skema minimum. Setiap angka harus disertai satuan dan konteks, sedangkan setiap berkas perlu diberi nama yang menjelaskan molekul, metode, atau tahap proses.

Tujuan, Input, dan Keluaran

Input resep meliputi parameter yang relevan dengan `read_csv`, `required columns`, dan konversi tipe. Sebelum eksekusi, periksa tipe data, bentuk array, jalur berkas, dan ketersediaan pustaka. Keluaran harus dibedakan menjadi keluaran langsung program, hasil validasi, dan interpretasi. Keluaran langsung dapat berupa nilai, tabel, gambar, atau berkas; hasil validasi menyatakan apakah syarat minimum terpenuhi; interpretasi menjelaskan makna hasil tanpa melampaui metode yang digunakan.

Kode Siap Google Colab

```
import pandas as pd

wajib = {"jarak_A", "energi_Eh"}
df = pd.read_csv("array.csv", header=None, names=["x", "y", "z"])
print(df.head())
assert set(df.columns) == {"x", "y", "z"}
```

Cara Membaca dan Memvalidasi Hasil

Setelah sel dijalankan, baca pesan dari atas ke bawah dan pastikan tidak ada pengecualian yang terlewat. Verifikasi bahwa keluaran sesuai dengan `DataFrame` yang memenuhi skema minimum. Untuk keluaran numerik, periksa bilangan hingga, satuan, tanda, skala, dan batas kewajaran. Untuk berkas, periksa keberadaan, ukuran, encoding, dan kemampuan untuk dibuka kembali. Untuk grafik atau visualisasi, pastikan label sumbu, satuan, legenda, dan judul menjelaskan konteks data.

Risiko utama pada resep ini adalah Membaca tanpa memeriksa header dapat menggeser baris pertama data menjadi nama kolom. Oleh karena itu, status keberhasilan tidak boleh disimpulkan hanya dari munculnya output. Jalankan kembali notebook dari sesi bersih, bandingkan hasil dengan perhitungan sederhana atau data acuan ketika tersedia, dan catat

versi lingkungan. Bila hasil berubah, identifikasi apakah perubahan berasal dari input, urutan eksekusi, versi pustaka, perangkat, atau algoritma.

Pengembangan dan Latihan

867. Buat CSV dengan header, lalu validasi jumlah baris dan nilai hilang.
868. Tambahkan satu pemeriksaan yang sengaja menghentikan proses ketika input tidak memenuhi syarat, lalu uji pesan kesalahannya.
869. Simpan parameter, status, waktu eksekusi, dan hasil utama ke JSON atau CSV; kemudian baca kembali berkas tersebut dalam runtime baru.
870. Tuliskan dua paragraf interpretasi: paragraf pertama hanya memuat fakta keluaran program, sedangkan paragraf kedua memuat makna dan keterbatasannya.

E.27 Membersihkan Nilai Hilang dan Duplikat

Resep ini membahas isna, dropna, fillna, duplicated, dan audit perubahan. Tujuannya bukan sekadar memperoleh keluaran yang tampak benar, tetapi membangun alur notebook yang dapat dijalankan ulang dan diperiksa. Pembaca perlu menyatakan input secara eksplisit, memisahkan konfigurasi dari proses utama, mencetak status penting, dan menyimpan artefak yang diperlukan. Hasil utama yang ditargetkan adalah tabel bersih beserta jumlah baris yang dihapus. Setiap angka harus disertai satuan dan konteks, sedangkan setiap berkas perlu diberi nama yang menjelaskan molekul, metode, atau tahap proses.

Tujuan, Input, dan Keluaran

Input resep meliputi parameter yang relevan dengan isna, dropna, fillna, duplicated, dan audit perubahan. Sebelum eksekusi, periksa tipe data, bentuk array, jalur berkas, dan ketersediaan pustaka. Keluaran harus dibedakan menjadi keluaran langsung program, hasil validasi, dan interpretasi. Keluaran langsung dapat berupa nilai, tabel, gambar, atau berkas; hasil validasi menyatakan apakah syarat minimum terpenuhi; interpretasi menjelaskan makna hasil tanpa melampaui metode yang digunakan.

Kode Siap Google Colab

```
import pandas as pd
import numpy as np

df = pd.DataFrame({"id": [1, 2, 2, 3], "nilai": [1.0, np.nan, np.nan, 3.0]})
print("Awal:", len(df))
df2 = df.drop_duplicates().dropna(subset=["nilai"]).copy()
print("Akhir:", len(df2))
print(df2)
```

Cara Membaca dan Memvalidasi Hasil

Setelah sel dijalankan, baca pesan dari atas ke bawah dan pastikan tidak ada pengecualian yang terlewat. Verifikasi bahwa keluaran sesuai dengan tabel bersih beserta jumlah baris

yang dihapus. Untuk keluaran numerik, periksa bilangan hingga, satuan, tanda, skala, dan batas kewajaran. Untuk berkas, periksa keberadaan, ukuran, encoding, dan kemampuan untuk dibuka kembali. Untuk grafik atau visualisasi, pastikan label sumbu, satuan, legenda, dan judul menjelaskan konteks data.

Risiko utama pada resep ini adalah Menghapus data tanpa menyimpan alasan dan jumlah perubahan mengurangi keterlacakan. Oleh karena itu, status keberhasilan tidak boleh disimpulkan hanya dari munculnya output. Jalankan kembali notebook dari sesi bersih, bandingkan hasil dengan perhitungan sederhana atau data acuan ketika tersedia, dan catat versi lingkungan. Bila hasil berubah, identifikasi apakah perubahan berasal dari input, urutan eksekusi, versi pustaka, perangkat, atau algoritma.

Pengembangan dan Latihan

871. Buat laporan jumlah NaN, duplikat, dan baris tersisa dalam dictionary.
872. Tambahkan satu pemeriksaan yang sengaja menghentikan proses ketika input tidak memenuhi syarat, lalu uji pesan kesalahannya.
873. Simpan parameter, status, waktu eksekusi, dan hasil utama ke JSON atau CSV; kemudian baca kembali berkas tersebut dalam runtime baru.
874. Tuliskan dua paragraf interpretasi: paragraf pertama hanya memuat fakta keluaran program, sedangkan paragraf kedua memuat makna dan keterbatasannya.

E.28 Menyaring dan Mengurutkan Data

Resep ini membahas boolean indexing, query, sort_values, dan reset_index. Tujuannya bukan sekadar memperoleh keluaran yang tampak benar, tetapi membangun alur notebook yang dapat dijalankan ulang dan diperiksa. Pembaca perlu menyatakan input secara eksplisit, memisahkan konfigurasi dari proses utama, mencetak status penting, dan menyimpan artefak yang diperlukan. Hasil utama yang ditargetkan adalah subset titik valid dengan urutan energi. Setiap angka harus disertai satuan dan konteks, sedangkan setiap berkas perlu diberi nama yang menjelaskan molekul, metode, atau tahap proses.

Tujuan, Input, dan Keluaran

Input resep meliputi parameter yang relevan dengan boolean indexing, query, sort_values, dan reset_index. Sebelum eksekusi, periksa tipe data, bentuk array, jalur berkas, dan ketersediaan pustaka. Keluaran harus dibedakan menjadi keluaran langsung program, hasil validasi, dan interpretasi. Keluaran langsung dapat berupa nilai, tabel, gambar, atau berkas; hasil validasi menyatakan apakah syarat minimum terpenuhi; interpretasi menjelaskan makna hasil tanpa melampaui metode yang digunakan.

Kode Siap Google Colab

```
import pandas as pd

df = pd.DataFrame({
    "jarak": [0.6, 0.7, 0.74, 0.8, 1.0],
```

```
"energi": [-1.05, -1.10, -1.117, -1.11, -1.00],  
"ok": [True, True, True, False, True]  
})  
valid = df.loc[df["ok"]].sort_values("energi").reset_index(drop=True)  
print(valid)
```

Cara Membaca dan Memvalidasi Hasil

Setelah sel dijalankan, baca pesan dari atas ke bawah dan pastikan tidak ada pengecualian yang terlewat. Verifikasi bahwa keluaran sesuai dengan subset titik valid dengan urutan energi. Untuk keluaran numerik, periksa bilangan hingga, satuan, tanda, skala, dan batas kewajaran. Untuk berkas, periksa keberadaan, ukuran, encoding, dan kemampuan untuk dibuka kembali. Untuk grafik atau visualisasi, pastikan label sumbu, satuan, legenda, dan judul menjelaskan konteks data.

Risiko utama pada resep ini adalah Filter yang menghasilkan salinan atau view perlu ditangani dengan `.copy()` sebelum modifikasi lanjutan. Oleh karena itu, status keberhasilan tidak boleh disimpulkan hanya dari munculnya output. Jalankan kembali notebook dari sesi bersih, bandingkan hasil dengan perhitungan sederhana atau data acuan ketika tersedia, dan catat versi lingkungan. Bila hasil berubah, identifikasi apakah perubahan berasal dari input, urutan eksekusi, versi pustaka, perangkat, atau algoritma.

Pengembangan dan Latihan

875. Tambahkan filter rentang jarak dan tampilkan tiga energi terendah.
876. Tambahkan satu pemeriksaan yang sengaja menghentikan proses ketika input tidak memenuhi syarat, lalu uji pesan kesalahannya.
877. Simpan parameter, status, waktu eksekusi, dan hasil utama ke JSON atau CSV; kemudian baca kembali berkas tersebut dalam runtime baru.
878. Tuliskan dua paragraf interpretasi: paragraf pertama hanya memuat fakta keluaran program, sedangkan paragraf kedua memuat makna dan keterbatasannya.

E.29 Mengelompokkan Data dengan *groupby*

Resep ini membahas pengelompokan, agregasi, dan penamaan hasil. Tujuannya bukan sekadar memperoleh keluaran yang tampak benar, tetapi membangun alur notebook yang dapat dijalankan ulang dan diperiksa. Pembaca perlu menyatakan input secara eksplisit, memisahkan konfigurasi dari proses utama, mencetak status penting, dan menyimpan artefak yang diperlukan. Hasil utama yang ditargetkan adalah ringkasan energi per metode. Setiap angka harus disertai satuan dan konteks, sedangkan setiap berkas perlu diberi nama yang menjelaskan molekul, metode, atau tahap proses.

Tujuan, Input, dan Keluaran

Input resep meliputi parameter yang relevan dengan pengelompokan, agregasi, dan penamaan hasil. Sebelum eksekusi, periksa tipe data, bentuk array, jalur berkas, dan ketersediaan pustaka. Keluaran harus dibedakan menjadi keluaran langsung program,

hasil validasi, dan interpretasi. Keluaran langsung dapat berupa nilai, tabel, gambar, atau berkas; hasil validasi menyatakan apakah syarat minimum terpenuhi; interpretasi menjelaskan makna hasil tanpa melampaui metode yang digunakan.

Kode Siap Google Colab

```
import pandas as pd

df = pd.DataFrame({
    "metode":["HF","HF","DFT","DFT"],
    "energi":[-75.9,-76.0,-76.3,-76.4],
    "waktu_s":[3.2,3.4,8.1,8.5]
})
ringkas = df.groupby("metode").agg(
    energi_rata=("energi","mean"),
    waktu_median=("waktu_s","median"),
    n=("energi","size"),
)
print(ringkas)
```

Cara Membaca dan Memvalidasi Hasil

Setelah sel dijalankan, baca pesan dari atas ke bawah dan pastikan tidak ada pengecualian yang terlewat. Verifikasi bahwa keluaran sesuai dengan ringkasan energi per metode. Untuk keluaran numerik, periksa bilangan hingga, satuan, tanda, skala, dan batas kewajaran. Untuk berkas, periksa keberadaan, ukuran, encoding, dan kemampuan untuk dibuka kembali. Untuk grafik atau visualisasi, pastikan label sumbu, satuan, legenda, dan judul menjelaskan konteks data.

Risiko utama pada resep ini adalah Rata-rata energi dari sistem atau basis berbeda tidak boleh dibandingkan tanpa konteks. Oleh karena itu, status keberhasilan tidak boleh disimpulkan hanya dari munculnya output. Jalankan kembali notebook dari sesi bersih, bandingkan hasil dengan perhitungan sederhana atau data acuan ketika tersedia, dan catat versi lingkungan. Bila hasil berubah, identifikasi apakah perubahan berasal dari input, urutan eksekusi, versi pustaka, perangkat, atau algoritma.

Pengembangan dan Latihan

879. Tambahkan simpangan baku dan ekspor ringkasan ke CSV.
880. Tambahkan satu pemeriksaan yang sengaja menghentikan proses ketika input tidak memenuhi syarat, lalu uji pesan kesalahannya.
881. Simpan parameter, status, waktu eksekusi, dan hasil utama ke JSON atau CSV; kemudian baca kembali berkas tersebut dalam runtime baru.
882. Tuliskan dua paragraf interpretasi: paragraf pertama hanya memuat fakta keluaran program, sedangkan paragraf kedua memuat makna dan keterbatasannya.

E.30 Menggabungkan Dua Tabel dengan merge

Resep ini membahas key, inner/left join, suffix, dan validasi cardinality. Tujuannya bukan sekadar memperoleh keluaran yang tampak benar, tetapi membangun alur notebook yang dapat dijalankan ulang dan diperiksa. Pembaca perlu menyatakan input secara eksplisit, memisahkan konfigurasi dari proses utama, mencetak status penting, dan menyimpan artefak yang diperlukan. Hasil utama yang ditargetkan adalah tabel gabungan parameter dan hasil. Setiap angka harus disertai satuan dan konteks, sedangkan setiap berkas perlu diberi nama yang menjelaskan molekul, metode, atau tahap proses.

Tujuan, Input, dan Keluaran

Input resep meliputi parameter yang relevan dengan key, inner/left join, suffix, dan validasi cardinality. Sebelum eksekusi, periksa tipe data, bentuk array, jalur berkas, dan ketersediaan pustaka. Keluaran harus dibedakan menjadi keluaran langsung program, hasil validasi, dan interpretasi. Keluaran langsung dapat berupa nilai, tabel, gambar, atau berkas; hasil validasi menyatakan apakah syarat minimum terpenuhi; interpretasi menjelaskan makna hasil tanpa melampaui metode yang digunakan.

Kode Siap Google Colab

```
import pandas as pd

param = pd.DataFrame({"job_id": [1, 2], "method": ["HF", "DFT"]})
hasil = pd.DataFrame({"job_id": [1, 2], "energy": [-1.1, -1.2]})
gabung = param.merge(hasil, on="job_id", how="inner", validate="one_to_one")
print(gabung)
```

Cara Membaca dan Memvalidasi Hasil

Setelah sel dijalankan, baca pesan dari atas ke bawah dan pastikan tidak ada pengecualian yang terlewat. Verifikasi bahwa keluaran sesuai dengan tabel gabungan parameter dan hasil. Untuk keluaran numerik, periksa bilangan hingga, satuan, tanda, skala, dan batas kewajaran. Untuk berkas, periksa keberadaan, ukuran, encoding, dan kemampuan untuk dibuka kembali. Untuk grafik atau visualisasi, pastikan label sumbu, satuan, legenda, dan judul menjelaskan konteks data.

Risiko utama pada resep ini adalah Key duplikat dapat menggandakan baris tanpa disadari bila cardinality tidak divalidasi. Oleh karena itu, status keberhasilan tidak boleh disimpulkan hanya dari munculnya output. Jalankan kembali notebook dari sesi bersih, bandingkan hasil dengan perhitungan sederhana atau data acuan ketika tersedia, dan catat versi lingkungan. Bila hasil berubah, identifikasi apakah perubahan berasal dari input, urutan eksekusi, versi pustaka, perangkat, atau algoritma.

Pengembangan dan Latihan

883. Uji merge dengan satu job_id yang hilang dan jelaskan perbedaan inner dan left join.
884. Tambahkan satu pemeriksaan yang sengaja menghentikan proses ketika input tidak memenuhi syarat, lalu uji pesan kesalahannya.
885. Simpan parameter, status, waktu eksekusi, dan hasil utama ke JSON atau CSV; kemudian baca kembali berkas tersebut dalam runtime baru.
886. Tuliskan dua paragraf interpretasi: paragraf pertama hanya memuat fakta keluaran program, sedangkan paragraf kedua memuat makna dan keterbatasannya.

E.31 Membuat Pivot Table

Resep ini membahas pivot_table, index, columns, values, dan fungsi agregasi. Tujuannya bukan sekadar memperoleh keluaran yang tampak benar, tetapi membangun alur notebook yang dapat dijalankan ulang dan diperiksa. Pembaca perlu menyatakan input secara eksplisit, memisahkan konfigurasi dari proses utama, mencetak status penting, dan menyimpan artefak yang diperlukan. Hasil utama yang ditargetkan adalah matriks perbandingan metode dan basis. Setiap angka harus disertai satuan dan konteks, sedangkan setiap berkas perlu diberi nama yang menjelaskan molekul, metode, atau tahap proses.

Tujuan, Input, dan Keluaran

Input resep meliputi parameter yang relevan dengan pivot_table, index, columns, values, dan fungsi agregasi. Sebelum eksekusi, periksa tipe data, bentuk array, jalur berkas, dan ketersediaan pustaka. Keluaran harus dibedakan menjadi keluaran langsung program, hasil validasi, dan interpretasi. Keluaran langsung dapat berupa nilai, tabel, gambar, atau berkas; hasil validasi menyatakan apakah syarat minimum terpenuhi; interpretasi menjelaskan makna hasil tanpa melampaui metode yang digunakan.

Kode Siap Google Colab

```
import pandas as pd

df = pd.DataFrame({
    "method": ["HF", "HF", "DFT", "DFT"],
    "basis": ["STO-3G", "6-31G", "STO-3G", "6-31G"],
    "energy": [-74.9, -75.9, -75.1, -76.2]
})
pivot = df.pivot_table(index="method", columns="basis", values="energy",
    aggfunc="mean")
print(pivot)
```

Cara Membaca dan Memvalidasi Hasil

Setelah sel dijalankan, baca pesan dari atas ke bawah dan pastikan tidak ada pengecualian yang terlewat. Verifikasi bahwa keluaran sesuai dengan matriks perbandingan metode

dan basis. Untuk keluaran numerik, periksa bilangan hingga, satuan, tanda, skala, dan batas kewajaran. Untuk berkas, periksa keberadaan, ukuran, encoding, dan kemampuan untuk dibuka kembali. Untuk grafik atau visualisasi, pastikan label sumbu, satuan, legenda, dan judul menjelaskan konteks data.

Risiko utama pada resep ini adalah Sel kosong pada pivot dapat berarti kombinasi belum dihitung, bukan nilai nol. Oleh karena itu, status keberhasilan tidak boleh disimpulkan hanya dari munculnya output. Jalankan kembali notebook dari sesi bersih, bandingkan hasil dengan perhitungan sederhana atau data acuan ketika tersedia, dan catat versi lingkungan. Bila hasil berubah, identifikasi apakah perubahan berasal dari input, urutan eksekusi, versi pustaka, perangkat, atau algoritma.

Pengembangan dan Latihan

887. Tambahkan kolom waktu dan buat pivot terpisah untuk durasi.
888. Tambahkan satu pemeriksaan yang sengaja menghentikan proses ketika input tidak memenuhi syarat, lalu uji pesan kesalahannya.
889. Simpan parameter, status, waktu eksekusi, dan hasil utama ke JSON atau CSV; kemudian baca kembali berkas tersebut dalam runtime baru.
890. Tuliskan dua paragraf interpretasi: paragraf pertama hanya memuat fakta keluaran program, sedangkan paragraf kedua memuat makna dan keterbatasannya.

E.32 Menghasilkan Statistik Deskriptif

Resep ini membahas describe, quantile, median, dan IQR. Tujuannya bukan sekadar memperoleh keluaran yang tampak benar, tetapi membangun alur notebook yang dapat dijalankan ulang dan diperiksa. Pembaca perlu menyatakan input secara eksplisit, memisahkan konfigurasi dari proses utama, mencetak status penting, dan menyimpan artefak yang diperlukan. Hasil utama yang ditargetkan adalah ringkasan statistik dengan pemeriksaan outlier awal. Setiap angka harus disertai satuan dan konteks, sedangkan setiap berkas perlu diberi nama yang menjelaskan molekul, metode, atau tahap proses.

Tujuan, Input, dan Keluaran

Input resep meliputi parameter yang relevan dengan describe, quantile, median, dan IQR. Sebelum eksekusi, periksa tipe data, bentuk array, jalur berkas, dan ketersediaan pustaka. Keluaran harus dibedakan menjadi keluaran langsung program, hasil validasi, dan interpretasi. Keluaran langsung dapat berupa nilai, tabel, gambar, atau berkas; hasil validasi menyatakan apakah syarat minimum terpenuhi; interpretasi menjelaskan makna hasil tanpa melampaui metode yang digunakan.

Kode Siap Google Colab

```
import pandas as pd

s = pd.Series([1.0,1.1,1.2,1.15,5.0], name="waktu_s")
q1, q3 = s.quantile([0.25,0.75])
```

```
iqr = q3-q1
bawah, atas = q1-1.5*iqr, q3+1.5*iqr
print(s.describe())
print("Outlier:", s[(s<bawah)|(s>atas)].tolist())
```

Cara Membaca dan Memvalidasi Hasil

Setelah sel dijalankan, baca pesan dari atas ke bawah dan pastikan tidak ada pengecualian yang terlewat. Verifikasi bahwa keluaran sesuai dengan ringkasan statistik dengan pemeriksaan outlier awal. Untuk keluaran numerik, periksa bilangan hingga, satuan, tanda, skala, dan batas kewajaran. Untuk berkas, periksa keberadaan, ukuran, encoding, dan kemampuan untuk dibuka kembali. Untuk grafik atau visualisasi, pastikan label sumbu, satuan, legenda, dan judul menjelaskan konteks data.

Risiko utama pada resep ini adalah Outlier statistik belum tentu kesalahan; konteks proses dan log harus diperiksa. Oleh karena itu, status keberhasilan tidak boleh disimpulkan hanya dari munculnya output. Jalankan kembali notebook dari sesi bersih, bandingkan hasil dengan perhitungan sederhana atau data acuan ketika tersedia, dan catat versi lingkungan. Bila hasil berubah, identifikasi apakah perubahan berasal dari input, urutan eksekusi, versi pustaka, perangkat, atau algoritma.

Pengembangan dan Latihan

891. Bandingkan aturan IQR dengan z-score untuk data yang sama.
892. Tambahkan satu pemeriksaan yang sengaja menghentikan proses ketika input tidak memenuhi syarat, lalu uji pesan kesalahannya.
893. Simpan parameter, status, waktu eksekusi, dan hasil utama ke JSON atau CSV; kemudian baca kembali berkas tersebut dalam runtime baru.
894. Tuliskan dua paragraf interpretasi: paragraf pertama hanya memuat fakta keluaran program, sedangkan paragraf kedua memuat makna dan keterbatasannya.

E.33 Membuat Grafik dari DataFrame

Resep ini membahas integrasi Pandas dan Matplotlib, label sumbu, dan penyimpanan gambar. Tujuannya bukan sekadar memperoleh keluaran yang tampak benar, tetapi membangun alur notebook yang dapat dijalankan ulang dan diperiksa. Pembaca perlu menyatakan input secara eksplisit, memisahkan konfigurasi dari proses utama, mencetak status penting, dan menyimpan artefak yang diperlukan. Hasil utama yang ditargetkan adalah grafik kurva energi yang tersimpan sebagai PNG. Setiap angka harus disertai satuan dan konteks, sedangkan setiap berkas perlu diberi nama yang menjelaskan molekul, metode, atau tahap proses.

Tujuan, Input, dan Keluaran

Input resep meliputi parameter yang relevan dengan integrasi Pandas dan Matplotlib, label sumbu, dan penyimpanan gambar. Sebelum eksekusi, periksa tipe data, bentuk array, jalur berkas, dan ketersediaan pustaka. Keluaran harus dibedakan menjadi keluaran

langsung program, hasil validasi, dan interpretasi. Keluaran langsung dapat berupa nilai, tabel, gambar, atau berkas; hasil validasi menyatakan apakah syarat minimum terpenuhi; interpretasi menjelaskan makna hasil tanpa melampaui metode yang digunakan.

Kode Siap Google Colab

```
import pandas as pd
import matplotlib.pyplot as plt

df = pd.DataFrame({"r": [0.6, 0.7, 0.74, 0.8, 1.0], "E": [-1.05, -1.10, -1.117, -1.11, -1.0]})
ax = df.plot(x="r", y="E", marker="o", legend=False)
ax.set_xlabel("Jarak H-H (Å)")
ax.set_ylabel("Energi total (Hartree)")
ax.set_title("Kurva Energi Potensial H2")
plt.tight_layout()
plt.savefig("kurva_energi.png", dpi=200)
plt.show()
```

Cara Membaca dan Memvalidasi Hasil

Setelah sel dijalankan, baca pesan dari atas ke bawah dan pastikan tidak ada pengecualian yang terlewat. Verifikasi bahwa keluaran sesuai dengan grafik kurva energi yang tersimpan sebagai PNG. Untuk keluaran numerik, periksa bilangan hingga, satuan, tanda, skala, dan batas kewajaran. Untuk berkas, periksa keberadaan, ukuran, encoding, dan kemampuan untuk dibuka kembali. Untuk grafik atau visualisasi, pastikan label sumbu, satuan, legenda, dan judul menjelaskan konteks data.

Risiko utama pada resep ini adalah Grafik tanpa satuan, metode, dan basis set dapat menyesatkan meskipun bentuknya benar. Oleh karena itu, status keberhasilan tidak boleh disimpulkan hanya dari munculnya output. Jalankan kembali notebook dari sesi bersih, bandingkan hasil dengan perhitungan sederhana atau data acuan ketika tersedia, dan catat versi lingkungan. Bila hasil berubah, identifikasi apakah perubahan berasal dari input, urutan eksekusi, versi pustaka, perangkat, atau algoritma.

Pengembangan dan Latihan

895. Tambahkan anotasi titik minimum dan tulis metode pada judul.
896. Tambahkan satu pemeriksaan yang sengaja menghentikan proses ketika input tidak memenuhi syarat, lalu uji pesan kesalahannya.
897. Simpan parameter, status, waktu eksekusi, dan hasil utama ke JSON atau CSV; kemudian baca kembali berkas tersebut dalam runtime baru.
898. Tuliskan dua paragraf interpretasi: paragraf pertama hanya memuat fakta keluaran program, sedangkan paragraf kedua memuat makna dan keterbatasannya.

E.34 Mengekspor Data ke CSV dan Excel

Resep ini membahas `to_csv`, `to_excel`, `index`, dan verifikasi berkas. Tujuannya bukan sekadar memperoleh keluaran yang tampak benar, tetapi membangun alur notebook yang dapat dijalankan ulang dan diperiksa. Pembaca perlu menyatakan input secara eksplisit, memisahkan konfigurasi dari proses utama, mencetak status penting, dan menyimpan artefak yang diperlukan. Hasil utama yang ditargetkan adalah dua format tabel yang dapat dibuka kembali. Setiap angka harus disertai satuan dan konteks, sedangkan setiap berkas perlu diberi nama yang menjelaskan molekul, metode, atau tahap proses.

Tujuan, Input, dan Keluaran

Input resep meliputi parameter yang relevan dengan `to_csv`, `to_excel`, `index`, dan verifikasi berkas. Sebelum eksekusi, periksa tipe data, bentuk array, jalur berkas, dan ketersediaan pustaka. Keluaran harus dibedakan menjadi keluaran langsung program, hasil validasi, dan interpretasi. Keluaran langsung dapat berupa nilai, tabel, gambar, atau berkas; hasil validasi menyatakan apakah syarat minimum terpenuhi; interpretasi menjelaskan makna hasil tanpa melampaui metode yang digunakan.

Kode Siap Google Colab

```
import pandas as pd
from pathlib import Path

df = pd.DataFrame({"parameter":["charge","spin"], "nilai":[0,0]})
df.to_csv("parameter.csv", index=False)
df.to_excel("parameter.xlsx", index=False)
for nama in ["parameter.csv","parameter.xlsx"]:
    path = Path(nama)
    print(nama, path.exists(), path.stat().st_size)
```

Cara Membaca dan Memvalidasi Hasil

Setelah sel dijalankan, baca pesan dari atas ke bawah dan pastikan tidak ada pengecualian yang terlewat. Verifikasi bahwa keluaran sesuai dengan dua format tabel yang dapat dibuka kembali. Untuk keluaran numerik, periksa bilangan hingga, satuan, tanda, skala, dan batas kewajaran. Untuk berkas, periksa keberadaan, ukuran, encoding, dan kemampuan untuk dibuka kembali. Untuk grafik atau visualisasi, pastikan label sumbu, satuan, legenda, dan judul menjelaskan konteks data.

Risiko utama pada resep ini adalah File Excel memerlukan engine yang sesuai; instalasi paket dapat diperlukan pada runtime baru. Oleh karena itu, status keberhasilan tidak boleh disimpulkan hanya dari munculnya output. Jalankan kembali notebook dari sesi bersih, bandingkan hasil dengan perhitungan sederhana atau data acuan ketika tersedia, dan catat versi lingkungan. Bila hasil berubah, identifikasi apakah perubahan berasal dari input, urutan eksekusi, versi pustaka, perangkat, atau algoritma.

Pengembangan dan Latihan

899. Baca kembali kedua berkas dan bandingkan isi tabel.
900. Tambahkan satu pemeriksaan yang sengaja menghentikan proses ketika input tidak memenuhi syarat, lalu uji pesan kesalahannya.
901. Simpan parameter, status, waktu eksekusi, dan hasil utama ke JSON atau CSV; kemudian baca kembali berkas tersebut dalam runtime baru.
902. Tuliskan dua paragraf interpretasi: paragraf pertama hanya memuat fakta keluaran program, sedangkan paragraf kedua memuat makna dan keterbatasannya.

E.4 Visualisasi dan Pelaporan

E.35 Membuat Grafik Garis yang Terukur

Resep ini membahas figure, plot, label, grid, `tight_layout`, dan ekspor. Tujuannya bukan sekadar memperoleh keluaran yang tampak benar, tetapi membangun alur notebook yang dapat dijalankan ulang dan diperiksa. Pembaca perlu menyatakan input secara eksplisit, memisahkan konfigurasi dari proses utama, mencetak status penting, dan menyimpan artefak yang diperlukan. Hasil utama yang ditargetkan adalah grafik garis dengan resolusi publikasi dasar. Setiap angka harus disertai satuan dan konteks, sedangkan setiap berkas perlu diberi nama yang menjelaskan molekul, metode, atau tahap proses.

Tujuan, Input, dan Keluaran

Input resep meliputi parameter yang relevan dengan figure, plot, label, grid, `tight_layout`, dan ekspor. Sebelum eksekusi, periksa tipe data, bentuk array, jalur berkas, dan ketersediaan pustaka. Keluaran harus dibedakan menjadi keluaran langsung program, hasil validasi, dan interpretasi. Keluaran langsung dapat berupa nilai, tabel, gambar, atau berkas; hasil validasi menyatakan apakah syarat minimum terpenuhi; interpretasi menjelaskan makna hasil tanpa melampaui metode yang digunakan.

Kode Siap Google Colab

```
import numpy as np
import matplotlib.pyplot as plt

x = np.linspace(0, 2*np.pi, 200)
y = np.sin(x)
plt.figure(figsize=(7,4.5))
plt.plot(x, y, linewidth=1.8)
plt.xlabel("x (rad)")
plt.ylabel("sin(x)")
plt.title("Fungsi Sinus")
plt.grid(True, alpha=0.3)
plt.tight_layout()
plt.savefig("sinus.png", dpi=200)
plt.show()
```

Cara Membaca dan Memvalidasi Hasil

Setelah sel dijalankan, baca pesan dari atas ke bawah dan pastikan tidak ada pengecualian yang terlewat. Verifikasi bahwa keluaran sesuai dengan grafik garis dengan resolusi publikasi dasar. Untuk keluaran numerik, periksa bilangan hingga, satuan, tanda, skala, dan batas kewajaran. Untuk berkas, periksa keberadaan, ukuran, encoding, dan kemampuan untuk dibuka kembali. Untuk grafik atau visualisasi, pastikan label sumbu, satuan, legenda, dan judul menjelaskan konteks data.

Risiko utama pada resep ini adalah Ukuran gambar di layar tidak sama dengan resolusi file; figsize dan dpi harus dicatat. Oleh karena itu, status keberhasilan tidak boleh disimpulkan hanya dari munculnya output. Jalankan kembali notebook dari sesi bersih, bandingkan hasil dengan perhitungan sederhana atau data acuan ketika tersedia, dan catat versi lingkungan. Bila hasil berubah, identifikasi apakah perubahan berasal dari input, urutan eksekusi, versi pustaka, perangkat, atau algoritma.

Pengembangan dan Latihan

903. Simpan versi PDF dan bandingkan sifat format vektor dengan PNG.
904. Tambahkan satu pemeriksaan yang sengaja menghentikan proses ketika input tidak memenuhi syarat, lalu uji pesan kesalahannya.
905. Simpan parameter, status, waktu eksekusi, dan hasil utama ke JSON atau CSV; kemudian baca kembali berkas tersebut dalam runtime baru.
906. Tuliskan dua paragraf interpretasi: paragraf pertama hanya memuat fakta keluaran program, sedangkan paragraf kedua memuat makna dan keterbatasannya.

E.36 Membuat Diagram Pencar dan Regresi Linear

Resep ini membahas scatter, polyfit, garis tren, dan koefisien determinasi. Tujuannya bukan sekadar memperoleh keluaran yang tampak benar, tetapi membangun alur notebook yang dapat dijalankan ulang dan diperiksa. Pembaca perlu menyatakan input secara eksplisit, memisahkan konfigurasi dari proses utama, mencetak status penting, dan menyimpan artefak yang diperlukan. Hasil utama yang ditargetkan adalah grafik hubungan dua variabel dengan ukuran kecocokan. Setiap angka harus disertai satuan dan konteks, sedangkan setiap berkas perlu diberi nama yang menjelaskan molekul, metode, atau tahap proses.

Tujuan, Input, dan Keluaran

Input resep meliputi parameter yang relevan dengan scatter, polyfit, garis tren, dan koefisien determinasi. Sebelum eksekusi, periksa tipe data, bentuk array, jalur berkas, dan ketersediaan pustaka. Keluaran harus dibedakan menjadi keluaran langsung program, hasil validasi, dan interpretasi. Keluaran langsung dapat berupa nilai, tabel, gambar, atau berkas; hasil validasi menyatakan apakah syarat minimum terpenuhi; interpretasi menjelaskan makna hasil tanpa melampaui metode yang digunakan.

Kode Siap Google Colab

```
import numpy as np
import matplotlib.pyplot as plt

x = np.array([1,2,3,4,5], float)
y = np.array([2.1,4.0,6.2,7.9,10.1], float)
a, b = np.polyfit(x, y, 1)
yhat = a*x+b
r2 = 1 - np.sum((y-yhat)**2)/np.sum((y-y.mean())**2)
plt.scatter(x,y)
plt.plot(x,yhat)
plt.xlabel("x")
plt.ylabel("y")
plt.title(f"Regresi linear, R²={r2:.4f}")
plt.tight_layout(); plt.show()
```

Cara Membaca dan Memvalidasi Hasil

Setelah sel dijalankan, baca pesan dari atas ke bawah dan pastikan tidak ada pengecualian yang terlewat. Verifikasi bahwa keluaran sesuai dengan grafik hubungan dua variabel dengan ukuran kecocokan. Untuk keluaran numerik, periksa bilangan hingga, satuan, tanda, skala, dan batas kewajaran. Untuk berkas, periksa keberadaan, ukuran, encoding, dan kemampuan untuk dibuka kembali. Untuk grafik atau visualisasi, pastikan label sumbu, satuan, legenda, dan judul menjelaskan konteks data.

Risiko utama pada resep ini adalah R kuadrat tinggi tidak membuktikan hubungan kausal atau validitas model di luar rentang data. Oleh karena itu, status keberhasilan tidak boleh disimpulkan hanya dari munculnya output. Jalankan kembali notebook dari sesi bersih, bandingkan hasil dengan perhitungan sederhana atau data acuan ketika tersedia, dan catat versi lingkungan. Bila hasil berubah, identifikasi apakah perubahan berasal dari input, urutan eksekusi, versi pustaka, perangkat, atau algoritma.

Pengembangan dan Latihan

907. Tambahkan residual plot dan periksa pola sistematis.
908. Tambahkan satu pemeriksaan yang sengaja menghentikan proses ketika input tidak memenuhi syarat, lalu uji pesan kesalahannya.
909. Simpan parameter, status, waktu eksekusi, dan hasil utama ke JSON atau CSV; kemudian baca kembali berkas tersebut dalam runtime baru.
910. Tuliskan dua paragraf interpretasi: paragraf pertama hanya memuat fakta keluaran program, sedangkan paragraf kedua memuat makna dan keterbatasannya.

E.37 Membuat Histogram dan Membaca Distribusi

Resep ini membahas histogram, bins, mean, median, dan visualisasi distribusi. Tujuannya bukan sekadar memperoleh keluaran yang tampak benar, tetapi membangun alur notebook yang dapat dijalankan ulang dan diperiksa. Pembaca perlu menyatakan input

secara eksplisit, memisahkan konfigurasi dari proses utama, mencetak status penting, dan menyimpan artefak yang diperlukan. Hasil utama yang ditargetkan adalah histogram waktu eksekusi. Setiap angka harus disertai satuan dan konteks, sedangkan setiap berkas perlu diberi nama yang menjelaskan molekul, metode, atau tahap proses.

Tujuan, Input, dan Keluaran

Input resep meliputi parameter yang relevan dengan histogram, bins, mean, median, dan visualisasi distribusi. Sebelum eksekusi, periksa tipe data, bentuk array, jalur berkas, dan ketersediaan pustaka. Keluaran harus dibedakan menjadi keluaran langsung program, hasil validasi, dan interpretasi. Keluaran langsung dapat berupa nilai, tabel, gambar, atau berkas; hasil validasi menyatakan apakah syarat minimum terpenuhi; interpretasi menjelaskan makna hasil tanpa melampaui metode yang digunakan.

Kode Siap Google Colab

```
import numpy as np
import matplotlib.pyplot as plt

rng = np.random.default_rng(2026)
waktu = rng.lognormal(mean=1.0, sigma=0.25, size=300)
plt.hist(waktu, bins=20, edgecolor="black")
plt.axvline(waktu.mean(), linestyle="--", label="rata-rata")
plt.axvline(np.median(waktu), linestyle=":", label="median")
plt.xlabel("Waktu (s)"); plt.ylabel("Frekuensi")
plt.legend(); plt.tight_layout(); plt.show()
```

Cara Membaca dan Memvalidasi Hasil

Setelah sel dijalankan, baca pesan dari atas ke bawah dan pastikan tidak ada pengecualian yang terlewat. Verifikasi bahwa keluaran sesuai dengan histogram waktu eksekusi. Untuk keluaran numerik, periksa bilangan hingga, satuan, tanda, skala, dan batas kewajaran. Untuk berkas, periksa keberadaan, ukuran, encoding, dan kemampuan untuk dibuka kembali. Untuk grafik atau visualisasi, pastikan label sumbu, satuan, legenda, dan judul menjelaskan konteks data.

Risiko utama pada resep ini adalah Pilihan jumlah bin dapat mengubah kesan visual distribusi. Oleh karena itu, status keberhasilan tidak boleh disimpulkan hanya dari munculnya output. Jalankan kembali notebook dari sesi bersih, bandingkan hasil dengan perhitungan sederhana atau data acuan ketika tersedia, dan catat versi lingkungan. Bila hasil berubah, identifikasi apakah perubahan berasal dari input, urutan eksekusi, versi pustaka, perangkat, atau algoritma.

Pengembangan dan Latihan

911. Bandingkan 10, 20, dan 40 bin serta jelaskan perbedaannya.
912. Tambahkan satu pemeriksaan yang sengaja menghentikan proses ketika input tidak memenuhi syarat, lalu uji pesan kesalahannya.

913. Simpan parameter, status, waktu eksekusi, dan hasil utama ke JSON atau CSV; kemudian baca kembali berkas tersebut dalam runtime baru.
914. Tuliskan dua paragraf interpretasi: paragraf pertama hanya memuat fakta keluaran program, sedangkan paragraf kedua memuat makna dan keterbatasannya.

E.38 Membuat Heatmap Matriks dengan Matplotlib

Resep ini membahas imshow, colorbar, tick label, dan skala data. Tujuannya bukan sekadar memperoleh keluaran yang tampak benar, tetapi membangun alur notebook yang dapat dijalankan ulang dan diperiksa. Pembaca perlu menyatakan input secara eksplisit, memisahkan konfigurasi dari proses utama, mencetak status penting, dan menyimpan artefak yang diperlukan. Hasil utama yang ditargetkan adalah visualisasi matriks numerik dengan legenda. Setiap angka harus disertai satuan dan konteks, sedangkan setiap berkas perlu diberi nama yang menjelaskan molekul, metode, atau tahap proses.

Tujuan, Input, dan Keluaran

Input resep meliputi parameter yang relevan dengan imshow, colorbar, tick label, dan skala data. Sebelum eksekusi, periksa tipe data, bentuk array, jalur berkas, dan ketersediaan pustaka. Keluaran harus dibedakan menjadi keluaran langsung program, hasil validasi, dan interpretasi. Keluaran langsung dapat berupa nilai, tabel, gambar, atau berkas; hasil validasi menyatakan apakah syarat minimum terpenuhi; interpretasi menjelaskan makna hasil tanpa melampaui metode yang digunakan.

Kode Siap Google Colab

```
import numpy as np
import matplotlib.pyplot as plt

M = np.array([[1.0,0.3,0.1],[0.3,1.0,0.5],[0.1,0.5,1.0]])
fig, ax = plt.subplots(figsize=(5,4))
im = ax.imshow(M, vmin=-1, vmax=1)
fig.colorbar(im, ax=ax, label="Nilai")
ax.set_xticks(range(3), ["A","B","C"])
ax.set_yticks(range(3), ["A","B","C"])
plt.tight_layout(); plt.show()
```

Cara Membaca dan Memvalidasi Hasil

Setelah sel dijalankan, baca pesan dari atas ke bawah dan pastikan tidak ada pengecualian yang terlewat. Verifikasi bahwa keluaran sesuai dengan visualisasi matriks numerik dengan legenda. Untuk keluaran numerik, periksa bilangan hingga, satuan, tanda, skala, dan batas kewajaran. Untuk berkas, periksa keberadaan, ukuran, encoding, dan kemampuan untuk dibuka kembali. Untuk grafik atau visualisasi, pastikan label sumbu, satuan, legenda, dan judul menjelaskan konteks data.

Risiko utama pada resep ini adalah Palet warna dan rentang skala memengaruhi interpretasi; nilai numerik tetap perlu tersedia. Oleh karena itu, status keberhasilan tidak

boleh disimpulkan hanya dari munculnya output. Jalankan kembali notebook dari sesi bersih, bandingkan hasil dengan perhitungan sederhana atau data acuan ketika tersedia, dan catat versi lingkungan. Bila hasil berubah, identifikasi apakah perubahan berasal dari input, urutan eksekusi, versi pustaka, perangkat, atau algoritma.

Pengembangan dan Latihan

915. Tambahkan anotasi angka pada setiap sel.
916. Tambahkan satu pemeriksaan yang sengaja menghentikan proses ketika input tidak memenuhi syarat, lalu uji pesan kesalahannya.
917. Simpan parameter, status, waktu eksekusi, dan hasil utama ke JSON atau CSV; kemudian baca kembali berkas tersebut dalam runtime baru.
918. Tuliskan dua paragraf interpretasi: paragraf pertama hanya memuat fakta keluaran program, sedangkan paragraf kedua memuat makna dan keterbatasannya.

E.39 Menyusun Ringkasan Hasil dalam Markdown

Resep ini membahas format Markdown, tabel, persamaan, dan pemisahan fakta dari interpretasi. Tujuannya bukan sekadar memperoleh keluaran yang tampak benar, tetapi membangun alur notebook yang dapat dijalankan ulang dan diperiksa. Pembaca perlu menyatakan input secara eksplisit, memisahkan konfigurasi dari proses utama, mencetak status penting, dan menyimpan artefak yang diperlukan. Hasil utama yang ditargetkan adalah teks laporan yang siap ditempatkan pada sel Markdown. Setiap angka harus disertai satuan dan konteks, sedangkan setiap berkas perlu diberi nama yang menjelaskan molekul, metode, atau tahap proses.

Tujuan, Input, dan Keluaran

Input resep meliputi parameter yang relevan dengan format Markdown, tabel, persamaan, dan pemisahan fakta dari interpretasi. Sebelum eksekusi, periksa tipe data, bentuk array, jalur berkas, dan ketersediaan pustaka. Keluaran harus dibedakan menjadi keluaran langsung program, hasil validasi, dan interpretasi. Keluaran langsung dapat berupa nilai, tabel, gambar, atau berkas; hasil validasi menyatakan apakah syarat minimum terpenuhi; interpretasi menjelaskan makna hasil tanpa melampaui metode yang digunakan.

Kode Siap Google Colab

```
metode = "B3LYP/def2-SVP"
energi = -76.368123
konvergen = True
ringkasan = f""
## Ringkasan Perhitungan
- Metode: **{metode}**
- Energi total: **{energi:.8f} Hartree**
- Konvergen: **{konvergen}**
```

Interpretasi harus mempertimbangkan metode, basis set, dan status optimasi.

```
""  
print(ringkasan)
```

Cara Membaca dan Memvalidasi Hasil

Setelah sel dijalankan, baca pesan dari atas ke bawah dan pastikan tidak ada pengecualian yang terlewat. Verifikasi bahwa keluaran sesuai dengan teks laporan yang siap ditempatkan pada sel Markdown. Untuk keluaran numerik, periksa bilangan hingga, satuan, tanda, skala, dan batas kewajaran. Untuk berkas, periksa keberadaan, ukuran, encoding, dan kemampuan untuk dibuka kembali. Untuk grafik atau visualisasi, pastikan label sumbu, satuan, legenda, dan judul menjelaskan konteks data.

Risiko utama pada resep ini adalah Markdown yang rapi tidak menjamin hasil ilmiah sah; validasi tetap harus dilakukan pada kode dan data. Oleh karena itu, status keberhasilan tidak boleh disimpulkan hanya dari munculnya output. Jalankan kembali notebook dari sesi bersih, bandingkan hasil dengan perhitungan sederhana atau data acuan ketika tersedia, dan catat versi lingkungan. Bila hasil berubah, identifikasi apakah perubahan berasal dari input, urutan eksekusi, versi pustaka, perangkat, atau algoritma.

Pengembangan dan Latihan

919. Tambahkan tabel parameter dan tautan relatif ke gambar hasil.
920. Tambahkan satu pemeriksaan yang sengaja menghentikan proses ketika input tidak memenuhi syarat, lalu uji pesan kesalahannya.
921. Simpan parameter, status, waktu eksekusi, dan hasil utama ke JSON atau CSV; kemudian baca kembali berkas tersebut dalam runtime baru.
922. Tuliskan dua paragraf interpretasi: paragraf pertama hanya memuat fakta keluaran program, sedangkan paragraf kedua memuat makna dan keterbatasannya.

E.40 Membuat Laporan HTML Sederhana

Resep ini membahas HTML, escaping, tabel, dan penyimpanan berkas. Tujuannya bukan sekadar memperoleh keluaran yang tampak benar, tetapi membangun alur notebook yang dapat dijalankan ulang dan diperiksa. Pembaca perlu menyatakan input secara eksplisit, memisahkan konfigurasi dari proses utama, mencetak status penting, dan menyimpan artefak yang diperlukan. Hasil utama yang ditargetkan adalah laporan HTML mandiri yang dapat dibuka di browser. Setiap angka harus disertai satuan dan konteks, sedangkan setiap berkas perlu diberi nama yang menjelaskan molekul, metode, atau tahap proses.

Tujuan, Input, dan Keluaran

Input resep meliputi parameter yang relevan dengan HTML, escaping, tabel, dan penyimpanan berkas. Sebelum eksekusi, periksa tipe data, bentuk array, jalur berkas, dan ketersediaan pustaka. Keluaran harus dibedakan menjadi keluaran langsung program, hasil validasi, dan interpretasi. Keluaran langsung dapat berupa nilai, tabel, gambar, atau

berkas; hasil validasi menyatakan apakah syarat minimum terpenuhi; interpretasi menjelaskan makna hasil tanpa melampaui metode yang digunakan.

Kode Siap Google Colab

```
from pathlib import Path
from html import escape

judul = "Laporan Notebook"
isi = "Perhitungan selesai dan berkas hasil telah divalidasi."
html = f"""<!doctype html><html><meta charset='utf-8'>
<title>{escape(judul)}</title><body><h1>{escape(judul)}</h1>
<p>{escape(isi)}</p></body></html>"""
Path("laporan.html").write_text(html, encoding="utf-8")
print("laporan.html dibuat")
```

Cara Membaca dan Memvalidasi Hasil

Setelah sel dijalankan, baca pesan dari atas ke bawah dan pastikan tidak ada pengecualian yang terlewat. Verifikasi bahwa keluaran sesuai dengan laporan HTML mandiri yang dapat dibuka di browser. Untuk keluaran numerik, periksa bilangan hingga, satuan, tanda, skala, dan batas kewajaran. Untuk berkas, periksa keberadaan, ukuran, encoding, dan kemampuan untuk dibuka kembali. Untuk grafik atau visualisasi, pastikan label sumbu, satuan, legenda, dan judul menjelaskan konteks data.

Risiko utama pada resep ini adalah Teks pengguna harus di-escape agar tidak dianggap sebagai elemen HTML. Oleh karena itu, status keberhasilan tidak boleh disimpulkan hanya dari munculnya output. Jalankan kembali notebook dari sesi bersih, bandingkan hasil dengan perhitungan sederhana atau data acuan ketika tersedia, dan catat versi lingkungan. Bila hasil berubah, identifikasi apakah perubahan berasal dari input, urutan eksekusi, versi pustaka, perangkat, atau algoritma.

Pengembangan dan Latihan

923. Tambahkan tabel parameter dan gambar lokal dengan alt text.
924. Tambahkan satu pemeriksaan yang sengaja menghentikan proses ketika input tidak memenuhi syarat, lalu uji pesan kesalahannya.
925. Simpan parameter, status, waktu eksekusi, dan hasil utama ke JSON atau CSV; kemudian baca kembali berkas tersebut dalam runtime baru.
926. Tuliskan dua paragraf interpretasi: paragraf pertama hanya memuat fakta keluaran program, sedangkan paragraf kedua memuat makna dan keterbatasannya.

E.41 Membuat Arsip ZIP Hasil Proyek

Resep ini membahas zipfile, pola berkas, dan daftar isi arsip. Tujuannya bukan sekadar memperoleh keluaran yang tampak benar, tetapi membangun alur notebook yang dapat dijalankan ulang dan diperiksa. Pembaca perlu menyatakan input secara eksplisit, memisahkan konfigurasi dari proses utama, mencetak status penting, dan menyimpan

artefak yang diperlukan. Hasil utama yang ditargetkan adalah arsip ZIP yang memuat data, gambar, dan log. Setiap angka harus disertai satuan dan konteks, sedangkan setiap berkas perlu diberi nama yang menjelaskan molekul, metode, atau tahap proses.

Tujuan, Input, dan Keluaran

Input resep meliputi parameter yang relevan dengan zipfile, pola berkas, dan daftar isi arsip. Sebelum eksekusi, periksa tipe data, bentuk array, jalur berkas, dan ketersediaan pustaka. Keluaran harus dibedakan menjadi keluaran langsung program, hasil validasi, dan interpretasi. Keluaran langsung dapat berupa nilai, tabel, gambar, atau berkas; hasil validasi menyatakan apakah syarat minimum terpenuhi; interpretasi menjelaskan makna hasil tanpa melampaui metode yang digunakan.

Kode Siap Google Colab

```
from pathlib import Path
from zipfile import ZipFile, ZIP_DEFLATED

files = [p for p in Path('.').glob('*') if p.suffix in
{' .csv', '.json', '.png', '.log'}]
with ZipFile('hasil_proyek.zip', 'w', ZIP_DEFLATED) as z:
    for p in files:
        z.write(p, arcname=p.name)
print("Isi arsip:")
with ZipFile('hasil_proyek.zip') as z:
    print('
'.join(z.namelist()))
```

Cara Membaca dan Memvalidasi Hasil

Setelah sel dijalankan, baca pesan dari atas ke bawah dan pastikan tidak ada pengecualian yang terlewat. Verifikasi bahwa keluaran sesuai dengan arsip ZIP yang memuat data, gambar, dan log. Untuk keluaran numerik, periksa bilangan hingga, satuan, tanda, skala, dan batas kewajaran. Untuk berkas, periksa keberadaan, ukuran, encoding, dan kemampuan untuk dibuka kembali. Untuk grafik atau visualisasi, pastikan label sumbu, satuan, legenda, dan judul menjelaskan konteks data.

Risiko utama pada resep ini adalah Memasukkan berkas sensitif atau berukuran sangat besar tanpa pemeriksaan dapat menjadi masalah. Oleh karena itu, status keberhasilan tidak boleh disimpulkan hanya dari munculnya output. Jalankan kembali notebook dari sesi bersih, bandingkan hasil dengan perhitungan sederhana atau data acuan ketika tersedia, dan catat versi lingkungan. Bila hasil berubah, identifikasi apakah perubahan berasal dari input, urutan eksekusi, versi pustaka, perangkat, atau algoritma.

Pengembangan dan Latihan

927. Tambahkan manifest JSON berisi ukuran dan checksum setiap berkas.

928. Tambahkan satu pemeriksaan yang sengaja menghentikan proses ketika input tidak memenuhi syarat, lalu uji pesan kesalahannya.
929. Simpan parameter, status, waktu eksekusi, dan hasil utama ke JSON atau CSV; kemudian baca kembali berkas tersebut dalam runtime baru.
930. Tuliskan dua paragraf interpretasi: paragraf pertama hanya memuat fakta keluaran program, sedangkan paragraf kedua memuat makna dan keterbatasannya.

E.42 Mengunduh Berkas dari Google Colab

Resep ini membahas `google.colab.files`, pemeriksaan keberadaan, dan unduhan. Tujuannya bukan sekadar memperoleh keluaran yang tampak benar, tetapi membangun alur notebook yang dapat dijalankan ulang dan diperiksa. Pembaca perlu menyatakan input secara eksplisit, memisahkan konfigurasi dari proses utama, mencetak status penting, dan menyimpan artefak yang diperlukan. Hasil utama yang ditargetkan adalah berkas hasil yang dipindahkan ke komputer lokal. Setiap angka harus disertai satuan dan konteks, sedangkan setiap berkas perlu diberi nama yang menjelaskan molekul, metode, atau tahap proses.

Tujuan, Input, dan Keluaran

Input resep meliputi parameter yang relevan dengan `google.colab.files`, pemeriksaan keberadaan, dan unduhan. Sebelum eksekusi, periksa tipe data, bentuk array, jalur berkas, dan ketersediaan pustaka. Keluaran harus dibedakan menjadi keluaran langsung program, hasil validasi, dan interpretasi. Keluaran langsung dapat berupa nilai, tabel, gambar, atau berkas; hasil validasi menyatakan apakah syarat minimum terpenuhi; interpretasi menjelaskan makna hasil tanpa melampaui metode yang digunakan.

Kode Siap Google Colab

```
from pathlib import Path
from google.colab import files

path = Path("hasil_proyek.zip")
if not path.exists():
    raise FileNotFoundError("hasil_proyek.zip belum dibuat")
print("Ukuran:", path.stat().st_size, "byte")
files.download(str(path))
```

Cara Membaca dan Memvalidasi Hasil

Setelah sel dijalankan, baca pesan dari atas ke bawah dan pastikan tidak ada pengecualian yang terlewat. Verifikasi bahwa keluaran sesuai dengan berkas hasil yang dipindahkan ke komputer lokal. Untuk keluaran numerik, periksa bilangan hingga, satuan, tanda, skala, dan batas kewajaran. Untuk berkas, periksa keberadaan, ukuran, encoding, dan kemampuan untuk dibuka kembali. Untuk grafik atau visualisasi, pastikan label sumbu, satuan, legenda, dan judul menjelaskan konteks data.

Risiko utama pada resep ini adalah Perintah `google.colab` hanya tersedia di Colab dan akan gagal di Jupyter lokal. Oleh karena itu, status keberhasilan tidak boleh disimpulkan hanya dari munculnya output. Jalankan kembali notebook dari sesi bersih, bandingkan hasil dengan perhitungan sederhana atau data acuan ketika tersedia, dan catat versi lingkungan. Bila hasil berubah, identifikasi apakah perubahan berasal dari input, urutan eksekusi, versi pustaka, perangkat, atau algoritma.

Pengembangan dan Latihan

931. Tambahkan deteksi lingkungan agar notebook memberi instruksi alternatif ketika tidak berada di Colab.
932. Tambahkan satu pemeriksaan yang sengaja menghentikan proses ketika input tidak memenuhi syarat, lalu uji pesan kesalahannya.
933. Simpan parameter, status, waktu eksekusi, dan hasil utama ke JSON atau CSV; kemudian baca kembali berkas tersebut dalam runtime baru.
934. Tuliskan dua paragraf interpretasi: paragraf pertama hanya memuat fakta keluaran program, sedangkan paragraf kedua memuat makna dan keterbatasannya.

E.43 Mengunggah Berkas ke Runtime

Resep ini membahas `files.upload`, daftar nama, dan validasi ekstensi. Tujuannya bukan sekadar memperoleh keluaran yang tampak benar, tetapi membangun alur notebook yang dapat dijalankan ulang dan diperiksa. Pembaca perlu menyatakan input secara eksplisit, memisahkan konfigurasi dari proses utama, mencetak status penting, dan menyimpan artefak yang diperlukan. Hasil utama yang ditargetkan adalah berkas input yang tersedia di `/content`. Setiap angka harus disertai satuan dan konteks, sedangkan setiap berkas perlu diberi nama yang menjelaskan molekul, metode, atau tahap proses.

Tujuan, Input, dan Keluaran

Input resep meliputi parameter yang relevan dengan `files.upload`, daftar nama, dan validasi ekstensi. Sebelum eksekusi, periksa tipe data, bentuk array, jalur berkas, dan ketersediaan pustaka. Keluaran harus dibedakan menjadi keluaran langsung program, hasil validasi, dan interpretasi. Keluaran langsung dapat berupa nilai, tabel, gambar, atau berkas; hasil validasi menyatakan apakah syarat minimum terpenuhi; interpretasi menjelaskan makna hasil tanpa melampaui metode yang digunakan.

Kode Siap Google Colab

```
from google.colab import files
from pathlib import Path

uploaded = files.upload()
for nama, data in uploaded.items():
    path = Path(nama)
    print(nama, len(data), "byte", path.suffix.lower())
```


Cara Membaca dan Memvalidasi Hasil

Setelah sel dijalankan, baca pesan dari atas ke bawah dan pastikan tidak ada pengecualian yang terlewat. Verifikasi bahwa keluaran sesuai dengan berkas input yang tersedia di /content. Untuk keluaran numerik, periksa bilangan hingga, satuan, tanda, skala, dan batas kewajaran. Untuk berkas, periksa keberadaan, ukuran, encoding, dan kemampuan untuk dibuka kembali. Untuk grafik atau visualisasi, pastikan label sumbu, satuan, legenda, dan judul menjelaskan konteks data.

Risiko utama pada resep ini adalah Nama berkas dari pengguna perlu diperiksa sebelum dipakai sebagai jalur keluaran atau perintah. Oleh karena itu, status keberhasilan tidak boleh disimpulkan hanya dari munculnya output. Jalankan kembali notebook dari sesi bersih, bandingkan hasil dengan perhitungan sederhana atau data acuan ketika tersedia, dan catat versi lingkungan. Bila hasil berubah, identifikasi apakah perubahan berasal dari input, urutan eksekusi, versi pustaka, perangkat, atau algoritma.

Pengembangan dan Latihan

935. Batasi ekstensi pada CSV, XYZ, dan JSON serta tolak berkas kosong.
936. Tambahkan satu pemeriksaan yang sengaja menghentikan proses ketika input tidak memenuhi syarat, lalu uji pesan kesalahannya.
937. Simpan parameter, status, waktu eksekusi, dan hasil utama ke JSON atau CSV; kemudian baca kembali berkas tersebut dalam runtime baru.
938. Tuliskan dua paragraf interpretasi: paragraf pertama hanya memuat fakta keluaran program, sedangkan paragraf kedua memuat makna dan keterbatasannya.

E.44 Menghubungkan Google Drive

Resep ini membahas `drive.mount`, jalur MyDrive, dan pembuatan folder proyek. Tujuannya bukan sekadar memperoleh keluaran yang tampak benar, tetapi membangun alur notebook yang dapat dijalankan ulang dan diperiksa. Pembaca perlu menyatakan input secara eksplisit, memisahkan konfigurasi dari proses utama, mencetak status penting, dan menyimpan artefak yang diperlukan. Hasil utama yang ditargetkan adalah folder persisten yang dapat dipakai lintas sesi. Setiap angka harus disertai satuan dan konteks, sedangkan setiap berkas perlu diberi nama yang menjelaskan molekul, metode, atau tahap proses.

Tujuan, Input, dan Keluaran

Input resep meliputi parameter yang relevan dengan `drive.mount`, jalur MyDrive, dan pembuatan folder proyek. Sebelum eksekusi, periksa tipe data, bentuk array, jalur berkas, dan ketersediaan pustaka. Keluaran harus dibedakan menjadi keluaran langsung program, hasil validasi, dan interpretasi. Keluaran langsung dapat berupa nilai, tabel, gambar, atau berkas; hasil validasi menyatakan apakah syarat minimum terpenuhi; interpretasi menjelaskan makna hasil tanpa melampaui metode yang digunakan.

Kode Siap Google Colab

```
from google.colab import drive
from pathlib import Path

drive.mount('/content/drive')
folder = Path('/content/drive/MyDrive/Proyek_Colab')
folder.mkdir(parents=True, exist_ok=True)
print("Folder proyek:", folder)
```

Cara Membaca dan Memvalidasi Hasil

Setelah sel dijalankan, baca pesan dari atas ke bawah dan pastikan tidak ada pengecualian yang terlewat. Verifikasi bahwa keluaran sesuai dengan folder persisten yang dapat dipakai lintas sesi. Untuk keluaran numerik, periksa bilangan hingga, satuan, tanda, skala, dan batas kewajaran. Untuk berkas, periksa keberadaan, ukuran, encoding, dan kemampuan untuk dibuka kembali. Untuk grafik atau visualisasi, pastikan label sumbu, satuan, legenda, dan judul menjelaskan konteks data.

Risiko utama pada resep ini adalah Mount memberi runtime akses ke Drive; jalankan hanya notebook yang dipercaya. Oleh karena itu, status keberhasilan tidak boleh disimpulkan hanya dari munculnya output. Jalankan kembali notebook dari sesi bersih, bandingkan hasil dengan perhitungan sederhana atau data acuan ketika tersedia, dan catat versi lingkungan. Bila hasil berubah, identifikasi apakah perubahan berasal dari input, urutan eksekusi, versi pustaka, perangkat, atau algoritma.

Pengembangan dan Latihan

939. Buat subfolder input, output, figures, dan logs serta uji tulis satu berkas kecil.
940. Tambahkan satu pemeriksaan yang sengaja menghentikan proses ketika input tidak memenuhi syarat, lalu uji pesan kesalahannya.
941. Simpan parameter, status, waktu eksekusi, dan hasil utama ke JSON atau CSV; kemudian baca kembali berkas tersebut dalam runtime baru.
942. Tuliskan dua paragraf interpretasi: paragraf pertama hanya memuat fakta keluaran program, sedangkan paragraf kedua memuat makna dan keterbatasannya.

E.45 Memeriksa Ruang Disk dan Memori

Resep ini membahas `shutil.disk_usage`, `psutil`, dan ambang peringatan. Tujuannya bukan sekadar memperoleh keluaran yang tampak benar, tetapi membangun alur notebook yang dapat dijalankan ulang dan diperiksa. Pembaca perlu menyatakan input secara eksplisit, memisahkan konfigurasi dari proses utama, mencetak status penting, dan menyimpan artefak yang diperlukan. Hasil utama yang ditargetkan adalah ringkasan kapasitas runtime sebelum proses besar. Setiap angka harus disertai satuan dan konteks, sedangkan setiap berkas perlu diberi nama yang menjelaskan molekul, metode, atau tahap proses.

Tujuan, Input, dan Keluaran

Input resep meliputi parameter yang relevan dengan `shutil.disk_usage`, `psutil`, dan ambang peringatan. Sebelum eksekusi, periksa tipe data, bentuk array, jalur berkas, dan ketersediaan pustaka. Keluaran harus dibedakan menjadi keluaran langsung program, hasil validasi, dan interpretasi. Keluaran langsung dapat berupa nilai, tabel, gambar, atau berkas; hasil validasi menyatakan apakah syarat minimum terpenuhi; interpretasi menjelaskan makna hasil tanpa melampaui metode yang digunakan.

Kode Siap Google Colab

```
import shutil
try:
    import psutil
except ImportError:
    %pip -q install psutil
    import psutil

total, used, free = shutil.disk_usage('/content')
ram = psutil.virtual_memory()
print(f"Disk bebas: {free/1024**3:.2f} GB")
print(f"RAM bebas : {ram.available/1024**3:.2f} GB")
```

Cara Membaca dan Memvalidasi Hasil

Setelah sel dijalankan, baca pesan dari atas ke bawah dan pastikan tidak ada pengecualian yang terlewat. Verifikasi bahwa keluaran sesuai dengan ringkasan kapasitas runtime sebelum proses besar. Untuk keluaran numerik, periksa bilangan hingga, satuan, tanda, skala, dan batas kewajaran. Untuk berkas, periksa keberadaan, ukuran, encoding, dan kemampuan untuk dibuka kembali. Untuk grafik atau visualisasi, pastikan label sumbu, satuan, legenda, dan judul menjelaskan konteks data.

Risiko utama pada resep ini adalah Kapasitas yang terlihat dapat berubah selama sesi dan tidak menjamin tugas besar akan selesai. Oleh karena itu, status keberhasilan tidak boleh disimpulkan hanya dari munculnya output. Jalankan kembali notebook dari sesi bersih, bandingkan hasil dengan perhitungan sederhana atau data acuan ketika tersedia, dan catat versi lingkungan. Bila hasil berubah, identifikasi apakah perubahan berasal dari input, urutan eksekusi, versi pustaka, perangkat, atau algoritma.

Pengembangan dan Latihan

943. Tambahkan ambang minimum dan hentikan proses bila ruang bebas terlalu kecil.
944. Tambahkan satu pemeriksaan yang sengaja menghentikan proses ketika input tidak memenuhi syarat, lalu uji pesan kesalahannya.
945. Simpan parameter, status, waktu eksekusi, dan hasil utama ke JSON atau CSV; kemudian baca kembali berkas tersebut dalam runtime baru.
946. Tuliskan dua paragraf interpretasi: paragraf pertama hanya memuat fakta keluaran program, sedangkan paragraf kedua memuat makna dan keterbatasannya.

E.46 Merekam Versi Pustaka ke requirements.txt

Resep ini membahas `importlib.metadata`, daftar paket, dan pencatatan versi. Tujuannya bukan sekadar memperoleh keluaran yang tampak benar, tetapi membangun alur notebook yang dapat dijalankan ulang dan diperiksa. Pembaca perlu menyatakan input secara eksplisit, memisahkan konfigurasi dari proses utama, mencetak status penting, dan menyimpan artefak yang diperlukan. Hasil utama yang ditargetkan adalah berkas versi minimum untuk reproduksibilitas. Setiap angka harus disertai satuan dan konteks, sedangkan setiap berkas perlu diberi nama yang menjelaskan molekul, metode, atau tahap proses.

Tujuan, Input, dan Keluaran

Input resep meliputi parameter yang relevan dengan `importlib.metadata`, daftar paket, dan pencatatan versi. Sebelum eksekusi, periksa tipe data, bentuk array, jalur berkas, dan ketersediaan pustaka. Keluaran harus dibedakan menjadi keluaran langsung program, hasil validasi, dan interpretasi. Keluaran langsung dapat berupa nilai, tabel, gambar, atau berkas; hasil validasi menyatakan apakah syarat minimum terpenuhi; interpretasi menjelaskan makna hasil tanpa melampaui metode yang digunakan.

Kode Siap Google Colab

```
from importlib.metadata import version, PackageNotFoundError
from pathlib import Path

paket = ["numpy", "pandas", "matplotlib", "pyscf", "rdkit", "py3Dmol"]
baris = []
for nama in paket:
    try:
        baris.append(f"{nama}=={version(nama)}")
    except PackageNotFoundError:
        baris.append(f"# {nama} tidak terpasang")
Path("requirements_runtime.txt").write_text(
    ".join(baris)+"
")
print(
    ".join(baris))
```

Cara Membaca dan Memvalidasi Hasil

Setelah sel dijalankan, baca pesan dari atas ke bawah dan pastikan tidak ada pengecualian yang terlewat. Verifikasi bahwa keluaran sesuai dengan berkas versi minimum untuk reproduksibilitas. Untuk keluaran numerik, periksa bilangan hingga, satuan, tanda, skala, dan batas kewajaran. Untuk berkas, periksa keberadaan, ukuran, encoding, dan kemampuan untuk dibuka kembali. Untuk grafik atau visualisasi, pastikan label sumbu, satuan, legenda, dan judul menjelaskan konteks data.

Risiko utama pada resep ini adalah Nama paket distribusi kadang berbeda dari nama modul impor. Oleh karena itu, status keberhasilan tidak boleh disimpulkan hanya dari

munculnya output. Jalankan kembali notebook dari sesi bersih, bandingkan hasil dengan perhitungan sederhana atau data acuan ketika tersedia, dan catat versi lingkungan. Bila hasil berubah, identifikasi apakah perubahan berasal dari input, urutan eksekusi, versi pustaka, perangkat, atau algoritma.

Pengembangan dan Latihan

947. Tambahkan versi Python dan sistem operasi pada berkas metadata terpisah.
948. Tambahkan satu pemeriksaan yang sengaja menghentikan proses ketika input tidak memenuhi syarat, lalu uji pesan kesalahannya.
949. Simpan parameter, status, waktu eksekusi, dan hasil utama ke JSON atau CSV; kemudian baca kembali berkas tersebut dalam runtime baru.
950. Tuliskan dua paragraf interpretasi: paragraf pertama hanya memuat fakta keluaran program, sedangkan paragraf kedua memuat makna dan keterbatasannya.

E.47 Membuat Manifest Hasil dengan Checksum

Resep ini membahas hashlib, ukuran berkas, dan integritas. Tujuannya bukan sekadar memperoleh keluaran yang tampak benar, tetapi membangun alur notebook yang dapat dijalankan ulang dan diperiksa. Pembaca perlu menyatakan input secara eksplisit, memisahkan konfigurasi dari proses utama, mencetak status penting, dan menyimpan artefak yang diperlukan. Hasil utama yang ditargetkan adalah manifest SHA-256 untuk seluruh artefak. Setiap angka harus disertai satuan dan konteks, sedangkan setiap berkas perlu diberi nama yang menjelaskan molekul, metode, atau tahap proses.

Tujuan, Input, dan Keluaran

Input resep meliputi parameter yang relevan dengan hashlib, ukuran berkas, dan integritas. Sebelum eksekusi, periksa tipe data, bentuk array, jalur berkas, dan ketersediaan pustaka. Keluaran harus dibedakan menjadi keluaran langsung program, hasil validasi, dan interpretasi. Keluaran langsung dapat berupa nilai, tabel, gambar, atau berkas; hasil validasi menyatakan apakah syarat minimum terpenuhi; interpretasi menjelaskan makna hasil tanpa melampaui metode yang digunakan.

Kode Siap Google Colab

```
from pathlib import Path
import hashlib, json

def sha256(path):
    h = hashlib.sha256()
    with open(path, 'rb') as f:
        for blok in iter(lambda: f.read(1024*1024), b''):
            h.update(blok)
    return h.hexdigest()

manifest = []
for p in Path('.').glob('*'):
```

```
if p.is_file() and p.name != 'manifest.json':

    manifest.append({"file":p.name,"size":p.stat().st_size,"sha256":sha256(p)})
    Path('manifest.json').write_text(json.dumps(manifest,indent=2))
    print("Jumlah berkas:", len(manifest))
```

Cara Membaca dan Memvalidasi Hasil

Setelah sel dijalankan, baca pesan dari atas ke bawah dan pastikan tidak ada pengecualian yang terlewat. Verifikasi bahwa keluaran sesuai dengan manifest SHA-256 untuk seluruh artefak. Untuk keluaran numerik, periksa bilangan hingga, satuan, tanda, skala, dan batas kewajaran. Untuk berkas, periksa keberadaan, ukuran, encoding, dan kemampuan untuk dibuka kembali. Untuk grafik atau visualisasi, pastikan label sumbu, satuan, legenda, dan judul menjelaskan konteks data.

Risiko utama pada resep ini adalah Checksum memverifikasi integritas berkas, bukan kebenaran ilmiah isinya. Oleh karena itu, status keberhasilan tidak boleh disimpulkan hanya dari munculnya output. Jalankan kembali notebook dari sesi bersih, bandingkan hasil dengan perhitungan sederhana atau data acuan ketika tersedia, dan catat versi lingkungan. Bila hasil berubah, identifikasi apakah perubahan berasal dari input, urutan eksekusi, versi pustaka, perangkat, atau algoritma.

Pengembangan dan Latihan

951. Ubah satu berkas, hitung ulang checksum, dan jelaskan perbedaannya.
952. Tambahkan satu pemeriksaan yang sengaja menghentikan proses ketika input tidak memenuhi syarat, lalu uji pesan kesalahannya.
953. Simpan parameter, status, waktu eksekusi, dan hasil utama ke JSON atau CSV; kemudian baca kembali berkas tersebut dalam runtime baru.
954. Tuliskan dua paragraf interpretasi: paragraf pertama hanya memuat fakta keluaran program, sedangkan paragraf kedua memuat makna dan keterbatasannya.

E.48 Mendeteksi GPU dengan nvidia-smi

Resep ini membahas perintah shell, return code, dan fallback CPU. Tujuannya bukan sekadar memperoleh keluaran yang tampak benar, tetapi membangun alur notebook yang dapat dijalankan ulang dan diperiksa. Pembaca perlu menyatakan input secara eksplisit, memisahkan konfigurasi dari proses utama, mencetak status penting, dan menyimpan artefak yang diperlukan. Hasil utama yang ditargetkan adalah status GPU yang tidak mengasumsikan ketersediaan. Setiap angka harus disertai satuan dan konteks, sedangkan setiap berkas perlu diberi nama yang menjelaskan molekul, metode, atau tahap proses.

Tujuan, Input, dan Keluaran

Input resep meliputi parameter yang relevan dengan perintah shell, return code, dan fallback CPU. Sebelum eksekusi, periksa tipe data, bentuk array, jalur berkas, dan ketersediaan pustaka. Keluaran harus dibedakan menjadi keluaran langsung program,

hasil validasi, dan interpretasi. Keluaran langsung dapat berupa nilai, tabel, gambar, atau berkas; hasil validasi menyatakan apakah syarat minimum terpenuhi; interpretasi menjelaskan makna hasil tanpa melampaui metode yang digunakan.

Kode Siap Google Colab

```
import subprocess

hasil = subprocess.run(
    ["nvidia-smi", "--query-gpu=name,memory.total", "--format=csv,noheader"],
    capture_output=True, text=True
)
if hasil.returncode == 0:
    print("GPU tersedia:", hasil.stdout.strip())
else:
    print("GPU tidak terdeteksi; gunakan mode CPU")
```

Cara Membaca dan Memvalidasi Hasil

Setelah sel dijalankan, baca pesan dari atas ke bawah dan pastikan tidak ada pengecualian yang terlewat. Verifikasi bahwa keluaran sesuai dengan status GPU yang tidak mengasumsikan ketersediaan. Untuk keluaran numerik, periksa bilangan hingga, satuan, tanda, skala, dan batas kewajaran. Untuk berkas, periksa keberadaan, ukuran, encoding, dan kemampuan untuk dibuka kembali. Untuk grafik atau visualisasi, pastikan label sumbu, satuan, legenda, dan judul menjelaskan konteks data.

Risiko utama pada resep ini adalah Runtime GPU aktif tidak berarti setiap pustaka otomatis memakai GPU. Oleh karena itu, status keberhasilan tidak boleh disimpulkan hanya dari munculnya output. Jalankan kembali notebook dari sesi bersih, bandingkan hasil dengan perhitungan sederhana atau data acuan ketika tersedia, dan catat versi lingkungan. Bila hasil berubah, identifikasi apakah perubahan berasal dari input, urutan eksekusi, versi pustaka, perangkat, atau algoritma.

Pengembangan dan Latihan

955. Tambahkan pencatatan CUDA dan uji impor backend GPU yang benar-benar digunakan.
956. Tambahkan satu pemeriksaan yang sengaja menghentikan proses ketika input tidak memenuhi syarat, lalu uji pesan kesalahannya.
957. Simpan parameter, status, waktu eksekusi, dan hasil utama ke JSON atau CSV; kemudian baca kembali berkas tersebut dalam runtime baru.
958. Tuliskan dua paragraf interpretasi: paragraf pertama hanya memuat fakta keluaran program, sedangkan paragraf kedua memuat makna dan keterbatasannya.

E.49 Membuat Fungsi Fallback CPU-GPU

Resep ini membahas deteksi backend, try-except, dan keputusan eksplisit. Tujuannya bukan sekadar memperoleh keluaran yang tampak benar, tetapi membangun alur

notebook yang dapat dijalankan ulang dan diperiksa. Pembaca perlu menyatakan input secara eksplisit, memisahkan konfigurasi dari proses utama, mencetak status penting, dan menyimpan artefak yang diperlukan. Hasil utama yang ditargetkan adalah mode auto yang tetap dapat berjalan pada CPU. Setiap angka harus disertai satuan dan konteks, sedangkan setiap berkas perlu diberi nama yang menjelaskan molekul, metode, atau tahap proses.

Tujuan, Input, dan Keluaran

Input resep meliputi parameter yang relevan dengan deteksi backend, try-except, dan keputusan eksplisit. Sebelum eksekusi, periksa tipe data, bentuk array, jalur berkas, dan ketersediaan pustaka. Keluaran harus dibedakan menjadi keluaran langsung program, hasil validasi, dan interpretasi. Keluaran langsung dapat berupa nilai, tabel, gambar, atau berkas; hasil validasi menyatakan apakah syarat minimum terpenuhi; interpretasi menjelaskan makna hasil tanpa melampaui metode yang digunakan.

Kode Siap Google Colab

```
def pilih_backend(mode="auto"):
    mode = mode.lower()
    if mode not in {"auto", "cpu", "gpu"}:
        raise ValueError("mode harus auto, cpu, atau gpu")
    if mode == "cpu":
        return "cpu"
    try:
        import cupy as cp
        _ = cp.zeros(1)
        return "gpu"
    except Exception as exc:
        if mode == "gpu":
            raise RuntimeError("GPU diminta tetapi tidak siap") from exc
        return "cpu"

print("Backend:", pilih_backend("auto"))
```

Cara Membaca dan Memvalidasi Hasil

Setelah sel dijalankan, baca pesan dari atas ke bawah dan pastikan tidak ada pengecualian yang terlewat. Verifikasi bahwa keluaran sesuai dengan mode auto yang tetap dapat berjalan pada CPU. Untuk keluaran numerik, periksa bilangan hingga, satuan, tanda, skala, dan batas kewajaran. Untuk berkas, periksa keberadaan, ukuran, encoding, dan kemampuan untuk dibuka kembali. Untuk grafik atau visualisasi, pastikan label sumbu, satuan, legenda, dan judul menjelaskan konteks data.

Risiko utama pada resep ini adalah Menangkap semua kesalahan hanya boleh disertai pencatatan sebab agar kegagalan kompatibilitas tidak tersembunyi. Oleh karena itu, status keberhasilan tidak boleh disimpulkan hanya dari munculnya output. Jalankan kembali notebook dari sesi bersih, bandingkan hasil dengan perhitungan sederhana atau data acuan

ketika tersedia, dan catat versi lingkungan. Bila hasil berubah, identifikasi apakah perubahan berasal dari input, urutan eksekusi, versi pustaka, perangkat, atau algoritma.

Pengembangan dan Latihan

959. Tambahkan log alasan fallback dan waktu deteksi backend.
960. Tambahkan satu pemeriksaan yang sengaja menghentikan proses ketika input tidak memenuhi syarat, lalu uji pesan kesalahannya.
961. Simpan parameter, status, waktu eksekusi, dan hasil utama ke JSON atau CSV; kemudian baca kembali berkas tersebut dalam runtime baru.
962. Tuliskan dua paragraf interpretasi: paragraf pertama hanya memuat fakta keluaran program, sedangkan paragraf kedua memuat makna dan keterbatasannya.

E.5 Kimia Komputasi Dasar

E.50 Mengubah SMILES Menjadi Molekul RDKit

Resep ini membahas SMILES, sanitasi, atom hidrogen, dan objek molekul. Tujuannya bukan sekadar memperoleh keluaran yang tampak benar, tetapi membangun alur notebook yang dapat dijalankan ulang dan diperiksa. Pembaca perlu menyatakan input secara eksplisit, memisahkan konfigurasi dari proses utama, mencetak status penting, dan menyimpan artefak yang diperlukan. Hasil utama yang ditargetkan adalah molekul valid dengan jumlah atom eksplisit. Setiap angka harus disertai satuan dan konteks, sedangkan setiap berkas perlu diberi nama yang menjelaskan molekul, metode, atau tahap proses.

Tujuan, Input, dan Keluaran

Input resep meliputi parameter yang relevan dengan SMILES, sanitasi, atom hidrogen, dan objek molekul. Sebelum eksekusi, periksa tipe data, bentuk array, jalur berkas, dan ketersediaan pustaka. Keluaran harus dibedakan menjadi keluaran langsung program, hasil validasi, dan interpretasi. Keluaran langsung dapat berupa nilai, tabel, gambar, atau berkas; hasil validasi menyatakan apakah syarat minimum terpenuhi; interpretasi menjelaskan makna hasil tanpa melampaui metode yang digunakan.

Kode Siap Google Colab

```
%pip -q install rdkit
from rdkit import Chem

smiles = "CCO"
mol = Chem.MolFromSmiles(smiles)
if mol is None:
    raise ValueError("SMILES tidak valid")
mol = Chem.AddHs(mol)
print("Jumlah atom:", mol.GetNumAtoms())
print("Rumus:", Chem.rdMolDescriptors.CalcMolFormula(mol))
```


Cara Membaca dan Memvalidasi Hasil

Setelah sel dijalankan, baca pesan dari atas ke bawah dan pastikan tidak ada pengecualian yang terlewat. Verifikasi bahwa keluaran sesuai dengan molekul valid dengan jumlah atom eksplisit. Untuk keluaran numerik, periksa bilangan hingga, satuan, tanda, skala, dan batas kewajaran. Untuk berkas, periksa keberadaan, ukuran, encoding, dan kemampuan untuk dibuka kembali. Untuk grafik atau visualisasi, pastikan label sumbu, satuan, legenda, dan judul menjelaskan konteks data.

Risiko utama pada resep ini adalah SMILES biasanya tidak memuat koordinat tiga dimensi dan hidrogen dapat bersifat implisit. Oleh karena itu, status keberhasilan tidak boleh disimpulkan hanya dari munculnya output. Jalankan kembali notebook dari sesi bersih, bandingkan hasil dengan perhitungan sederhana atau data acuan ketika tersedia, dan catat versi lingkungan. Bila hasil berubah, identifikasi apakah perubahan berasal dari input, urutan eksekusi, versi pustaka, perangkat, atau algoritma.

Pengembangan dan Latihan

963. Ganti etanol dengan parasetamol dan periksa muatan formal setiap atom.
964. Tambahkan satu pemeriksaan yang sengaja menghentikan proses ketika input tidak memenuhi syarat, lalu uji pesan kesalahannya.
965. Simpan parameter, status, waktu eksekusi, dan hasil utama ke JSON atau CSV; kemudian baca kembali berkas tersebut dalam runtime baru.
966. Tuliskan dua paragraf interpretasi: paragraf pertama hanya memuat fakta keluaran program, sedangkan paragraf kedua memuat makna dan keterbatasannya.

E.51 Membuat Konformer ETKDGv3

Resep ini membahas embedding 3D, random seed, dan status hasil. Tujuannya bukan sekadar memperoleh keluaran yang tampak benar, tetapi membangun alur notebook yang dapat dijalankan ulang dan diperiksa. Pembaca perlu menyatakan input secara eksplisit, memisahkan konfigurasi dari proses utama, mencetak status penting, dan menyimpan artefak yang diperlukan. Hasil utama yang ditargetkan adalah koordinat tiga dimensi yang dapat diulang. Setiap angka harus disertai satuan dan konteks, sedangkan setiap berkas perlu diberi nama yang menjelaskan molekul, metode, atau tahap proses.

Tujuan, Input, dan Keluaran

Input resep meliputi parameter yang relevan dengan embedding 3D, random seed, dan status hasil. Sebelum eksekusi, periksa tipe data, bentuk array, jalur berkas, dan ketersediaan pustaka. Keluaran harus dibedakan menjadi keluaran langsung program, hasil validasi, dan interpretasi. Keluaran langsung dapat berupa nilai, tabel, gambar, atau berkas; hasil validasi menyatakan apakah syarat minimum terpenuhi; interpretasi menjelaskan makna hasil tanpa melampaui metode yang digunakan.

Kode Siap Google Colab

```
%pip -q install rdkit
from rdkit import Chem
from rdkit.Chem import AllChem

mol = Chem.AddHs(Chem.MolFromSmiles("CCO"))
params = AllChem.ETKDGv3()
params.randomSeed = 2026
status = AllChem.EmbedMolecule(mol, params)
if status != 0:
    raise RuntimeError("Embedding gagal")
conf = mol.GetConformer()
print("Konformer 3D:", conf.Is3D())
```

Cara Membaca dan Memvalidasi Hasil

Setelah sel dijalankan, baca pesan dari atas ke bawah dan pastikan tidak ada pengecualian yang terlewat. Verifikasi bahwa keluaran sesuai dengan koordinat tiga dimensi yang dapat diulang. Untuk keluaran numerik, periksa bilangan hingga, satuan, tanda, skala, dan batas kewajaran. Untuk berkas, periksa keberadaan, ukuran, encoding, dan kemampuan untuk dibuka kembali. Untuk grafik atau visualisasi, pastikan label sumbu, satuan, legenda, dan judul menjelaskan konteks data.

Risiko utama pada resep ini adalah Satu konformer tidak selalu mewakili minimum global untuk molekul fleksibel. Oleh karena itu, status keberhasilan tidak boleh disimpulkan hanya dari munculnya output. Jalankan kembali notebook dari sesi bersih, bandingkan hasil dengan perhitungan sederhana atau data acuan ketika tersedia, dan catat versi lingkungan. Bila hasil berubah, identifikasi apakah perubahan berasal dari input, urutan eksekusi, versi pustaka, perangkat, atau algoritma.

Pengembangan dan Latihan

967. Buat sepuluh konformer, optimasi, dan pilih energi force-field terendah.
968. Tambahkan satu pemeriksaan yang sengaja menghentikan proses ketika input tidak memenuhi syarat, lalu uji pesan kesalahannya.
969. Simpan parameter, status, waktu eksekusi, dan hasil utama ke JSON atau CSV; kemudian baca kembali berkas tersebut dalam runtime baru.
970. Tuliskan dua paragraf interpretasi: paragraf pertama hanya memuat fakta keluaran program, sedangkan paragraf kedua memuat makna dan keterbatasannya.

E.52 Mengoptimasi Awal dengan MMFF94 atau UFF

Resep ini membahas force field, ketersediaan parameter, dan fallback. Tujuannya bukan sekadar memperoleh keluaran yang tampak benar, tetapi membangun alur notebook yang dapat dijalankan ulang dan diperiksa. Pembaca perlu menyatakan input secara eksplisit, memisahkan konfigurasi dari proses utama, mencetak status penting, dan menyimpan artefak yang diperlukan. Hasil utama yang ditargetkan adalah geometri awal yang lebih

layak untuk DFT. Setiap angka harus disertai satuan dan konteks, sedangkan setiap berkas perlu diberi nama yang menjelaskan molekul, metode, atau tahap proses.

Tujuan, Input, dan Keluaran

Input resep meliputi parameter yang relevan dengan force field, ketersediaan parameter, dan fallback. Sebelum eksekusi, periksa tipe data, bentuk array, jalur berkas, dan ketersediaan pustaka. Keluaran harus dibedakan menjadi keluaran langsung program, hasil validasi, dan interpretasi. Keluaran langsung dapat berupa nilai, tabel, gambar, atau berkas; hasil validasi menyatakan apakah syarat minimum terpenuhi; interpretasi menjelaskan makna hasil tanpa melampaui metode yang digunakan.

Kode Siap Google Colab

```
from rdkit import Chem
from rdkit.Chem import AllChem

mol = Chem.AddHs(Chem.MolFromSmiles("CCO"))
AllChem.EmbedMolecule(mol, AllChem.ETKDGv3())
if AllChem.MMFFHasAllMoleculeParams(mol):
    status = AllChem.MMFFOptimizeMolecule(mol, maxIters=500)
    metode = "MMFF94"
else:
    status = AllChem.UFFOptimizeMolecule(mol, maxIters=500)
    metode = "UFF"
print("Metode:", metode, "status:", status)
```

Cara Membaca dan Memvalidasi Hasil

Setelah sel dijalankan, baca pesan dari atas ke bawah dan pastikan tidak ada pengecualian yang terlewat. Verifikasi bahwa keluaran sesuai dengan geometri awal yang lebih layak untuk DFT. Untuk keluaran numerik, periksa bilangan hingga, satuan, tanda, skala, dan batas kewajaran. Untuk berkas, periksa keberadaan, ukuran, encoding, dan kemampuan untuk dibuka kembali. Untuk grafik atau visualisasi, pastikan label sumbu, satuan, legenda, dan judul menjelaskan konteks data.

Risiko utama pada resep ini adalah Optimasi mekanika molekul bukan pengganti optimasi kuantum ketika sifat elektronik diperlukan. Oleh karena itu, status keberhasilan tidak boleh disimpulkan hanya dari munculnya output. Jalankan kembali notebook dari sesi bersih, bandingkan hasil dengan perhitungan sederhana atau data acuan ketika tersedia, dan catat versi lingkungan. Bila hasil berubah, identifikasi apakah perubahan berasal dari input, urutan eksekusi, versi pustaka, perangkat, atau algoritma.

Pengembangan dan Latihan

971. Bandingkan energi relatif beberapa konformer menggunakan force field yang sama.

972. Tambahkan satu pemeriksaan yang sengaja menghentikan proses ketika input tidak memenuhi syarat, lalu uji pesan kesalahannya.
973. Simpan parameter, status, waktu eksekusi, dan hasil utama ke JSON atau CSV; kemudian baca kembali berkas tersebut dalam runtime baru.
974. Tuliskan dua paragraf interpretasi: paragraf pertama hanya memuat fakta keluaran program, sedangkan paragraf kedua memuat makna dan keterbatasannya.

E.53 Mengekspor Koordinat RDKit ke XYZ

Resep ini membahas conformer, simbol atom, koordinat, dan format XYZ. Tujuannya bukan sekadar memperoleh keluaran yang tampak benar, tetapi membangun alur notebook yang dapat dijalankan ulang dan diperiksa. Pembaca perlu menyatakan input secara eksplisit, memisahkan konfigurasi dari proses utama, mencetak status penting, dan menyimpan artefak yang diperlukan. Hasil utama yang ditargetkan adalah berkas XYZ dengan jumlah atom dan komentar. Setiap angka harus disertai satuan dan konteks, sedangkan setiap berkas perlu diberi nama yang menjelaskan molekul, metode, atau tahap proses.

Tujuan, Input, dan Keluaran

Input resep meliputi parameter yang relevan dengan conformer, simbol atom, koordinat, dan format XYZ. Sebelum eksekusi, periksa tipe data, bentuk array, jalur berkas, dan ketersediaan pustaka. Keluaran harus dibedakan menjadi keluaran langsung program, hasil validasi, dan interpretasi. Keluaran langsung dapat berupa nilai, tabel, gambar, atau berkas; hasil validasi menyatakan apakah syarat minimum terpenuhi; interpretasi menjelaskan makna hasil tanpa melampaui metode yang digunakan.

Kode Siap Google Colab

```
from pathlib import Path

conf = mol.GetConformer()
baris = [str(mol.GetNumAtoms()), "Etanol hasil optimasi awal"]
for atom in mol.GetAtoms():
    i = atom.GetIdx(); p = conf.GetAtomPosition(i)
    baris.append(f"{atom.GetSymbol():2s} {p.x:14.8f} {p.y:14.8f} {p.z:14.8f}")
Path("etanol.xyz").write_text("".join(baris)+"")
print("etanol.xyz dibuat")
```

Cara Membaca dan Memvalidasi Hasil

Setelah sel dijalankan, baca pesan dari atas ke bawah dan pastikan tidak ada pengecualian yang terlewat. Verifikasi bahwa keluaran sesuai dengan berkas XYZ dengan jumlah atom dan komentar. Untuk keluaran numerik, periksa bilangan hingga, satuan, tanda, skala, dan batas kewajaran. Untuk berkas, periksa keberadaan, ukuran, encoding, dan kemampuan

untuk dibuka kembali. Untuk grafik atau visualisasi, pastikan label sumbu, satuan, legenda, dan judul menjelaskan konteks data.

Risiko utama pada resep ini adalah Jumlah atom pada baris pertama harus sama dengan jumlah baris koordinat. Oleh karena itu, status keberhasilan tidak boleh disimpulkan hanya dari munculnya output. Jalankan kembali notebook dari sesi bersih, bandingkan hasil dengan perhitungan sederhana atau data acuan ketika tersedia, dan catat versi lingkungan. Bila hasil berubah, identifikasi apakah perubahan berasal dari input, urutan eksekusi, versi pustaka, perangkat, atau algoritma.

Pengembangan dan Latihan

975. Baca kembali XYZ dan validasi semua koordinat sebagai bilangan hingga.
976. Tambahkan satu pemeriksaan yang sengaja menghentikan proses ketika input tidak memenuhi syarat, lalu uji pesan kesalahannya.
977. Simpan parameter, status, waktu eksekusi, dan hasil utama ke JSON atau CSV; kemudian baca kembali berkas tersebut dalam runtime baru.
978. Tuliskan dua paragraf interpretasi: paragraf pertama hanya memuat fakta keluaran program, sedangkan paragraf kedua memuat makna dan keterbatasannya.

E.54 Menampilkan Molekul dengan py3Dmol

Resep ini membahas format XYZ, model stick-sphere, dan kontrol tampilan. Tujuannya bukan sekadar memperoleh keluaran yang tampak benar, tetapi membangun alur notebook yang dapat dijalankan ulang dan diperiksa. Pembaca perlu menyatakan input secara eksplisit, memisahkan konfigurasi dari proses utama, mencetak status penting, dan menyimpan artefak yang diperlukan. Hasil utama yang ditargetkan adalah visualisasi tiga dimensi interaktif. Setiap angka harus disertai satuan dan konteks, sedangkan setiap berkas perlu diberi nama yang menjelaskan molekul, metode, atau tahap proses.

Tujuan, Input, dan Keluaran

Input resep meliputi parameter yang relevan dengan format XYZ, model stick-sphere, dan kontrol tampilan. Sebelum eksekusi, periksa tipe data, bentuk array, jalur berkas, dan ketersediaan pustaka. Keluaran harus dibedakan menjadi keluaran langsung program, hasil validasi, dan interpretasi. Keluaran langsung dapat berupa nilai, tabel, gambar, atau berkas; hasil validasi menyatakan apakah syarat minimum terpenuhi; interpretasi menjelaskan makna hasil tanpa melampaui metode yang digunakan.

Kode Siap Google Colab

```
%pip -q install py3Dmol
import py3Dmol
from pathlib import Path

xyz = Path("etanol.xyz").read_text()
view = py3Dmol.view(width=700, height=450)
```

```
view.addModel(xyz, "xyz")
view.setStyle({"stick":{}} , "sphere":{"scale":0.28})
view.zoomTo()
view.show()
```

Cara Membaca dan Memvalidasi Hasil

Setelah sel dijalankan, baca pesan dari atas ke bawah dan pastikan tidak ada pengecualian yang terlewat. Verifikasi bahwa keluaran sesuai dengan visualisasi tiga dimensi interaktif. Untuk keluaran numerik, periksa bilangan hingga, satuan, tanda, skala, dan batas kewajaran. Untuk berkas, periksa keberadaan, ukuran, encoding, dan kemampuan untuk dibuka kembali. Untuk grafik atau visualisasi, pastikan label sumbu, satuan, legenda, dan judul menjelaskan konteks data.

Risiko utama pada resep ini adalah Visualisasi menunjukkan geometri, tetapi tidak membuktikan struktur telah teroptimasi atau minimum. Oleh karena itu, status keberhasilan tidak boleh disimpulkan hanya dari munculnya output. Jalankan kembali notebook dari sesi bersih, bandingkan hasil dengan perhitungan sederhana atau data acuan ketika tersedia, dan catat versi lingkungan. Bila hasil berubah, identifikasi apakah perubahan berasal dari input, urutan eksekusi, versi pustaka, perangkat, atau algoritma.

Pengembangan dan Latihan

979. Tambahkan label nomor atom dan tampilkan sebelum-sesudah optimasi.
980. Tambahkan satu pemeriksaan yang sengaja menghentikan proses ketika input tidak memenuhi syarat, lalu uji pesan kesalahannya.
981. Simpan parameter, status, waktu eksekusi, dan hasil utama ke JSON atau CSV; kemudian baca kembali berkas tersebut dalam runtime baru.
982. Tuliskan dua paragraf interpretasi: paragraf pertama hanya memuat fakta keluaran program, sedangkan paragraf kedua memuat makna dan keterbatasannya.

E.55 Menghitung Energi H₂ dengan RHF

Resep ini membahas objek `gto.M`, basis set, charge, spin, RHF, dan konvergensi. Tujuannya bukan sekadar memperoleh keluaran yang tampak benar, tetapi membangun alur notebook yang dapat dijalankan ulang dan diperiksa. Pembaca perlu menyatakan input secara eksplisit, memisahkan konfigurasi dari proses utama, mencetak status penting, dan menyimpan artefak yang diperlukan. Hasil utama yang ditargetkan adalah energi total H₂ pada tingkat HF/STO-3G. Setiap angka harus disertai satuan dan konteks, sedangkan setiap berkas perlu diberi nama yang menjelaskan molekul, metode, atau tahap proses.

Tujuan, Input, dan Keluaran

Input resep meliputi parameter yang relevan dengan objek `gto.M`, basis set, charge, spin, RHF, dan konvergensi. Sebelum eksekusi, periksa tipe data, bentuk array, jalur berkas, dan ketersediaan pustaka. Keluaran harus dibedakan menjadi keluaran langsung program,

hasil validasi, dan interpretasi. Keluaran langsung dapat berupa nilai, tabel, gambar, atau berkas; hasil validasi menyatakan apakah syarat minimum terpenuhi; interpretasi menjelaskan makna hasil tanpa melampaui metode yang digunakan.

Kode Siap Google Colab

```
%pip -q install --prefer-binary pyscf
from pyscf import gto, scf

mol = gto.M(atom="H 0 0 0; H 0 0 0.74", unit="Angstrom",
            basis="sto-3g", charge=0, spin=0, verbose=0)
mf = scf.RHF(mol)
mf.max_cycle = 200
energi = mf.kernel()
if not mf.converged:
    raise RuntimeError("SCF tidak konvergen")
print(f"Energi = {energi:.12f} Hartree")
```

Cara Membaca dan Memvalidasi Hasil

Setelah sel dijalankan, baca pesan dari atas ke bawah dan pastikan tidak ada pengecualian yang terlewat. Verifikasi bahwa keluaran sesuai dengan energi total H₂ pada tingkat HF/STO-3G. Untuk keluaran numerik, periksa bilangan hingga, satuan, tanda, skala, dan batas kewajaran. Untuk berkas, periksa keberadaan, ukuran, encoding, dan kemampuan untuk dibuka kembali. Untuk grafik atau visualisasi, pastikan label sumbu, satuan, legenda, dan judul menjelaskan konteks data.

Risiko utama pada resep ini adalah Energi absolut hanya boleh dibandingkan pada sistem dan tingkat teori yang konsisten. Oleh karena itu, status keberhasilan tidak boleh disimpulkan hanya dari munculnya output. Jalankan kembali notebook dari sesi bersih, bandingkan hasil dengan perhitungan sederhana atau data acuan ketika tersedia, dan catat versi lingkungan. Bila hasil berubah, identifikasi apakah perubahan berasal dari input, urutan eksekusi, versi pustaka, perangkat, atau algoritma.

Pengembangan dan Latihan

983. Ulangi dengan basis 6-31G dan jelaskan mengapa energi berubah.
984. Tambahkan satu pemeriksaan yang sengaja menghentikan proses ketika input tidak memenuhi syarat, lalu uji pesan kesalahannya.
985. Simpan parameter, status, waktu eksekusi, dan hasil utama ke JSON atau CSV; kemudian baca kembali berkas tersebut dalam runtime baru.
986. Tuliskan dua paragraf interpretasi: paragraf pertama hanya memuat fakta keluaran program, sedangkan paragraf kedua memuat makna dan keterbatasannya.

E.56 Memindai Kurva Energi H₂

Resep ini membahas loop geometri, reuse density matrix, status konvergensi, dan minimum. Tujuannya bukan sekadar memperoleh keluaran yang tampak benar, tetapi

membangun alur notebook yang dapat dijalankan ulang dan diperiksa. Pembaca perlu menyatakan input secara eksplisit, memisahkan konfigurasi dari proses utama, mencetak status penting, dan menyimpan artefak yang diperlukan. Hasil utama yang ditargetkan adalah tabel energi terhadap jarak H-H. Setiap angka harus disertai satuan dan konteks, sedangkan setiap berkas perlu diberi nama yang menjelaskan molekul, metode, atau tahap proses.

Tujuan, Input, dan Keluaran

Input resep meliputi parameter yang relevan dengan loop geometri, reuse density matrix, status konvergensi, dan minimum. Sebelum eksekusi, periksa tipe data, bentuk array, jalur berkas, dan ketersediaan pustaka. Keluaran harus dibedakan menjadi keluaran langsung program, hasil validasi, dan interpretasi. Keluaran langsung dapat berupa nilai, tabel, gambar, atau berkas; hasil validasi menyatakan apakah syarat minimum terpenuhi; interpretasi menjelaskan makna hasil tanpa melampaui metode yang digunakan.

Kode Siap Google Colab

```
import numpy as np
from pyscf import gto, scf

jarak = np.arange(0.40, 2.01, 0.10)
data, dm = [], None
for r in jarak:
    mol = gto.M(atom=f"H 0 0 0; H 0 0 {r}", unit="Angstrom",
                basis="sto-3g", charge=0, spin=0, verbose=0)
    mf = scf.RHF(mol); mf.max_cycle = 200
    E = mf.kernel(dm0=dm)
    data.append((r, E, mf.converged))
    if mf.converged: dm = mf.make_rdm1()
print(min((x for x in data if x[2]), key=lambda x:x[1]))
```

Cara Membaca dan Memvalidasi Hasil

Setelah sel dijalankan, baca pesan dari atas ke bawah dan pastikan tidak ada pengecualian yang terlewat. Verifikasi bahwa keluaran sesuai dengan tabel energi terhadap jarak H-H. Untuk keluaran numerik, periksa bilangan hingga, satuan, tanda, skala, dan batas kewajaran. Untuk berkas, periksa keberadaan, ukuran, encoding, dan kemampuan untuk dibuka kembali. Untuk grafik atau visualisasi, pastikan label sumbu, satuan, legenda, dan judul menjelaskan konteks data.

Risiko utama pada resep ini adalah Grid diskret memberi perkiraan minimum; optimasi geometri diperlukan untuk hasil lebih presisi. Oleh karena itu, status keberhasilan tidak boleh disimpulkan hanya dari munculnya output. Jalankan kembali notebook dari sesi bersih, bandingkan hasil dengan perhitungan sederhana atau data acuan ketika tersedia, dan catat versi lingkungan. Bila hasil berubah, identifikasi apakah perubahan berasal dari input, urutan eksekusi, versi pustaka, perangkat, atau algoritma.

Pengembangan dan Latihan

987. Simpan data ke CSV dan buat grafik energi relatif terhadap minimum.
988. Tambahkan satu pemeriksaan yang sengaja menghentikan proses ketika input tidak memenuhi syarat, lalu uji pesan kesalahannya.
989. Simpan parameter, status, waktu eksekusi, dan hasil utama ke JSON atau CSV; kemudian baca kembali berkas tersebut dalam runtime baru.
990. Tuliskan dua paragraf interpretasi: paragraf pertama hanya memuat fakta keluaran program, sedangkan paragraf kedua memuat makna dan keterbatasannya.

E.57 Menghitung H₂O dengan DFT B3LYP

Resep ini membahas RKS, fungsional, grid, toleransi, dan basis set. Tujuannya bukan sekadar memperoleh keluaran yang tampak benar, tetapi membangun alur notebook yang dapat dijalankan ulang dan diperiksa. Pembaca perlu menyatakan input secara eksplisit, memisahkan konfigurasi dari proses utama, mencetak status penting, dan menyimpan artefak yang diperlukan. Hasil utama yang ditargetkan adalah energi H₂O dengan status konvergensi. Setiap angka harus disertai satuan dan konteks, sedangkan setiap berkas perlu diberi nama yang menjelaskan molekul, metode, atau tahap proses.

Tujuan, Input, dan Keluaran

Input resep meliputi parameter yang relevan dengan RKS, fungsional, grid, toleransi, dan basis set. Sebelum eksekusi, periksa tipe data, bentuk array, jalur berkas, dan ketersediaan pustaka. Keluaran harus dibedakan menjadi keluaran langsung program, hasil validasi, dan interpretasi. Keluaran langsung dapat berupa nilai, tabel, gambar, atau berkas; hasil validasi menyatakan apakah syarat minimum terpenuhi; interpretasi menjelaskan makna hasil tanpa melampaui metode yang digunakan.

Kode Siap Google Colab

```
from pyscf import gto, dft

geom = """O 0 0 0
H 0 0.757 0.586
H 0 -0.757 0.586"""
mol = gto.M(atom=geom, unit="Angstrom", basis="def2-svp",
            charge=0, spin=0, verbose=0)
mf = dft.RKS(mol)
mf.xc = "b3lyp"; mf.grids.level = 4
mf.conv_tol = 1e-9; mf.max_cycle = 200
E = mf.kernel()
print("Konvergen:", mf.converged, "Energi:", E)
```

Cara Membaca dan Memvalidasi Hasil

Setelah sel dijalankan, baca pesan dari atas ke bawah dan pastikan tidak ada pengecualian yang terlewat. Verifikasi bahwa keluaran sesuai dengan energi H₂O dengan status

konvergensi. Untuk keluaran numerik, periksa bilangan hingga, satuan, tanda, skala, dan batas kewajaran. Untuk berkas, periksa keberadaan, ukuran, encoding, dan kemampuan untuk dibuka kembali. Untuk grafik atau visualisasi, pastikan label sumbu, satuan, legenda, dan judul menjelaskan konteks data.

Risiko utama pada resep ini adalah DFT memerlukan grid numerik; perubahan grid dan fungsional dapat memengaruhi hasil. Oleh karena itu, status keberhasilan tidak boleh disimpulkan hanya dari munculnya output. Jalankan kembali notebook dari sesi bersih, bandingkan hasil dengan perhitungan sederhana atau data acuan ketika tersedia, dan catat versi lingkungan. Bila hasil berubah, identifikasi apakah perubahan berasal dari input, urutan eksekusi, versi pustaka, perangkat, atau algoritma.

Pengembangan dan Latihan

991. Bandingkan grid level 3 dan 4 serta catat energi dan waktu.
992. Tambahkan satu pemeriksaan yang sengaja menghentikan proses ketika input tidak memenuhi syarat, lalu uji pesan kesalahannya.
993. Simpan parameter, status, waktu eksekusi, dan hasil utama ke JSON atau CSV; kemudian baca kembali berkas tersebut dalam runtime baru.
994. Tuliskan dua paragraf interpretasi: paragraf pertama hanya memuat fakta keluaran program, sedangkan paragraf kedua memuat makna dan keterbatasannya.

E.58 Mengoptimasi Geometri H₂O dengan geomeTRIC

Resep ini membahas optimasi geometri, kriteria konvergensi, dan energi akhir. Tujuannya bukan sekadar memperoleh keluaran yang tampak benar, tetapi membangun alur notebook yang dapat dijalankan ulang dan diperiksa. Pembaca perlu menyatakan input secara eksplisit, memisahkan konfigurasi dari proses utama, mencetak status penting, dan menyimpan artefak yang diperlukan. Hasil utama yang ditargetkan adalah koordinat teroptimasi yang dapat diekspor. Setiap angka harus disertai satuan dan konteks, sedangkan setiap berkas perlu diberi nama yang menjelaskan molekul, metode, atau tahap proses.

Tujuan, Input, dan Keluaran

Input resep meliputi parameter yang relevan dengan optimasi geometri, kriteria konvergensi, dan energi akhir. Sebelum eksekusi, periksa tipe data, bentuk array, jalur berkas, dan ketersediaan pustaka. Keluaran harus dibedakan menjadi keluaran langsung program, hasil validasi, dan interpretasi. Keluaran langsung dapat berupa nilai, tabel, gambar, atau berkas; hasil validasi menyatakan apakah syarat minimum terpenuhi; interpretasi menjelaskan makna hasil tanpa melampaui metode yang digunakan.

Kode Siap Google Colab

```
%pip -q install geometric
from pyscf.geomopt.geometric_solver import optimize
```

```
from pyscf import dft

mol_opt = optimize(mf, maxsteps=100)
final = dft.RKS(mol_opt); final.xc="b3lyp"; final.grids.level=4
E_final = final.kernel()
print("Konvergen akhir:", final.converged)
print("Energi akhir:", E_final)
print(mol_opt.atom_coords(unit="Angstrom"))
```

Cara Membaca dan Memvalidasi Hasil

Setelah sel dijalankan, baca pesan dari atas ke bawah dan pastikan tidak ada pengecualian yang terlewat. Verifikasi bahwa keluaran sesuai dengan koordinat teroptimasi yang dapat diekspor. Untuk keluaran numerik, periksa bilangan hingga, satuan, tanda, skala, dan batas kewajaran. Untuk berkas, periksa keberadaan, ukuran, encoding, dan kemampuan untuk dibuka kembali. Untuk grafik atau visualisasi, pastikan label sumbu, satuan, legenda, dan judul menjelaskan konteks data.

Risiko utama pada resep ini adalah Optimasi geometri selesai belum memastikan titik tersebut minimum; frekuensi perlu diperiksa. Oleh karena itu, status keberhasilan tidak boleh disimpulkan hanya dari munculnya output. Jalankan kembali notebook dari sesi bersih, bandingkan hasil dengan perhitungan sederhana atau data acuan ketika tersedia, dan catat versi lingkungan. Bila hasil berubah, identifikasi apakah perubahan berasal dari input, urutan eksekusi, versi pustaka, perangkat, atau algoritma.

Pengembangan dan Latihan

995. Hitung panjang O-H dan sudut H-O-H sebelum dan sesudah optimasi.
996. Tambahkan satu pemeriksaan yang sengaja menghentikan proses ketika input tidak memenuhi syarat, lalu uji pesan kesalahannya.
997. Simpan parameter, status, waktu eksekusi, dan hasil utama ke JSON atau CSV; kemudian baca kembali berkas tersebut dalam runtime baru.
998. Tuliskan dua paragraf interpretasi: paragraf pertama hanya memuat fakta keluaran program, sedangkan paragraf kedua memuat makna dan keterbatasannya.

E.59 Menganalisis HOMO, LUMO, dan Celah Energi

Resep ini membahas okupansi orbital, indeks HOMO, energi orbital, dan konversi eV. Tujuannya bukan sekadar memperoleh keluaran yang tampak benar, tetapi membangun alur notebook yang dapat dijalankan ulang dan diperiksa. Pembaca perlu menyatakan input secara eksplisit, memisahkan konfigurasi dari proses utama, mencetak status penting, dan menyimpan artefak yang diperlukan. Hasil utama yang ditargetkan adalah energi HOMO-LUMO dengan pemeriksaan okupansi. Setiap angka harus disertai satuan dan konteks, sedangkan setiap berkas perlu diberi nama yang menjelaskan molekul, metode, atau tahap proses.

Tujuan, Input, dan Keluaran

Input resep meliputi parameter yang relevan dengan okupansi orbital, indeks HOMO, energi orbital, dan konversi eV. Sebelum eksekusi, periksa tipe data, bentuk array, jalur berkas, dan ketersediaan pustaka. Keluaran harus dibedakan menjadi keluaran langsung program, hasil validasi, dan interpretasi. Keluaran langsung dapat berupa nilai, tabel, gambar, atau berkas; hasil validasi menyatakan apakah syarat minimum terpenuhi; interpretasi menjelaskan makna hasil tanpa melampaui metode yang digunakan.

Kode Siap Google Colab

```
import numpy as np

occ = np.asarray(final.mo_occ)
eps = np.asarray(final.mo_energy)
idx_occ = np.where(occ > 0)[0]
idx_vir = np.where(occ == 0)[0]
if not len(idx_occ) or not len(idx_vir):
    raise RuntimeError("Orbital terisi/kosong tidak ditemukan")
homo, lumo = idx_occ[-1], idx_vir[0]
faktor = 27.211386245988
print("HOMO (eV):", eps[homo]*faktor)
print("LUMO (eV):", eps[lumo]*faktor)
print("Gap (eV):", (eps[lumo]-eps[homo])*faktor)
```

Cara Membaca dan Memvalidasi Hasil

Setelah sel dijalankan, baca pesan dari atas ke bawah dan pastikan tidak ada pengecualian yang terlewat. Verifikasi bahwa keluaran sesuai dengan energi HOMO-LUMO dengan pemeriksaan okupansi. Untuk keluaran numerik, periksa bilangan hingga, satuan, tanda, skala, dan batas kewajaran. Untuk berkas, periksa keberadaan, ukuran, encoding, dan kemampuan untuk dibuka kembali. Untuk grafik atau visualisasi, pastikan label sumbu, satuan, legenda, dan judul menjelaskan konteks data.

Risiko utama pada resep ini adalah Celah orbital Kohn-Sham tidak selalu sama dengan celah eksperimental atau energi eksitasi. Oleh karena itu, status keberhasilan tidak boleh disimpulkan hanya dari munculnya output. Jalankan kembali notebook dari sesi bersih, bandingkan hasil dengan perhitungan sederhana atau data acuan ketika tersedia, dan catat versi lingkungan. Bila hasil berubah, identifikasi apakah perubahan berasal dari input, urutan eksekusi, versi pustaka, perangkat, atau algoritma.

Pengembangan dan Latihan

999. Bandingkan hasil HF dan DFT menggunakan geometri yang sama.
1000. Tambahkan satu pemeriksaan yang sengaja menghentikan proses ketika input tidak memenuhi syarat, lalu uji pesan kesalahannya.
1001. Simpan parameter, status, waktu eksekusi, dan hasil utama ke JSON atau CSV; kemudian baca kembali berkas tersebut dalam runtime baru.

1002. Tuliskan dua paragraf interpretasi: paragraf pertama hanya memuat fakta keluaran program, sedangkan paragraf kedua memuat makna dan keterbatasannya.

E.60 Menghitung Momen Dipol

Resep ini membahas matriks densitas, distribusi muatan, komponen dipol, dan satuan. Tujuannya bukan sekadar memperoleh keluaran yang tampak benar, tetapi membangun alur notebook yang dapat dijalankan ulang dan diperiksa. Pembaca perlu menyatakan input secara eksplisit, memisahkan konfigurasi dari proses utama, mencetak status penting, dan menyimpan artefak yang diperlukan. Hasil utama yang ditargetkan adalah vektor dan besar momen dipol. Setiap angka harus disertai satuan dan konteks, sedangkan setiap berkas perlu diberi nama yang menjelaskan molekul, metode, atau tahap proses.

Tujuan, Input, dan Keluaran

Input resep meliputi parameter yang relevan dengan matriks densitas, distribusi muatan, komponen dipol, dan satuan. Sebelum eksekusi, periksa tipe data, bentuk array, jalur berkas, dan ketersediaan pustaka. Keluaran harus dibedakan menjadi keluaran langsung program, hasil validasi, dan interpretasi. Keluaran langsung dapat berupa nilai, tabel, gambar, atau berkas; hasil validasi menyatakan apakah syarat minimum terpenuhi; interpretasi menjelaskan makna hasil tanpa melampaui metode yang digunakan.

Kode Siap Google Colab

```
# Jalankan setelah objek final konvergen.
if not final.converged:
    raise RuntimeError("SCF akhir belum konvergen")
dipol = final.dip_moment(unit="Debye", verbose=0)
import numpy as np
print("Komponen (Debye):", dipol)
print("Besar (Debye):", np.linalg.norm(dipol))
```

Cara Membaca dan Memvalidasi Hasil

Setelah sel dijalankan, baca pesan dari atas ke bawah dan pastikan tidak ada pengecualian yang terlewat. Verifikasi bahwa keluaran sesuai dengan vektor dan besar momen dipol. Untuk keluaran numerik, periksa bilangan hingga, satuan, tanda, skala, dan batas kewajaran. Untuk berkas, periksa keberadaan, ukuran, encoding, dan kemampuan untuk dibuka kembali. Untuk grafik atau visualisasi, pastikan label sumbu, satuan, legenda, dan judul menjelaskan konteks data.

Risiko utama pada resep ini adalah Tanda komponen bergantung pada orientasi koordinat, sedangkan besar vektor harus invariant terhadap rotasi. Oleh karena itu, status keberhasilan tidak boleh disimpulkan hanya dari munculnya output. Jalankan kembali notebook dari sesi bersih, bandingkan hasil dengan perhitungan sederhana atau data acuan ketika tersedia, dan catat versi lingkungan. Bila hasil berubah, identifikasi apakah perubahan berasal dari input, urutan eksekusi, versi pustaka, perangkat, atau algoritma.

Pengembangan dan Latihan

1003. Rotasikan koordinat, hitung ulang, dan bandingkan besar dipol.
1004. Tambahkan satu pemeriksaan yang sengaja menghentikan proses ketika input tidak memenuhi syarat, lalu uji pesan kesalahannya.
1005. Simpan parameter, status, waktu eksekusi, dan hasil utama ke JSON atau CSV; kemudian baca kembali berkas tersebut dalam runtime baru.
1006. Tuliskan dua paragraf interpretasi: paragraf pertama hanya memuat fakta keluaran program, sedangkan paragraf kedua memuat makna dan keterbatasannya.

E.61 Menyimpan Ringkasan Perhitungan Kimia

Resep ini membahas JSON, XYZ, CSV, status konvergensi, dan metadata metode. Tujuannya bukan sekadar memperoleh keluaran yang tampak benar, tetapi membangun alur notebook yang dapat dijalankan ulang dan diperiksa. Pembaca perlu menyatakan input secara eksplisit, memisahkan konfigurasi dari proses utama, mencetak status penting, dan menyimpan artefak yang diperlukan. Hasil utama yang ditargetkan adalah paket hasil yang dapat diaudit. Setiap angka harus disertai satuan dan konteks, sedangkan setiap berkas perlu diberi nama yang menjelaskan molekul, metode, atau tahap proses.

Tujuan, Input, dan Keluaran

Input resep meliputi parameter yang relevan dengan JSON, XYZ, CSV, status konvergensi, dan metadata metode. Sebelum eksekusi, periksa tipe data, bentuk array, jalur berkas, dan ketersediaan pustaka. Keluaran harus dibedakan menjadi keluaran langsung program, hasil validasi, dan interpretasi. Keluaran langsung dapat berupa nilai, tabel, gambar, atau berkas; hasil validasi menyatakan apakah syarat minimum terpenuhi; interpretasi menjelaskan makna hasil tanpa melampaui metode yang digunakan.

Kode Siap Google Colab

```
import json
from pathlib import Path

ringkasan = {
    "molecule": "H2O", "method": "B3LYP", "basis": "def2-SVP",
    "charge": 0, "spin": 0, "converged": bool(final.converged),
    "energy_hartree": float(E_final),
}
Path("ringkasan_h2o.json").write_text(json.dumps(ringkasan, indent=2))
print(ringkasan)
```

Cara Membaca dan Memvalidasi Hasil

Setelah sel dijalankan, baca pesan dari atas ke bawah dan pastikan tidak ada pengecualian yang terlewat. Verifikasi bahwa keluaran sesuai dengan paket hasil yang dapat diaudit. Untuk keluaran numerik, periksa bilangan hingga, satuan, tanda, skala, dan batas kewajaran. Untuk berkas, periksa keberadaan, ukuran, encoding, dan kemampuan untuk

dibuka kembali. Untuk grafik atau visualisasi, pastikan label sumbu, satuan, legenda, dan judul menjelaskan konteks data.

Risiko utama pada resep ini adalah Menyimpan angka tanpa metode, basis set, satuan, dan status konvergensi membuat hasil sulit digunakan kembali. Oleh karena itu, status keberhasilan tidak boleh disimpulkan hanya dari munculnya output. Jalankan kembali notebook dari sesi bersih, bandingkan hasil dengan perhitungan sederhana atau data acuan ketika tersedia, dan catat versi lingkungan. Bila hasil berubah, identifikasi apakah perubahan berasal dari input, urutan eksekusi, versi pustaka, perangkat, atau algoritma.

Pengembangan dan Latihan

1007. Tambahkan panjang ikatan, sudut, HOMO, LUMO, gap, dipol, versi PySCF, dan checksum XYZ.
1008. Tambahkan satu pemeriksaan yang sengaja menghentikan proses ketika input tidak memenuhi syarat, lalu uji pesan kesalahannya.
1009. Simpan parameter, status, waktu eksekusi, dan hasil utama ke JSON atau CSV; kemudian baca kembali berkas tersebut dalam runtime baru.
1010. Tuliskan dua paragraf interpretasi: paragraf pertama hanya memuat fakta keluaran program, sedangkan paragraf kedua memuat makna dan keterbatasannya.

LAMPIRAN F - 50 LATIHAN, PROYEK, DAN RUBRIK PENILAIAN

Lampiran ini memuat latihan berjenjang yang dapat digunakan untuk belajar mandiri, tugas kuliah, workshop, atau asesmen. Setiap aktivitas menuntut bukti kerja berupa notebook yang dapat dijalankan dari runtime bersih, berkas hasil, catatan validasi, dan interpretasi singkat. Penilaian tidak hanya didasarkan pada keberhasilan kode, tetapi juga pada ketepatan asumsi, keterlacakan parameter, penanganan kesalahan, dan kesesuaian klaim dengan hasil.

F.1 Python dan Notebook

F.01 Membandingkan state kernel sebelum dan sesudah Restart session

Skenario

Anda diminta menyelesaikan aktivitas 'Membandingkan state kernel sebelum dan sesudah Restart session' dalam sebuah notebook Google Colab yang dapat dijalankan dari awal sampai akhir. Notebook harus memuat identitas kegiatan, tujuan, teori singkat, parameter, kode, keluaran, validasi, interpretasi, dan refleksi. Seluruh variabel penting harus diberi nama yang bermakna. Bila aktivitas melibatkan data atau kimia komputasi, satuan, metode, basis set, muatan, spin, dan status konvergensi harus dicatat secara eksplisit.

Tugas yang Harus Dikerjakan

1011. Rancang alur input-proses-validasi-ekspor untuk membandingkan state kernel sebelum dan sesudah restart session.
1012. Sediakan minimal satu kasus uji benar dan satu kasus uji salah yang menghasilkan pesan kesalahan informatif.
1013. Jalankan notebook setelah Restart session menggunakan Run all, kemudian simpan bukti keluaran utama.
1014. Ekspor hasil dalam paling sedikit dua format yang sesuai, misalnya CSV dan JSON, atau XYZ dan gambar.
1015. Tuliskan interpretasi yang membedakan fakta komputasi, kesimpulan, dan keterbatasan.

Kriteria Produk

Produk dinyatakan lengkap apabila tidak bergantung pada variabel tersembunyi, seluruh pustaka dipasang atau diperiksa, parameter dapat diubah pada satu tempat, pesan kesalahan tidak disembunyikan, dan artefak dapat dibuka kembali. Nama berkas harus informatif dan tidak menimpa hasil sebelumnya tanpa peringatan. Grafik harus memiliki label dan satuan, sedangkan tabel harus mempunyai nama kolom yang jelas.

Rubrik Analitik

- Skor 4: notebook berjalan dari runtime bersih, validasi lengkap, keluaran tersimpan, dan interpretasi tepat serta kritis.
- Skor 3: notebook berjalan dan keluaran benar, tetapi dokumentasi, kasus uji, atau interpretasi belum lengkap.
- Skor 2: notebook hanya berjalan setelah intervensi manual atau masih bergantung pada state lama; validasi terbatas.
- Skor 1: kode belum berjalan konsisten, sumber kesalahan belum diidentifikasi, dan bukti hasil tidak tersedia.

Pertanyaan Refleksi

Jelaskan perubahan paling penting yang Anda lakukan setelah menemukan kesalahan. Sebutkan bukti yang menunjukkan proses berhasil, bagian yang masih memerlukan verifikasi eksternal, dan satu keputusan desain notebook yang meningkatkan reproduksibilitas. Refleksi harus merujuk pada sel, pesan, tabel, atau berkas tertentu, bukan hanya menyatakan bahwa tugas telah selesai.

F.02 Membuat fungsi konversi satuan dengan unit test

Skenario

Anda diminta menyelesaikan aktivitas 'Membuat fungsi konversi satuan dengan unit test' dalam sebuah notebook Google Colab yang dapat dijalankan dari awal sampai akhir. Notebook harus memuat identitas kegiatan, tujuan, teori singkat, parameter, kode, keluaran, validasi, interpretasi, dan refleksi. Seluruh variabel penting harus diberi nama yang bermakna. Bila aktivitas melibatkan data atau kimia komputasi, satuan, metode, basis set, muatan, spin, dan status konvergensi harus dicatat secara eksplisit.

Tugas yang Harus Dikerjakan

1016. Rancang alur input-proses-validasi-ekspor untuk membuat fungsi konversi satuan dengan unit test.
1017. Sediakan minimal satu kasus uji benar dan satu kasus uji salah yang menghasilkan pesan kesalahan informatif.
1018. Jalankan notebook setelah Restart session menggunakan Run all, kemudian simpan bukti keluaran utama.
1019. Ekspor hasil dalam paling sedikit dua format yang sesuai, misalnya CSV dan JSON, atau XYZ dan gambar.
1020. Tuliskan interpretasi yang membedakan fakta komputasi, kesimpulan, dan keterbatasan.

Kriteria Produk

Produk dinyatakan lengkap apabila tidak bergantung pada variabel tersembunyi, seluruh pustaka dipasang atau diperiksa, parameter dapat diubah pada satu tempat, pesan kesalahan tidak disembunyikan, dan artefak dapat dibuka kembali. Nama berkas harus informatif dan tidak menimpa hasil sebelumnya tanpa peringatan. Grafik harus memiliki label dan satuan, sedangkan tabel harus mempunyai nama kolom yang jelas.

Rubrik Analitik

- Skor 4: notebook berjalan dari runtime bersih, validasi lengkap, keluaran tersimpan, dan interpretasi tepat serta kritis.
- Skor 3: notebook berjalan dan keluaran benar, tetapi dokumentasi, kasus uji, atau interpretasi belum lengkap.
- Skor 2: notebook hanya berjalan setelah intervensi manual atau masih bergantung pada state lama; validasi terbatas.
- Skor 1: kode belum berjalan konsisten, sumber kesalahan belum diidentifikasi, dan bukti hasil tidak tersedia.

Pertanyaan Refleksi

Jelaskan perubahan paling penting yang Anda lakukan setelah menemukan kesalahan. Sebutkan bukti yang menunjukkan proses berhasil, bagian yang masih memerlukan verifikasi eksternal, dan satu keputusan desain notebook yang meningkatkan reproduksibilitas. Refleksi harus merujuk pada sel, pesan, tabel, atau berkas tertentu, bukan hanya menyatakan bahwa tugas telah selesai.

F.03 Menyusun konfigurasi proyek dalam JSON

Skenario

Anda diminta menyelesaikan aktivitas 'Menyusun konfigurasi proyek dalam JSON' dalam sebuah notebook Google Colab yang dapat dijalankan dari awal sampai akhir. Notebook harus memuat identitas kegiatan, tujuan, teori singkat, parameter, kode, keluaran, validasi, interpretasi, dan refleksi. Seluruh variabel penting harus diberi nama yang bermakna. Bila aktivitas melibatkan data atau kimia komputasi, satuan, metode, basis set, muatan, spin, dan status konvergensi harus dicatat secara eksplisit.

Tugas yang Harus Dikerjakan

1021. Rancang alur input-proses-validasi-ekspor untuk menyusun konfigurasi proyek dalam json.
1022. Sediakan minimal satu kasus uji benar dan satu kasus uji salah yang menghasilkan pesan kesalahan informatif.
1023. Jalankan notebook setelah Restart session menggunakan Run all, kemudian simpan bukti keluaran utama.

1024. Ekspor hasil dalam paling sedikit dua format yang sesuai, misalnya CSV dan JSON, atau XYZ dan gambar.
1025. Tuliskan interpretasi yang membedakan fakta komputasi, kesimpulan, dan keterbatasan.

Kriteria Produk

Produk dinyatakan lengkap apabila tidak bergantung pada variabel tersembunyi, seluruh pustaka dipasang atau diperiksa, parameter dapat diubah pada satu tempat, pesan kesalahan tidak disembunyikan, dan artefak dapat dibuka kembali. Nama berkas harus informatif dan tidak menimpa hasil sebelumnya tanpa peringatan. Grafik harus memiliki label dan satuan, sedangkan tabel harus mempunyai nama kolom yang jelas.

Rubrik Analitik

- Skor 4: notebook berjalan dari runtime bersih, validasi lengkap, keluaran tersimpan, dan interpretasi tepat serta kritis.
- Skor 3: notebook berjalan dan keluaran benar, tetapi dokumentasi, kasus uji, atau interpretasi belum lengkap.
- Skor 2: notebook hanya berjalan setelah intervensi manual atau masih bergantung pada state lama; validasi terbatas.
- Skor 1: kode belum berjalan konsisten, sumber kesalahan belum diidentifikasi, dan bukti hasil tidak tersedia.

Pertanyaan Refleksi

Jelaskan perubahan paling penting yang Anda lakukan setelah menemukan kesalahan. Sebutkan bukti yang menunjukkan proses berhasil, bagian yang masih memerlukan verifikasi eksternal, dan satu keputusan desain notebook yang meningkatkan reproduksibilitas. Refleksi harus merujuk pada sel, pesan, tabel, atau berkas tertentu, bukan hanya menyatakan bahwa tugas telah selesai.

F.04 Membuat pencatat waktu untuk beberapa fungsi

Skenario

Anda diminta menyelesaikan aktivitas 'Membuat pencatat waktu untuk beberapa fungsi' dalam sebuah notebook Google Colab yang dapat dijalankan dari awal sampai akhir. Notebook harus memuat identitas kegiatan, tujuan, teori singkat, parameter, kode, keluaran, validasi, interpretasi, dan refleksi. Seluruh variabel penting harus diberi nama yang bermakna. Bila aktivitas melibatkan data atau kimia komputasi, satuan, metode, basis set, muatan, spin, dan status konvergensi harus dicatat secara eksplisit.

Tugas yang Harus Dikerjakan

1026. Rancang alur input-proses-validasi-ekspor untuk membuat pencatat waktu untuk beberapa fungsi.
1027. Sediakan minimal satu kasus uji benar dan satu kasus uji salah yang menghasilkan pesan kesalahan informatif.
1028. Jalankan notebook setelah Restart session menggunakan Run all, kemudian simpan bukti keluaran utama.
1029. Ekspor hasil dalam paling sedikit dua format yang sesuai, misalnya CSV dan JSON, atau XYZ dan gambar.
1030. Tuliskan interpretasi yang membedakan fakta komputasi, kesimpulan, dan keterbatasan.

Kriteria Produk

Produk dinyatakan lengkap apabila tidak bergantung pada variabel tersembunyi, seluruh pustaka dipasang atau diperiksa, parameter dapat diubah pada satu tempat, pesan kesalahan tidak disembunyikan, dan artefak dapat dibuka kembali. Nama berkas harus informatif dan tidak menimpa hasil sebelumnya tanpa peringatan. Grafik harus memiliki label dan satuan, sedangkan tabel harus mempunyai nama kolom yang jelas.

Rubrik Analitik

- Skor 4: notebook berjalan dari runtime bersih, validasi lengkap, keluaran tersimpan, dan interpretasi tepat serta kritis.
- Skor 3: notebook berjalan dan keluaran benar, tetapi dokumentasi, kasus uji, atau interpretasi belum lengkap.
- Skor 2: notebook hanya berjalan setelah intervensi manual atau masih bergantung pada state lama; validasi terbatas.
- Skor 1: kode belum berjalan konsisten, sumber kesalahan belum diidentifikasi, dan bukti hasil tidak tersedia.

Pertanyaan Refleksi

Jelaskan perubahan paling penting yang Anda lakukan setelah menemukan kesalahan. Sebutkan bukti yang menunjukkan proses berhasil, bagian yang masih memerlukan verifikasi eksternal, dan satu keputusan desain notebook yang meningkatkan reproduksibilitas. Refleksi harus merujuk pada sel, pesan, tabel, atau berkas tertentu, bukan hanya menyatakan bahwa tugas telah selesai.

F.05 Mendeteksi dan menjelaskan tiga jenis exception

Skenario

Anda diminta menyelesaikan aktivitas 'Mendeteksi dan menjelaskan tiga jenis exception' dalam sebuah notebook Google Colab yang dapat dijalankan dari awal sampai akhir.

Notebook harus memuat identitas kegiatan, tujuan, teori singkat, parameter, kode, keluaran, validasi, interpretasi, dan refleksi. Seluruh variabel penting harus diberi nama yang bermakna. Bila aktivitas melibatkan data atau kimia komputasi, satuan, metode, basis set, muatan, spin, dan status konvergensi harus dicatat secara eksplisit.

Tugas yang Harus Dikerjakan

1031. Rancang alur input-proses-validasi-ekspor untuk mendeteksi dan menjelaskan tiga jenis exception.
1032. Sediakan minimal satu kasus uji benar dan satu kasus uji salah yang menghasilkan pesan kesalahan informatif.
1033. Jalankan notebook setelah Restart session menggunakan Run all, kemudian simpan bukti keluaran utama.
1034. Ekspor hasil dalam paling sedikit dua format yang sesuai, misalnya CSV dan JSON, atau XYZ dan gambar.
1035. Tuliskan interpretasi yang membedakan fakta komputasi, kesimpulan, dan keterbatasan.

Kriteria Produk

Produk dinyatakan lengkap apabila tidak bergantung pada variabel tersembunyi, seluruh pustaka dipasang atau diperiksa, parameter dapat diubah pada satu tempat, pesan kesalahan tidak disembunyikan, dan artefak dapat dibuka kembali. Nama berkas harus informatif dan tidak menimpa hasil sebelumnya tanpa peringatan. Grafik harus memiliki label dan satuan, sedangkan tabel harus mempunyai nama kolom yang jelas.

Rubrik Analitik

- Skor 4: notebook berjalan dari runtime bersih, validasi lengkap, keluaran tersimpan, dan interpretasi tepat serta kritis.
- Skor 3: notebook berjalan dan keluaran benar, tetapi dokumentasi, kasus uji, atau interpretasi belum lengkap.
- Skor 2: notebook hanya berjalan setelah intervensi manual atau masih bergantung pada state lama; validasi terbatas.
- Skor 1: kode belum berjalan konsisten, sumber kesalahan belum diidentifikasi, dan bukti hasil tidak tersedia.

Pertanyaan Refleksi

Jelaskan perubahan paling penting yang Anda lakukan setelah menemukan kesalahan. Sebutkan bukti yang menunjukkan proses berhasil, bagian yang masih memerlukan verifikasi eksternal, dan satu keputusan desain notebook yang meningkatkan reproduksibilitas. Refleksi harus merujuk pada sel, pesan, tabel, atau berkas tertentu, bukan hanya menyatakan bahwa tugas telah selesai.

F.06 Menyusun struktur folder proyek menggunakan pathlib

Skenario

Anda diminta menyelesaikan aktivitas 'Menyusun struktur folder proyek menggunakan pathlib' dalam sebuah notebook Google Colab yang dapat dijalankan dari awal sampai akhir. Notebook harus memuat identitas kegiatan, tujuan, teori singkat, parameter, kode, keluaran, validasi, interpretasi, dan refleksi. Seluruh variabel penting harus diberi nama yang bermakna. Bila aktivitas melibatkan data atau kimia komputasi, satuan, metode, basis set, muatan, spin, dan status konvergensi harus dicatat secara eksplisit.

Tugas yang Harus Dikerjakan

1036. Rancang alur input-proses-validasi-ekspor untuk menyusun struktur folder proyek menggunakan pathlib.
1037. Sediakan minimal satu kasus uji benar dan satu kasus uji salah yang menghasilkan pesan kesalahan informatif.
1038. Jalankan notebook setelah Restart session menggunakan Run all, kemudian simpan bukti keluaran utama.
1039. Ekspor hasil dalam paling sedikit dua format yang sesuai, misalnya CSV dan JSON, atau XYZ dan gambar.
1040. Tuliskan interpretasi yang membedakan fakta komputasi, kesimpulan, dan keterbatasan.

Kriteria Produk

Produk dinyatakan lengkap apabila tidak bergantung pada variabel tersembunyi, seluruh pustaka dipasang atau diperiksa, parameter dapat diubah pada satu tempat, pesan kesalahan tidak disembunyikan, dan artefak dapat dibuka kembali. Nama berkas harus informatif dan tidak menimpa hasil sebelumnya tanpa peringatan. Grafik harus memiliki label dan satuan, sedangkan tabel harus mempunyai nama kolom yang jelas.

Rubrik Analitik

- Skor 4: notebook berjalan dari runtime bersih, validasi lengkap, keluaran tersimpan, dan interpretasi tepat serta kritis.
- Skor 3: notebook berjalan dan keluaran benar, tetapi dokumentasi, kasus uji, atau interpretasi belum lengkap.
- Skor 2: notebook hanya berjalan setelah intervensi manual atau masih bergantung pada state lama; validasi terbatas.
- Skor 1: kode belum berjalan konsisten, sumber kesalahan belum diidentifikasi, dan bukti hasil tidak tersedia.

Pertanyaan Refleksi

Jelaskan perubahan paling penting yang Anda lakukan setelah menemukan kesalahan. Sebutkan bukti yang menunjukkan proses berhasil, bagian yang masih memerlukan verifikasi eksternal, dan satu keputusan desain notebook yang meningkatkan reproduksibilitas. Refleksi harus merujuk pada sel, pesan, tabel, atau berkas tertentu, bukan hanya menyatakan bahwa tugas telah selesai.

F.07 Membuat log eksekusi dengan tiga level pesan

Skenario

Anda diminta menyelesaikan aktivitas 'Membuat log eksekusi dengan tiga level pesan' dalam sebuah notebook Google Colab yang dapat dijalankan dari awal sampai akhir. Notebook harus memuat identitas kegiatan, tujuan, teori singkat, parameter, kode, keluaran, validasi, interpretasi, dan refleksi. Seluruh variabel penting harus diberi nama yang bermakna. Bila aktivitas melibatkan data atau kimia komputasi, satuan, metode, basis set, muatan, spin, dan status konvergensi harus dicatat secara eksplisit.

Tugas yang Harus Dikerjakan

1041. Rancang alur input-proses-validasi-ekspor untuk membuat log eksekusi dengan tiga level pesan.
1042. Sediakan minimal satu kasus uji benar dan satu kasus uji salah yang menghasilkan pesan kesalahan informatif.
1043. Jalankan notebook setelah Restart session menggunakan Run all, kemudian simpan bukti keluaran utama.
1044. Ekspor hasil dalam paling sedikit dua format yang sesuai, misalnya CSV dan JSON, atau XYZ dan gambar.
1045. Tuliskan interpretasi yang membedakan fakta komputasi, kesimpulan, dan keterbatasan.

Kriteria Produk

Produk dinyatakan lengkap apabila tidak bergantung pada variabel tersembunyi, seluruh pustaka dipasang atau diperiksa, parameter dapat diubah pada satu tempat, pesan kesalahan tidak disembunyikan, dan artefak dapat dibuka kembali. Nama berkas harus informatif dan tidak menimpa hasil sebelumnya tanpa peringatan. Grafik harus memiliki label dan satuan, sedangkan tabel harus mempunyai nama kolom yang jelas.

Rubrik Analitik

- Skor 4: notebook berjalan dari runtime bersih, validasi lengkap, keluaran tersimpan, dan interpretasi tepat serta kritis.
- Skor 3: notebook berjalan dan keluaran benar, tetapi dokumentasi, kasus uji, atau interpretasi belum lengkap.

- Skor 2: notebook hanya berjalan setelah intervensi manual atau masih bergantung pada state lama; validasi terbatas.
- Skor 1: kode belum berjalan konsisten, sumber kesalahan belum diidentifikasi, dan bukti hasil tidak tersedia.

Pertanyaan Refleksi

Jelaskan perubahan paling penting yang Anda lakukan setelah menemukan kesalahan. Sebutkan bukti yang menunjukkan proses berhasil, bagian yang masih memerlukan verifikasi eksternal, dan satu keputusan desain notebook yang meningkatkan reproduksibilitas. Refleksi harus merujuk pada sel, pesan, tabel, atau berkas tertentu, bukan hanya menyatakan bahwa tugas telah selesai.

F.08 Mengemas notebook dan artefak ke ZIP

Skenario

Anda diminta menyelesaikan aktivitas 'Mengemas notebook dan artefak ke ZIP' dalam sebuah notebook Google Colab yang dapat dijalankan dari awal sampai akhir. Notebook harus memuat identitas kegiatan, tujuan, teori singkat, parameter, kode, keluaran, validasi, interpretasi, dan refleksi. Seluruh variabel penting harus diberi nama yang bermakna. Bila aktivitas melibatkan data atau kimia komputasi, satuan, metode, basis set, muatan, spin, dan status konvergensi harus dicatat secara eksplisit.

Tugas yang Harus Dikerjakan

1046. Rancang alur input-proses-validasi-ekspor untuk mengemas notebook dan artefak ke zip.
1047. Sediakan minimal satu kasus uji benar dan satu kasus uji salah yang menghasilkan pesan kesalahan informatif.
1048. Jalankan notebook setelah Restart session menggunakan Run all, kemudian simpan bukti keluaran utama.
1049. Ekspor hasil dalam paling sedikit dua format yang sesuai, misalnya CSV dan JSON, atau XYZ dan gambar.
1050. Tuliskan interpretasi yang membedakan fakta komputasi, kesimpulan, dan keterbatasan.

Kriteria Produk

Produk dinyatakan lengkap apabila tidak bergantung pada variabel tersembunyi, seluruh pustaka dipasang atau diperiksa, parameter dapat diubah pada satu tempat, pesan kesalahan tidak disembunyikan, dan artefak dapat dibuka kembali. Nama berkas harus informatif dan tidak menimpa hasil sebelumnya tanpa peringatan. Grafik harus memiliki label dan satuan, sedangkan tabel harus mempunyai nama kolom yang jelas.

Rubrik Analitik

- Skor 4: notebook berjalan dari runtime bersih, validasi lengkap, keluaran tersimpan, dan interpretasi tepat serta kritis.
- Skor 3: notebook berjalan dan keluaran benar, tetapi dokumentasi, kasus uji, atau interpretasi belum lengkap.
- Skor 2: notebook hanya berjalan setelah intervensi manual atau masih bergantung pada state lama; validasi terbatas.
- Skor 1: kode belum berjalan konsisten, sumber kesalahan belum diidentifikasi, dan bukti hasil tidak tersedia.

Pertanyaan Refleksi

Jelaskan perubahan paling penting yang Anda lakukan setelah menemukan kesalahan. Sebutkan bukti yang menunjukkan proses berhasil, bagian yang masih memerlukan verifikasi eksternal, dan satu keputusan desain notebook yang meningkatkan reproduksibilitas. Refleksi harus merujuk pada sel, pesan, tabel, atau berkas tertentu, bukan hanya menyatakan bahwa tugas telah selesai.

F.09 Membuat manifest checksum hasil

Skenario

Anda diminta menyelesaikan aktivitas 'Membuat manifest checksum hasil' dalam sebuah notebook Google Colab yang dapat dijalankan dari awal sampai akhir. Notebook harus memuat identitas kegiatan, tujuan, teori singkat, parameter, kode, keluaran, validasi, interpretasi, dan refleksi. Seluruh variabel penting harus diberi nama yang bermakna. Bila aktivitas melibatkan data atau kimia komputasi, satuan, metode, basis set, muatan, spin, dan status konvergensi harus dicatat secara eksplisit.

Tugas yang Harus Dikerjakan

1051. Rancang alur input-proses-validasi-ekspor untuk membuat manifest checksum hasil.
1052. Sediakan minimal satu kasus uji benar dan satu kasus uji salah yang menghasilkan pesan kesalahan informatif.
1053. Jalankan notebook setelah Restart session menggunakan Run all, kemudian simpan bukti keluaran utama.
1054. Ekspor hasil dalam paling sedikit dua format yang sesuai, misalnya CSV dan JSON, atau XYZ dan gambar.
1055. Tuliskan interpretasi yang membedakan fakta komputasi, kesimpulan, dan keterbatasan.

Kriteria Produk

Produk dinyatakan lengkap apabila tidak bergantung pada variabel tersembunyi, seluruh pustaka dipasang atau diperiksa, parameter dapat diubah pada satu tempat, pesan kesalahan tidak disembunyikan, dan artefak dapat dibuka kembali. Nama berkas harus informatif dan tidak menimpa hasil sebelumnya tanpa peringatan. Grafik harus memiliki label dan satuan, sedangkan tabel harus mempunyai nama kolom yang jelas.

Rubrik Analitik

- Skor 4: notebook berjalan dari runtime bersih, validasi lengkap, keluaran tersimpan, dan interpretasi tepat serta kritis.
- Skor 3: notebook berjalan dan keluaran benar, tetapi dokumentasi, kasus uji, atau interpretasi belum lengkap.
- Skor 2: notebook hanya berjalan setelah intervensi manual atau masih bergantung pada state lama; validasi terbatas.
- Skor 1: kode belum berjalan konsisten, sumber kesalahan belum diidentifikasi, dan bukti hasil tidak tersedia.

Pertanyaan Refleksi

Jelaskan perubahan paling penting yang Anda lakukan setelah menemukan kesalahan. Sebutkan bukti yang menunjukkan proses berhasil, bagian yang masih memerlukan verifikasi eksternal, dan satu keputusan desain notebook yang meningkatkan reproduksibilitas. Refleksi harus merujuk pada sel, pesan, tabel, atau berkas tertentu, bukan hanya menyatakan bahwa tugas telah selesai.

F.10 Menguji reproduksibilitas dengan Run all

Skenario

Anda diminta menyelesaikan aktivitas 'Menguji reproduksibilitas dengan Run all' dalam sebuah notebook Google Colab yang dapat dijalankan dari awal sampai akhir. Notebook harus memuat identitas kegiatan, tujuan, teori singkat, parameter, kode, keluaran, validasi, interpretasi, dan refleksi. Seluruh variabel penting harus diberi nama yang bermakna. Bila aktivitas melibatkan data atau kimia komputasi, satuan, metode, basis set, muatan, spin, dan status konvergensi harus dicatat secara eksplisit.

Tugas yang Harus Dikerjakan

1056. Rancang alur input-proses-validasi-ekspor untuk menguji reproduksibilitas dengan run all.
1057. Sediakan minimal satu kasus uji benar dan satu kasus uji salah yang menghasilkan pesan kesalahan informatif.
1058. Jalankan notebook setelah Restart session menggunakan Run all, kemudian simpan bukti keluaran utama.

1059. Ekspor hasil dalam paling sedikit dua format yang sesuai, misalnya CSV dan JSON, atau XYZ dan gambar.
1060. Tuliskan interpretasi yang membedakan fakta komputasi, kesimpulan, dan keterbatasan.

Kriteria Produk

Produk dinyatakan lengkap apabila tidak bergantung pada variabel tersembunyi, seluruh pustaka dipasang atau diperiksa, parameter dapat diubah pada satu tempat, pesan kesalahan tidak disembunyikan, dan artefak dapat dibuka kembali. Nama berkas harus informatif dan tidak menimpa hasil sebelumnya tanpa peringatan. Grafik harus memiliki label dan satuan, sedangkan tabel harus mempunyai nama kolom yang jelas.

Rubrik Analitik

- Skor 4: notebook berjalan dari runtime bersih, validasi lengkap, keluaran tersimpan, dan interpretasi tepat serta kritis.
- Skor 3: notebook berjalan dan keluaran benar, tetapi dokumentasi, kasus uji, atau interpretasi belum lengkap.
- Skor 2: notebook hanya berjalan setelah intervensi manual atau masih bergantung pada state lama; validasi terbatas.
- Skor 1: kode belum berjalan konsisten, sumber kesalahan belum diidentifikasi, dan bukti hasil tidak tersedia.

Pertanyaan Refleksi

Jelaskan perubahan paling penting yang Anda lakukan setelah menemukan kesalahan. Sebutkan bukti yang menunjukkan proses berhasil, bagian yang masih memerlukan verifikasi eksternal, dan satu keputusan desain notebook yang meningkatkan reproduksibilitas. Refleksi harus merujuk pada sel, pesan, tabel, atau berkas tertentu, bukan hanya menyatakan bahwa tugas telah selesai.

F.2 NumPy

F.11 Menghitung jarak semua pasangan atom

Skenario

Anda diminta menyelesaikan aktivitas 'Menghitung jarak semua pasangan atom' dalam sebuah notebook Google Colab yang dapat dijalankan dari awal sampai akhir. Notebook harus memuat identitas kegiatan, tujuan, teori singkat, parameter, kode, keluaran, validasi, interpretasi, dan refleksi. Seluruh variabel penting harus diberi nama yang bermakna. Bila aktivitas melibatkan data atau kimia komputasi, satuan, metode, basis set, muatan, spin, dan status konvergensi harus dicatat secara eksplisit.

Tugas yang Harus Dikerjakan

1061. Rancang alur input-proses-validasi-ekspor untuk menghitung jarak semua pasangan atom.
1062. Sediakan minimal satu kasus uji benar dan satu kasus uji salah yang menghasilkan pesan kesalahan informatif.
1063. Jalankan notebook setelah Restart session menggunakan Run all, kemudian simpan bukti keluaran utama.
1064. Ekspor hasil dalam paling sedikit dua format yang sesuai, misalnya CSV dan JSON, atau XYZ dan gambar.
1065. Tuliskan interpretasi yang membedakan fakta komputasi, kesimpulan, dan keterbatasan.

Kriteria Produk

Produk dinyatakan lengkap apabila tidak bergantung pada variabel tersembunyi, seluruh pustaka dipasang atau diperiksa, parameter dapat diubah pada satu tempat, pesan kesalahan tidak disembunyikan, dan artefak dapat dibuka kembali. Nama berkas harus informatif dan tidak menimpa hasil sebelumnya tanpa peringatan. Grafik harus memiliki label dan satuan, sedangkan tabel harus mempunyai nama kolom yang jelas.

Rubrik Analitik

- Skor 4: notebook berjalan dari runtime bersih, validasi lengkap, keluaran tersimpan, dan interpretasi tepat serta kritis.
- Skor 3: notebook berjalan dan keluaran benar, tetapi dokumentasi, kasus uji, atau interpretasi belum lengkap.
- Skor 2: notebook hanya berjalan setelah intervensi manual atau masih bergantung pada state lama; validasi terbatas.
- Skor 1: kode belum berjalan konsisten, sumber kesalahan belum diidentifikasi, dan bukti hasil tidak tersedia.

Pertanyaan Refleksi

Jelaskan perubahan paling penting yang Anda lakukan setelah menemukan kesalahan. Sebutkan bukti yang menunjukkan proses berhasil, bagian yang masih memerlukan verifikasi eksternal, dan satu keputusan desain notebook yang meningkatkan reproduksibilitas. Refleksi harus merujuk pada sel, pesan, tabel, atau berkas tertentu, bukan hanya menyatakan bahwa tugas telah selesai.

F.12 Menghitung sudut ikatan dari koordinat Cartesian

Skenario

Anda diminta menyelesaikan aktivitas 'Menghitung sudut ikatan dari koordinat Cartesian' dalam sebuah notebook Google Colab yang dapat dijalankan dari awal sampai akhir.

Notebook harus memuat identitas kegiatan, tujuan, teori singkat, parameter, kode, keluaran, validasi, interpretasi, dan refleksi. Seluruh variabel penting harus diberi nama yang bermakna. Bila aktivitas melibatkan data atau kimia komputasi, satuan, metode, basis set, muatan, spin, dan status konvergensi harus dicatat secara eksplisit.

Tugas yang Harus Dikerjakan

1066. Rancang alur input-proses-validasi-ekspor untuk menghitung sudut ikatan dari koordinat cartesian.
1067. Sediakan minimal satu kasus uji benar dan satu kasus uji salah yang menghasilkan pesan kesalahan informatif.
1068. Jalankan notebook setelah Restart session menggunakan Run all, kemudian simpan bukti keluaran utama.
1069. Ekspor hasil dalam paling sedikit dua format yang sesuai, misalnya CSV dan JSON, atau XYZ dan gambar.
1070. Tuliskan interpretasi yang membedakan fakta komputasi, kesimpulan, dan keterbatasan.

Kriteria Produk

Produk dinyatakan lengkap apabila tidak bergantung pada variabel tersembunyi, seluruh pustaka dipasang atau diperiksa, parameter dapat diubah pada satu tempat, pesan kesalahan tidak disembunyikan, dan artefak dapat dibuka kembali. Nama berkas harus informatif dan tidak menimpa hasil sebelumnya tanpa peringatan. Grafik harus memiliki label dan satuan, sedangkan tabel harus mempunyai nama kolom yang jelas.

Rubrik Analitik

- Skor 4: notebook berjalan dari runtime bersih, validasi lengkap, keluaran tersimpan, dan interpretasi tepat serta kritis.
- Skor 3: notebook berjalan dan keluaran benar, tetapi dokumentasi, kasus uji, atau interpretasi belum lengkap.
- Skor 2: notebook hanya berjalan setelah intervensi manual atau masih bergantung pada state lama; validasi terbatas.
- Skor 1: kode belum berjalan konsisten, sumber kesalahan belum diidentifikasi, dan bukti hasil tidak tersedia.

Pertanyaan Refleksi

Jelaskan perubahan paling penting yang Anda lakukan setelah menemukan kesalahan. Sebutkan bukti yang menunjukkan proses berhasil, bagian yang masih memerlukan verifikasi eksternal, dan satu keputusan desain notebook yang meningkatkan reproduksibilitas. Refleksi harus merujuk pada sel, pesan, tabel, atau berkas tertentu, bukan hanya menyatakan bahwa tugas telah selesai.

F.13 Mendeteksi NaN dan inf pada data energi

Skenario

Anda diminta menyelesaikan aktivitas 'Mendeteksi NaN dan inf pada data energi' dalam sebuah notebook Google Colab yang dapat dijalankan dari awal sampai akhir. Notebook harus memuat identitas kegiatan, tujuan, teori singkat, parameter, kode, keluaran, validasi, interpretasi, dan refleksi. Seluruh variabel penting harus diberi nama yang bermakna. Bila aktivitas melibatkan data atau kimia komputasi, satuan, metode, basis set, muatan, spin, dan status konvergensi harus dicatat secara eksplisit.

Tugas yang Harus Dikerjakan

1071. Rancang alur input-proses-validasi-ekspor untuk mendeteksi nan dan inf pada data energi.
1072. Sediakan minimal satu kasus uji benar dan satu kasus uji salah yang menghasilkan pesan kesalahan informatif.
1073. Jalankan notebook setelah Restart session menggunakan Run all, kemudian simpan bukti keluaran utama.
1074. Ekspor hasil dalam paling sedikit dua format yang sesuai, misalnya CSV dan JSON, atau XYZ dan gambar.
1075. Tuliskan interpretasi yang membedakan fakta komputasi, kesimpulan, dan keterbatasan.

Kriteria Produk

Produk dinyatakan lengkap apabila tidak bergantung pada variabel tersembunyi, seluruh pustaka dipasang atau diperiksa, parameter dapat diubah pada satu tempat, pesan kesalahan tidak disembunyikan, dan artefak dapat dibuka kembali. Nama berkas harus informatif dan tidak menimpa hasil sebelumnya tanpa peringatan. Grafik harus memiliki label dan satuan, sedangkan tabel harus mempunyai nama kolom yang jelas.

Rubrik Analitik

- Skor 4: notebook berjalan dari runtime bersih, validasi lengkap, keluaran tersimpan, dan interpretasi tepat serta kritis.
- Skor 3: notebook berjalan dan keluaran benar, tetapi dokumentasi, kasus uji, atau interpretasi belum lengkap.
- Skor 2: notebook hanya berjalan setelah intervensi manual atau masih bergantung pada state lama; validasi terbatas.
- Skor 1: kode belum berjalan konsisten, sumber kesalahan belum diidentifikasi, dan bukti hasil tidak tersedia.

Pertanyaan Refleksi

Jelaskan perubahan paling penting yang Anda lakukan setelah menemukan kesalahan. Sebutkan bukti yang menunjukkan proses berhasil, bagian yang masih memerlukan verifikasi eksternal, dan satu keputusan desain notebook yang meningkatkan reproduksibilitas. Refleksi harus merujuk pada sel, pesan, tabel, atau berkas tertentu, bukan hanya menyatakan bahwa tugas telah selesai.

F.14 Membandingkan loop Python dan operasi vektor

Skenario

Anda diminta menyelesaikan aktivitas 'Membandingkan loop Python dan operasi vektor' dalam sebuah notebook Google Colab yang dapat dijalankan dari awal sampai akhir. Notebook harus memuat identitas kegiatan, tujuan, teori singkat, parameter, kode, keluaran, validasi, interpretasi, dan refleksi. Seluruh variabel penting harus diberi nama yang bermakna. Bila aktivitas melibatkan data atau kimia komputasi, satuan, metode, basis set, muatan, spin, dan status konvergensi harus dicatat secara eksplisit.

Tugas yang Harus Dikerjakan

1076. Rancang alur input-proses-validasi-ekspor untuk membandingkan loop python dan operasi vektor.
1077. Sediakan minimal satu kasus uji benar dan satu kasus uji salah yang menghasilkan pesan kesalahan informatif.
1078. Jalankan notebook setelah Restart session menggunakan Run all, kemudian simpan bukti keluaran utama.
1079. Ekspor hasil dalam paling sedikit dua format yang sesuai, misalnya CSV dan JSON, atau XYZ dan gambar.
1080. Tuliskan interpretasi yang membedakan fakta komputasi, kesimpulan, dan keterbatasan.

Kriteria Produk

Produk dinyatakan lengkap apabila tidak bergantung pada variabel tersembunyi, seluruh pustaka dipasang atau diperiksa, parameter dapat diubah pada satu tempat, pesan kesalahan tidak disembunyikan, dan artefak dapat dibuka kembali. Nama berkas harus informatif dan tidak menimpa hasil sebelumnya tanpa peringatan. Grafik harus memiliki label dan satuan, sedangkan tabel harus mempunyai nama kolom yang jelas.

Rubrik Analitik

- Skor 4: notebook berjalan dari runtime bersih, validasi lengkap, keluaran tersimpan, dan interpretasi tepat serta kritis.
- Skor 3: notebook berjalan dan keluaran benar, tetapi dokumentasi, kasus uji, atau interpretasi belum lengkap.

- Skor 2: notebook hanya berjalan setelah intervensi manual atau masih bergantung pada state lama; validasi terbatas.
- Skor 1: kode belum berjalan konsisten, sumber kesalahan belum diidentifikasi, dan bukti hasil tidak tersedia.

Pertanyaan Refleksi

Jelaskan perubahan paling penting yang Anda lakukan setelah menemukan kesalahan. Sebutkan bukti yang menunjukkan proses berhasil, bagian yang masih memerlukan verifikasi eksternal, dan satu keputusan desain notebook yang meningkatkan reproduksibilitas. Refleksi harus merujuk pada sel, pesan, tabel, atau berkas tertentu, bukan hanya menyatakan bahwa tugas telah selesai.

F.15 Menyelesaikan sistem persamaan linear sederhana

Skenario

Anda diminta menyelesaikan aktivitas 'Menyelesaikan sistem persamaan linear sederhana' dalam sebuah notebook Google Colab yang dapat dijalankan dari awal sampai akhir. Notebook harus memuat identitas kegiatan, tujuan, teori singkat, parameter, kode, keluaran, validasi, interpretasi, dan refleksi. Seluruh variabel penting harus diberi nama yang bermakna. Bila aktivitas melibatkan data atau kimia komputasi, satuan, metode, basis set, muatan, spin, dan status konvergensi harus dicatat secara eksplisit.

Tugas yang Harus Dikerjakan

1081. Rancang alur input-proses-validasi-ekspor untuk menyelesaikan sistem persamaan linear sederhana.
1082. Sediakan minimal satu kasus uji benar dan satu kasus uji salah yang menghasilkan pesan kesalahan informatif.
1083. Jalankan notebook setelah Restart session menggunakan Run all, kemudian simpan bukti keluaran utama.
1084. Ekspor hasil dalam paling sedikit dua format yang sesuai, misalnya CSV dan JSON, atau XYZ dan gambar.
1085. Tuliskan interpretasi yang membedakan fakta komputasi, kesimpulan, dan keterbatasan.

Kriteria Produk

Produk dinyatakan lengkap apabila tidak bergantung pada variabel tersembunyi, seluruh pustaka dipasang atau diperiksa, parameter dapat diubah pada satu tempat, pesan kesalahan tidak disembunyikan, dan artefak dapat dibuka kembali. Nama berkas harus informatif dan tidak menimpa hasil sebelumnya tanpa peringatan. Grafik harus memiliki label dan satuan, sedangkan tabel harus mempunyai nama kolom yang jelas.

Rubrik Analitik

- Skor 4: notebook berjalan dari runtime bersih, validasi lengkap, keluaran tersimpan, dan interpretasi tepat serta kritis.
- Skor 3: notebook berjalan dan keluaran benar, tetapi dokumentasi, kasus uji, atau interpretasi belum lengkap.
- Skor 2: notebook hanya berjalan setelah intervensi manual atau masih bergantung pada state lama; validasi terbatas.
- Skor 1: kode belum berjalan konsisten, sumber kesalahan belum diidentifikasi, dan bukti hasil tidak tersedia.

Pertanyaan Refleksi

Jelaskan perubahan paling penting yang Anda lakukan setelah menemukan kesalahan. Sebutkan bukti yang menunjukkan proses berhasil, bagian yang masih memerlukan verifikasi eksternal, dan satu keputusan desain notebook yang meningkatkan reproduksibilitas. Refleksi harus merujuk pada sel, pesan, tabel, atau berkas tertentu, bukan hanya menyatakan bahwa tugas telah selesai.

F.16 Menghitung eigenvalue matriks simetris

Skenario

Anda diminta menyelesaikan aktivitas 'Menghitung eigenvalue matriks simetris' dalam sebuah notebook Google Colab yang dapat dijalankan dari awal sampai akhir. Notebook harus memuat identitas kegiatan, tujuan, teori singkat, parameter, kode, keluaran, validasi, interpretasi, dan refleksi. Seluruh variabel penting harus diberi nama yang bermakna. Bila aktivitas melibatkan data atau kimia komputasi, satuan, metode, basis set, muatan, spin, dan status konvergensi harus dicatat secara eksplisit.

Tugas yang Harus Dikerjakan

1086. Rancang alur input-proses-validasi-ekspor untuk menghitung eigenvalue matriks simetris.
1087. Sediakan minimal satu kasus uji benar dan satu kasus uji salah yang menghasilkan pesan kesalahan informatif.
1088. Jalankan notebook setelah Restart session menggunakan Run all, kemudian simpan bukti keluaran utama.
1089. Ekspor hasil dalam paling sedikit dua format yang sesuai, misalnya CSV dan JSON, atau XYZ dan gambar.
1090. Tuliskan interpretasi yang membedakan fakta komputasi, kesimpulan, dan keterbatasan.

Kriteria Produk

Produk dinyatakan lengkap apabila tidak bergantung pada variabel tersembunyi, seluruh pustaka dipasang atau diperiksa, parameter dapat diubah pada satu tempat, pesan kesalahan tidak disembunyikan, dan artefak dapat dibuka kembali. Nama berkas harus informatif dan tidak menimpa hasil sebelumnya tanpa peringatan. Grafik harus memiliki label dan satuan, sedangkan tabel harus mempunyai nama kolom yang jelas.

Rubrik Analitik

- Skor 4: notebook berjalan dari runtime bersih, validasi lengkap, keluaran tersimpan, dan interpretasi tepat serta kritis.
- Skor 3: notebook berjalan dan keluaran benar, tetapi dokumentasi, kasus uji, atau interpretasi belum lengkap.
- Skor 2: notebook hanya berjalan setelah intervensi manual atau masih bergantung pada state lama; validasi terbatas.
- Skor 1: kode belum berjalan konsisten, sumber kesalahan belum diidentifikasi, dan bukti hasil tidak tersedia.

Pertanyaan Refleksi

Jelaskan perubahan paling penting yang Anda lakukan setelah menemukan kesalahan. Sebutkan bukti yang menunjukkan proses berhasil, bagian yang masih memerlukan verifikasi eksternal, dan satu keputusan desain notebook yang meningkatkan reproduksibilitas. Refleksi harus merujuk pada sel, pesan, tabel, atau berkas tertentu, bukan hanya menyatakan bahwa tugas telah selesai.

F.17 Menginterpolasi kurva energi

Skenario

Anda diminta menyelesaikan aktivitas 'Menginterpolasi kurva energi' dalam sebuah notebook Google Colab yang dapat dijalankan dari awal sampai akhir. Notebook harus memuat identitas kegiatan, tujuan, teori singkat, parameter, kode, keluaran, validasi, interpretasi, dan refleksi. Seluruh variabel penting harus diberi nama yang bermakna. Bila aktivitas melibatkan data atau kimia komputasi, satuan, metode, basis set, muatan, spin, dan status konvergensi harus dicatat secara eksplisit.

Tugas yang Harus Dikerjakan

1091. Rancang alur input-proses-validasi-ekspor untuk menginterpolasi kurva energi.
1092. Sediakan minimal satu kasus uji benar dan satu kasus uji salah yang menghasilkan pesan kesalahan informatif.
1093. Jalankan notebook setelah Restart session menggunakan Run all, kemudian simpan bukti keluaran utama.

1094. Ekspor hasil dalam paling sedikit dua format yang sesuai, misalnya CSV dan JSON, atau XYZ dan gambar.
1095. Tuliskan interpretasi yang membedakan fakta komputasi, kesimpulan, dan keterbatasan.

Kriteria Produk

Produk dinyatakan lengkap apabila tidak bergantung pada variabel tersembunyi, seluruh pustaka dipasang atau diperiksa, parameter dapat diubah pada satu tempat, pesan kesalahan tidak disembunyikan, dan artefak dapat dibuka kembali. Nama berkas harus informatif dan tidak menimpa hasil sebelumnya tanpa peringatan. Grafik harus memiliki label dan satuan, sedangkan tabel harus mempunyai nama kolom yang jelas.

Rubrik Analitik

- Skor 4: notebook berjalan dari runtime bersih, validasi lengkap, keluaran tersimpan, dan interpretasi tepat serta kritis.
- Skor 3: notebook berjalan dan keluaran benar, tetapi dokumentasi, kasus uji, atau interpretasi belum lengkap.
- Skor 2: notebook hanya berjalan setelah intervensi manual atau masih bergantung pada state lama; validasi terbatas.
- Skor 1: kode belum berjalan konsisten, sumber kesalahan belum diidentifikasi, dan bukti hasil tidak tersedia.

Pertanyaan Refleksi

Jelaskan perubahan paling penting yang Anda lakukan setelah menemukan kesalahan. Sebutkan bukti yang menunjukkan proses berhasil, bagian yang masih memerlukan verifikasi eksternal, dan satu keputusan desain notebook yang meningkatkan reproduksibilitas. Refleksi harus merujuk pada sel, pesan, tabel, atau berkas tertentu, bukan hanya menyatakan bahwa tugas telah selesai.

F.18 Mengestimasi turunan numerik

Skenario

Anda diminta menyelesaikan aktivitas 'Mengestimasi turunan numerik' dalam sebuah notebook Google Colab yang dapat dijalankan dari awal sampai akhir. Notebook harus memuat identitas kegiatan, tujuan, teori singkat, parameter, kode, keluaran, validasi, interpretasi, dan refleksi. Seluruh variabel penting harus diberi nama yang bermakna. Bila aktivitas melibatkan data atau kimia komputasi, satuan, metode, basis set, muatan, spin, dan status konvergensi harus dicatat secara eksplisit.

Tugas yang Harus Dikerjakan

1096. Rancang alur input-proses-validasi-ekspor untuk mengestimasi turunan numerik.

1097. Sediakan minimal satu kasus uji benar dan satu kasus uji salah yang menghasilkan pesan kesalahan informatif.
1098. Jalankan notebook setelah Restart session menggunakan Run all, kemudian simpan bukti keluaran utama.
1099. Ekspor hasil dalam paling sedikit dua format yang sesuai, misalnya CSV dan JSON, atau XYZ dan gambar.
1100. Tuliskan interpretasi yang membedakan fakta komputasi, kesimpulan, dan keterbatasan.

Kriteria Produk

Produk dinyatakan lengkap apabila tidak bergantung pada variabel tersembunyi, seluruh pustaka dipasang atau diperiksa, parameter dapat diubah pada satu tempat, pesan kesalahan tidak disembunyikan, dan artefak dapat dibuka kembali. Nama berkas harus informatif dan tidak menimpa hasil sebelumnya tanpa peringatan. Grafik harus memiliki label dan satuan, sedangkan tabel harus mempunyai nama kolom yang jelas.

Rubrik Analitik

- Skor 4: notebook berjalan dari runtime bersih, validasi lengkap, keluaran tersimpan, dan interpretasi tepat serta kritis.
- Skor 3: notebook berjalan dan keluaran benar, tetapi dokumentasi, kasus uji, atau interpretasi belum lengkap.
- Skor 2: notebook hanya berjalan setelah intervensi manual atau masih bergantung pada state lama; validasi terbatas.
- Skor 1: kode belum berjalan konsisten, sumber kesalahan belum diidentifikasi, dan bukti hasil tidak tersedia.

Pertanyaan Refleksi

Jelaskan perubahan paling penting yang Anda lakukan setelah menemukan kesalahan. Sebutkan bukti yang menunjukkan proses berhasil, bagian yang masih memerlukan verifikasi eksternal, dan satu keputusan desain notebook yang meningkatkan reproduksibilitas. Refleksi harus merujuk pada sel, pesan, tabel, atau berkas tertentu, bukan hanya menyatakan bahwa tugas telah selesai.

F.19 Mengintegalkan spektrum diskret

Skenario

Anda diminta menyelesaikan aktivitas 'Mengintegalkan spektrum diskret' dalam sebuah notebook Google Colab yang dapat dijalankan dari awal sampai akhir. Notebook harus memuat identitas kegiatan, tujuan, teori singkat, parameter, kode, keluaran, validasi, interpretasi, dan refleksi. Seluruh variabel penting harus diberi nama yang bermakna. Bila aktivitas melibatkan data atau kimia komputasi, satuan, metode, basis set, muatan, spin, dan status konvergensi harus dicatat secara eksplisit.

Tugas yang Harus Dikerjakan

1101. Rancang alur input-proses-validasi-ekspor untuk mengintegrasikan spektrum diskret.
1102. Sediakan minimal satu kasus uji benar dan satu kasus uji salah yang menghasilkan pesan kesalahan informatif.
1103. Jalankan notebook setelah Restart session menggunakan Run all, kemudian simpan bukti keluaran utama.
1104. Ekspor hasil dalam paling sedikit dua format yang sesuai, misalnya CSV dan JSON, atau XYZ dan gambar.
1105. Tuliskan interpretasi yang membedakan fakta komputasi, kesimpulan, dan keterbatasan.

Kriteria Produk

Produk dinyatakan lengkap apabila tidak bergantung pada variabel tersembunyi, seluruh pustaka dipasang atau diperiksa, parameter dapat diubah pada satu tempat, pesan kesalahan tidak disembunyikan, dan artefak dapat dibuka kembali. Nama berkas harus informatif dan tidak menimpa hasil sebelumnya tanpa peringatan. Grafik harus memiliki label dan satuan, sedangkan tabel harus mempunyai nama kolom yang jelas.

Rubrik Analitik

- Skor 4: notebook berjalan dari runtime bersih, validasi lengkap, keluaran tersimpan, dan interpretasi tepat serta kritis.
- Skor 3: notebook berjalan dan keluaran benar, tetapi dokumentasi, kasus uji, atau interpretasi belum lengkap.
- Skor 2: notebook hanya berjalan setelah intervensi manual atau masih bergantung pada state lama; validasi terbatas.
- Skor 1: kode belum berjalan konsisten, sumber kesalahan belum diidentifikasi, dan bukti hasil tidak tersedia.

Pertanyaan Refleksi

Jelaskan perubahan paling penting yang Anda lakukan setelah menemukan kesalahan. Sebutkan bukti yang menunjukkan proses berhasil, bagian yang masih memerlukan verifikasi eksternal, dan satu keputusan desain notebook yang meningkatkan reproduksibilitas. Refleksi harus merujuk pada sel, pesan, tabel, atau berkas tertentu, bukan hanya menyatakan bahwa tugas telah selesai.

F.20 Menyimpan dan memuat array dalam NPY dan CSV

Skenario

Anda diminta menyelesaikan aktivitas 'Menyimpan dan memuat array dalam NPY dan CSV' dalam sebuah notebook Google Colab yang dapat dijalankan dari awal sampai

akhir. Notebook harus memuat identitas kegiatan, tujuan, teori singkat, parameter, kode, keluaran, validasi, interpretasi, dan refleksi. Seluruh variabel penting harus diberi nama yang bermakna. Bila aktivitas melibatkan data atau kimia komputasi, satuan, metode, basis set, muatan, spin, dan status konvergensi harus dicatat secara eksplisit.

Tugas yang Harus Dikerjakan

1106. Rancang alur input-proses-validasi-ekspor untuk menyimpan dan memuat array dalam npy dan csv.
1107. Sediakan minimal satu kasus uji benar dan satu kasus uji salah yang menghasilkan pesan kesalahan informatif.
1108. Jalankan notebook setelah Restart session menggunakan Run all, kemudian simpan bukti keluaran utama.
1109. Ekspor hasil dalam paling sedikit dua format yang sesuai, misalnya CSV dan JSON, atau XYZ dan gambar.
1110. Tuliskan interpretasi yang membedakan fakta komputasi, kesimpulan, dan keterbatasan.

Kriteria Produk

Produk dinyatakan lengkap apabila tidak bergantung pada variabel tersembunyi, seluruh pustaka dipasang atau diperiksa, parameter dapat diubah pada satu tempat, pesan kesalahan tidak disembunyikan, dan artefak dapat dibuka kembali. Nama berkas harus informatif dan tidak menimpa hasil sebelumnya tanpa peringatan. Grafik harus memiliki label dan satuan, sedangkan tabel harus mempunyai nama kolom yang jelas.

Rubrik Analitik

- Skor 4: notebook berjalan dari runtime bersih, validasi lengkap, keluaran tersimpan, dan interpretasi tepat serta kritis.
- Skor 3: notebook berjalan dan keluaran benar, tetapi dokumentasi, kasus uji, atau interpretasi belum lengkap.
- Skor 2: notebook hanya berjalan setelah intervensi manual atau masih bergantung pada state lama; validasi terbatas.
- Skor 1: kode belum berjalan konsisten, sumber kesalahan belum diidentifikasi, dan bukti hasil tidak tersedia.

Pertanyaan Refleksi

Jelaskan perubahan paling penting yang Anda lakukan setelah menemukan kesalahan. Sebutkan bukti yang menunjukkan proses berhasil, bagian yang masih memerlukan verifikasi eksternal, dan satu keputusan desain notebook yang meningkatkan reproduksibilitas. Refleksi harus merujuk pada sel, pesan, tabel, atau berkas tertentu, bukan hanya menyatakan bahwa tugas telah selesai.

F.3 Pandas

F.21 Membersihkan tabel dengan nilai hilang

Skenario

Anda diminta menyelesaikan aktivitas 'Membersihkan tabel dengan nilai hilang' dalam sebuah notebook Google Colab yang dapat dijalankan dari awal sampai akhir. Notebook harus memuat identitas kegiatan, tujuan, teori singkat, parameter, kode, keluaran, validasi, interpretasi, dan refleksi. Seluruh variabel penting harus diberi nama yang bermakna. Bila aktivitas melibatkan data atau kimia komputasi, satuan, metode, basis set, muatan, spin, dan status konvergensi harus dicatat secara eksplisit.

Tugas yang Harus Dikerjakan

1111. Rancang alur input-proses-validasi-ekspor untuk membersihkan tabel dengan nilai hilang.
1112. Sediakan minimal satu kasus uji benar dan satu kasus uji salah yang menghasilkan pesan kesalahan informatif.
1113. Jalankan notebook setelah Restart session menggunakan Run all, kemudian simpan bukti keluaran utama.
1114. Ekspor hasil dalam paling sedikit dua format yang sesuai, misalnya CSV dan JSON, atau XYZ dan gambar.
1115. Tuliskan interpretasi yang membedakan fakta komputasi, kesimpulan, dan keterbatasan.

Kriteria Produk

Produk dinyatakan lengkap apabila tidak bergantung pada variabel tersembunyi, seluruh pustaka dipasang atau diperiksa, parameter dapat diubah pada satu tempat, pesan kesalahan tidak disembunyikan, dan artefak dapat dibuka kembali. Nama berkas harus informatif dan tidak menimpa hasil sebelumnya tanpa peringatan. Grafik harus memiliki label dan satuan, sedangkan tabel harus mempunyai nama kolom yang jelas.

Rubrik Analitik

- Skor 4: notebook berjalan dari runtime bersih, validasi lengkap, keluaran tersimpan, dan interpretasi tepat serta kritis.
- Skor 3: notebook berjalan dan keluaran benar, tetapi dokumentasi, kasus uji, atau interpretasi belum lengkap.
- Skor 2: notebook hanya berjalan setelah intervensi manual atau masih bergantung pada state lama; validasi terbatas.
- Skor 1: kode belum berjalan konsisten, sumber kesalahan belum diidentifikasi, dan bukti hasil tidak tersedia.

Pertanyaan Refleksi

Jelaskan perubahan paling penting yang Anda lakukan setelah menemukan kesalahan. Sebutkan bukti yang menunjukkan proses berhasil, bagian yang masih memerlukan verifikasi eksternal, dan satu keputusan desain notebook yang meningkatkan reproduksibilitas. Refleksi harus merujuk pada sel, pesan, tabel, atau berkas tertentu, bukan hanya menyatakan bahwa tugas telah selesai.

F.22 Menggabungkan tabel parameter dan hasil

Skenario

Anda diminta menyelesaikan aktivitas 'Menggabungkan tabel parameter dan hasil' dalam sebuah notebook Google Colab yang dapat dijalankan dari awal sampai akhir. Notebook harus memuat identitas kegiatan, tujuan, teori singkat, parameter, kode, keluaran, validasi, interpretasi, dan refleksi. Seluruh variabel penting harus diberi nama yang bermakna. Bila aktivitas melibatkan data atau kimia komputasi, satuan, metode, basis set, muatan, spin, dan status konvergensi harus dicatat secara eksplisit.

Tugas yang Harus Dikerjakan

1116. Rancang alur input-proses-validasi-ekspor untuk menggabungkan tabel parameter dan hasil.
1117. Sediakan minimal satu kasus uji benar dan satu kasus uji salah yang menghasilkan pesan kesalahan informatif.
1118. Jalankan notebook setelah Restart session menggunakan Run all, kemudian simpan bukti keluaran utama.
1119. Ekspor hasil dalam paling sedikit dua format yang sesuai, misalnya CSV dan JSON, atau XYZ dan gambar.
1120. Tuliskan interpretasi yang membedakan fakta komputasi, kesimpulan, dan keterbatasan.

Kriteria Produk

Produk dinyatakan lengkap apabila tidak bergantung pada variabel tersembunyi, seluruh pustaka dipasang atau diperiksa, parameter dapat diubah pada satu tempat, pesan kesalahan tidak disembunyikan, dan artefak dapat dibuka kembali. Nama berkas harus informatif dan tidak menimpa hasil sebelumnya tanpa peringatan. Grafik harus memiliki label dan satuan, sedangkan tabel harus mempunyai nama kolom yang jelas.

Rubrik Analitik

- Skor 4: notebook berjalan dari runtime bersih, validasi lengkap, keluaran tersimpan, dan interpretasi tepat serta kritis.
- Skor 3: notebook berjalan dan keluaran benar, tetapi dokumentasi, kasus uji, atau interpretasi belum lengkap.

- Skor 2: notebook hanya berjalan setelah intervensi manual atau masih bergantung pada state lama; validasi terbatas.
- Skor 1: kode belum berjalan konsisten, sumber kesalahan belum diidentifikasi, dan bukti hasil tidak tersedia.

Pertanyaan Refleksi

Jelaskan perubahan paling penting yang Anda lakukan setelah menemukan kesalahan. Sebutkan bukti yang menunjukkan proses berhasil, bagian yang masih memerlukan verifikasi eksternal, dan satu keputusan desain notebook yang meningkatkan reproduksibilitas. Refleksi harus merujuk pada sel, pesan, tabel, atau berkas tertentu, bukan hanya menyatakan bahwa tugas telah selesai.

F.23 Membuat pivot metode versus basis set

Skenario

Anda diminta menyelesaikan aktivitas 'Membuat pivot metode versus basis set' dalam sebuah notebook Google Colab yang dapat dijalankan dari awal sampai akhir. Notebook harus memuat identitas kegiatan, tujuan, teori singkat, parameter, kode, keluaran, validasi, interpretasi, dan refleksi. Seluruh variabel penting harus diberi nama yang bermakna. Bila aktivitas melibatkan data atau kimia komputasi, satuan, metode, basis set, muatan, spin, dan status konvergensi harus dicatat secara eksplisit.

Tugas yang Harus Dikerjakan

1121. Rancang alur input-proses-validasi-ekspor untuk membuat pivot metode versus basis set.
1122. Sediakan minimal satu kasus uji benar dan satu kasus uji salah yang menghasilkan pesan kesalahan informatif.
1123. Jalankan notebook setelah Restart session menggunakan Run all, kemudian simpan bukti keluaran utama.
1124. Ekspor hasil dalam paling sedikit dua format yang sesuai, misalnya CSV dan JSON, atau XYZ dan gambar.
1125. Tuliskan interpretasi yang membedakan fakta komputasi, kesimpulan, dan keterbatasan.

Kriteria Produk

Produk dinyatakan lengkap apabila tidak bergantung pada variabel tersembunyi, seluruh pustaka dipasang atau diperiksa, parameter dapat diubah pada satu tempat, pesan kesalahan tidak disembunyikan, dan artefak dapat dibuka kembali. Nama berkas harus informatif dan tidak menimpa hasil sebelumnya tanpa peringatan. Grafik harus memiliki label dan satuan, sedangkan tabel harus mempunyai nama kolom yang jelas.

Rubrik Analitik

- Skor 4: notebook berjalan dari runtime bersih, validasi lengkap, keluaran tersimpan, dan interpretasi tepat serta kritis.
- Skor 3: notebook berjalan dan keluaran benar, tetapi dokumentasi, kasus uji, atau interpretasi belum lengkap.
- Skor 2: notebook hanya berjalan setelah intervensi manual atau masih bergantung pada state lama; validasi terbatas.
- Skor 1: kode belum berjalan konsisten, sumber kesalahan belum diidentifikasi, dan bukti hasil tidak tersedia.

Pertanyaan Refleksi

Jelaskan perubahan paling penting yang Anda lakukan setelah menemukan kesalahan. Sebutkan bukti yang menunjukkan proses berhasil, bagian yang masih memerlukan verifikasi eksternal, dan satu keputusan desain notebook yang meningkatkan reproduksibilitas. Refleksi harus merujuk pada sel, pesan, tabel, atau berkas tertentu, bukan hanya menyatakan bahwa tugas telah selesai.

F.24 Mengelompokkan waktu eksekusi per backend

Skenario

Anda diminta menyelesaikan aktivitas 'Mengelompokkan waktu eksekusi per backend' dalam sebuah notebook Google Colab yang dapat dijalankan dari awal sampai akhir. Notebook harus memuat identitas kegiatan, tujuan, teori singkat, parameter, kode, keluaran, validasi, interpretasi, dan refleksi. Seluruh variabel penting harus diberi nama yang bermakna. Bila aktivitas melibatkan data atau kimia komputasi, satuan, metode, basis set, muatan, spin, dan status konvergensi harus dicatat secara eksplisit.

Tugas yang Harus Dikerjakan

1126. Rancang alur input-proses-validasi-ekspor untuk mengelompokkan waktu eksekusi per backend.
1127. Sediakan minimal satu kasus uji benar dan satu kasus uji salah yang menghasilkan pesan kesalahan informatif.
1128. Jalankan notebook setelah Restart session menggunakan Run all, kemudian simpan bukti keluaran utama.
1129. Ekspor hasil dalam paling sedikit dua format yang sesuai, misalnya CSV dan JSON, atau XYZ dan gambar.
1130. Tuliskan interpretasi yang membedakan fakta komputasi, kesimpulan, dan keterbatasan.

Kriteria Produk

Produk dinyatakan lengkap apabila tidak bergantung pada variabel tersembunyi, seluruh pustaka dipasang atau diperiksa, parameter dapat diubah pada satu tempat, pesan kesalahan tidak disembunyikan, dan artefak dapat dibuka kembali. Nama berkas harus informatif dan tidak menimpa hasil sebelumnya tanpa peringatan. Grafik harus memiliki label dan satuan, sedangkan tabel harus mempunyai nama kolom yang jelas.

Rubrik Analitik

- Skor 4: notebook berjalan dari runtime bersih, validasi lengkap, keluaran tersimpan, dan interpretasi tepat serta kritis.
- Skor 3: notebook berjalan dan keluaran benar, tetapi dokumentasi, kasus uji, atau interpretasi belum lengkap.
- Skor 2: notebook hanya berjalan setelah intervensi manual atau masih bergantung pada state lama; validasi terbatas.
- Skor 1: kode belum berjalan konsisten, sumber kesalahan belum diidentifikasi, dan bukti hasil tidak tersedia.

Pertanyaan Refleksi

Jelaskan perubahan paling penting yang Anda lakukan setelah menemukan kesalahan. Sebutkan bukti yang menunjukkan proses berhasil, bagian yang masih memerlukan verifikasi eksternal, dan satu keputusan desain notebook yang meningkatkan reproduksibilitas. Refleksi harus merujuk pada sel, pesan, tabel, atau berkas tertentu, bukan hanya menyatakan bahwa tugas telah selesai.

F.25 Mendeteksi duplikat job_id

Skenario

Anda diminta menyelesaikan aktivitas 'Mendeteksi duplikat job_id' dalam sebuah notebook Google Colab yang dapat dijalankan dari awal sampai akhir. Notebook harus memuat identitas kegiatan, tujuan, teori singkat, parameter, kode, keluaran, validasi, interpretasi, dan refleksi. Seluruh variabel penting harus diberi nama yang bermakna. Bila aktivitas melibatkan data atau kimia komputasi, satuan, metode, basis set, muatan, spin, dan status konvergensi harus dicatat secara eksplisit.

Tugas yang Harus Dikerjakan

1131. Rancang alur input-proses-validasi-ekspor untuk mendeteksi duplikat job_id.
1132. Sediakan minimal satu kasus uji benar dan satu kasus uji salah yang menghasilkan pesan kesalahan informatif.
1133. Jalankan notebook setelah Restart session menggunakan Run all, kemudian simpan bukti keluaran utama.

1134. Ekspor hasil dalam paling sedikit dua format yang sesuai, misalnya CSV dan JSON, atau XYZ dan gambar.
1135. Tuliskan interpretasi yang membedakan fakta komputasi, kesimpulan, dan keterbatasan.

Kriteria Produk

Produk dinyatakan lengkap apabila tidak bergantung pada variabel tersembunyi, seluruh pustaka dipasang atau diperiksa, parameter dapat diubah pada satu tempat, pesan kesalahan tidak disembunyikan, dan artefak dapat dibuka kembali. Nama berkas harus informatif dan tidak menimpa hasil sebelumnya tanpa peringatan. Grafik harus memiliki label dan satuan, sedangkan tabel harus mempunyai nama kolom yang jelas.

Rubrik Analitik

- Skor 4: notebook berjalan dari runtime bersih, validasi lengkap, keluaran tersimpan, dan interpretasi tepat serta kritis.
- Skor 3: notebook berjalan dan keluaran benar, tetapi dokumentasi, kasus uji, atau interpretasi belum lengkap.
- Skor 2: notebook hanya berjalan setelah intervensi manual atau masih bergantung pada state lama; validasi terbatas.
- Skor 1: kode belum berjalan konsisten, sumber kesalahan belum diidentifikasi, dan bukti hasil tidak tersedia.

Pertanyaan Refleksi

Jelaskan perubahan paling penting yang Anda lakukan setelah menemukan kesalahan. Sebutkan bukti yang menunjukkan proses berhasil, bagian yang masih memerlukan verifikasi eksternal, dan satu keputusan desain notebook yang meningkatkan reproduksibilitas. Refleksi harus merujuk pada sel, pesan, tabel, atau berkas tertentu, bukan hanya menyatakan bahwa tugas telah selesai.

F.26 Membuat statistik deskriptif dan outlier

Skenario

Anda diminta menyelesaikan aktivitas 'Membuat statistik deskriptif dan outlier' dalam sebuah notebook Google Colab yang dapat dijalankan dari awal sampai akhir. Notebook harus memuat identitas kegiatan, tujuan, teori singkat, parameter, kode, keluaran, validasi, interpretasi, dan refleksi. Seluruh variabel penting harus diberi nama yang bermakna. Bila aktivitas melibatkan data atau kimia komputasi, satuan, metode, basis set, muatan, spin, dan status konvergensi harus dicatat secara eksplisit.

Tugas yang Harus Dikerjakan

1136. Rancang alur input-proses-validasi-ekspor untuk membuat statistik deskriptif dan outlier.
1137. Sediakan minimal satu kasus uji benar dan satu kasus uji salah yang menghasilkan pesan kesalahan informatif.
1138. Jalankan notebook setelah Restart session menggunakan Run all, kemudian simpan bukti keluaran utama.
1139. Ekspor hasil dalam paling sedikit dua format yang sesuai, misalnya CSV dan JSON, atau XYZ dan gambar.
1140. Tuliskan interpretasi yang membedakan fakta komputasi, kesimpulan, dan keterbatasan.

Kriteria Produk

Produk dinyatakan lengkap apabila tidak bergantung pada variabel tersembunyi, seluruh pustaka dipasang atau diperiksa, parameter dapat diubah pada satu tempat, pesan kesalahan tidak disembunyikan, dan artefak dapat dibuka kembali. Nama berkas harus informatif dan tidak menimpa hasil sebelumnya tanpa peringatan. Grafik harus memiliki label dan satuan, sedangkan tabel harus mempunyai nama kolom yang jelas.

Rubrik Analitik

- Skor 4: notebook berjalan dari runtime bersih, validasi lengkap, keluaran tersimpan, dan interpretasi tepat serta kritis.
- Skor 3: notebook berjalan dan keluaran benar, tetapi dokumentasi, kasus uji, atau interpretasi belum lengkap.
- Skor 2: notebook hanya berjalan setelah intervensi manual atau masih bergantung pada state lama; validasi terbatas.
- Skor 1: kode belum berjalan konsisten, sumber kesalahan belum diidentifikasi, dan bukti hasil tidak tersedia.

Pertanyaan Refleksi

Jelaskan perubahan paling penting yang Anda lakukan setelah menemukan kesalahan. Sebutkan bukti yang menunjukkan proses berhasil, bagian yang masih memerlukan verifikasi eksternal, dan satu keputusan desain notebook yang meningkatkan reproduksibilitas. Refleksi harus merujuk pada sel, pesan, tabel, atau berkas tertentu, bukan hanya menyatakan bahwa tugas telah selesai.

F.27 Membuat laporan CSV dan Excel

Skenario

Anda diminta menyelesaikan aktivitas 'Membuat laporan CSV dan Excel' dalam sebuah notebook Google Colab yang dapat dijalankan dari awal sampai akhir. Notebook harus

memuat identitas kegiatan, tujuan, teori singkat, parameter, kode, keluaran, validasi, interpretasi, dan refleksi. Seluruh variabel penting harus diberi nama yang bermakna. Bila aktivitas melibatkan data atau kimia komputasi, satuan, metode, basis set, muatan, spin, dan status konvergensi harus dicatat secara eksplisit.

Tugas yang Harus Dikerjakan

1141. Rancang alur input-proses-validasi-ekspor untuk membuat laporan csv dan excel.
1142. Sediakan minimal satu kasus uji benar dan satu kasus uji salah yang menghasilkan pesan kesalahan informatif.
1143. Jalankan notebook setelah Restart session menggunakan Run all, kemudian simpan bukti keluaran utama.
1144. Ekspor hasil dalam paling sedikit dua format yang sesuai, misalnya CSV dan JSON, atau XYZ dan gambar.
1145. Tuliskan interpretasi yang membedakan fakta komputasi, kesimpulan, dan keterbatasan.

Kriteria Produk

Produk dinyatakan lengkap apabila tidak bergantung pada variabel tersembunyi, seluruh pustaka dipasang atau diperiksa, parameter dapat diubah pada satu tempat, pesan kesalahan tidak disembunyikan, dan artefak dapat dibuka kembali. Nama berkas harus informatif dan tidak menimpa hasil sebelumnya tanpa peringatan. Grafik harus memiliki label dan satuan, sedangkan tabel harus mempunyai nama kolom yang jelas.

Rubrik Analitik

- Skor 4: notebook berjalan dari runtime bersih, validasi lengkap, keluaran tersimpan, dan interpretasi tepat serta kritis.
- Skor 3: notebook berjalan dan keluaran benar, tetapi dokumentasi, kasus uji, atau interpretasi belum lengkap.
- Skor 2: notebook hanya berjalan setelah intervensi manual atau masih bergantung pada state lama; validasi terbatas.
- Skor 1: kode belum berjalan konsisten, sumber kesalahan belum diidentifikasi, dan bukti hasil tidak tersedia.

Pertanyaan Refleksi

Jelaskan perubahan paling penting yang Anda lakukan setelah menemukan kesalahan. Sebutkan bukti yang menunjukkan proses berhasil, bagian yang masih memerlukan verifikasi eksternal, dan satu keputusan desain notebook yang meningkatkan reproduksibilitas. Refleksi harus merujuk pada sel, pesan, tabel, atau berkas tertentu, bukan hanya menyatakan bahwa tugas telah selesai.

F.28 Membuat grafik dari DataFrame

Skenario

Anda diminta menyelesaikan aktivitas 'Membuat grafik dari DataFrame' dalam sebuah notebook Google Colab yang dapat dijalankan dari awal sampai akhir. Notebook harus memuat identitas kegiatan, tujuan, teori singkat, parameter, kode, keluaran, validasi, interpretasi, dan refleksi. Seluruh variabel penting harus diberi nama yang bermakna. Bila aktivitas melibatkan data atau kimia komputasi, satuan, metode, basis set, muatan, spin, dan status konvergensi harus dicatat secara eksplisit.

Tugas yang Harus Dikerjakan

1146. Rancang alur input-proses-validasi-ekspor untuk membuat grafik dari dataframe.
1147. Sediakan minimal satu kasus uji benar dan satu kasus uji salah yang menghasilkan pesan kesalahan informatif.
1148. Jalankan notebook setelah Restart session menggunakan Run all, kemudian simpan bukti keluaran utama.
1149. Ekspor hasil dalam paling sedikit dua format yang sesuai, misalnya CSV dan JSON, atau XYZ dan gambar.
1150. Tuliskan interpretasi yang membedakan fakta komputasi, kesimpulan, dan keterbatasan.

Kriteria Produk

Produk dinyatakan lengkap apabila tidak bergantung pada variabel tersembunyi, seluruh pustaka dipasang atau diperiksa, parameter dapat diubah pada satu tempat, pesan kesalahan tidak disembunyikan, dan artefak dapat dibuka kembali. Nama berkas harus informatif dan tidak menimpa hasil sebelumnya tanpa peringatan. Grafik harus memiliki label dan satuan, sedangkan tabel harus mempunyai nama kolom yang jelas.

Rubrik Analitik

- Skor 4: notebook berjalan dari runtime bersih, validasi lengkap, keluaran tersimpan, dan interpretasi tepat serta kritis.
- Skor 3: notebook berjalan dan keluaran benar, tetapi dokumentasi, kasus uji, atau interpretasi belum lengkap.
- Skor 2: notebook hanya berjalan setelah intervensi manual atau masih bergantung pada state lama; validasi terbatas.
- Skor 1: kode belum berjalan konsisten, sumber kesalahan belum diidentifikasi, dan bukti hasil tidak tersedia.

Pertanyaan Refleksi

Jelaskan perubahan paling penting yang Anda lakukan setelah menemukan kesalahan. Sebutkan bukti yang menunjukkan proses berhasil, bagian yang masih memerlukan

verifikasi eksternal, dan satu keputusan desain notebook yang meningkatkan reproduksibilitas. Refleksi harus merujuk pada sel, pesan, tabel, atau berkas tertentu, bukan hanya menyatakan bahwa tugas telah selesai.

F.29 Menyusun tabel perbandingan energi relatif

Skenario

Anda diminta menyelesaikan aktivitas 'Menyusun tabel perbandingan energi relatif' dalam sebuah notebook Google Colab yang dapat dijalankan dari awal sampai akhir. Notebook harus memuat identitas kegiatan, tujuan, teori singkat, parameter, kode, keluaran, validasi, interpretasi, dan refleksi. Seluruh variabel penting harus diberi nama yang bermakna. Bila aktivitas melibatkan data atau kimia komputasi, satuan, metode, basis set, muatan, spin, dan status konvergensi harus dicatat secara eksplisit.

Tugas yang Harus Dikerjakan

1151. Rancang alur input-proses-validasi-ekspor untuk menyusun tabel perbandingan energi relatif.
1152. Sediakan minimal satu kasus uji benar dan satu kasus uji salah yang menghasilkan pesan kesalahan informatif.
1153. Jalankan notebook setelah Restart session menggunakan Run all, kemudian simpan bukti keluaran utama.
1154. Ekspor hasil dalam paling sedikit dua format yang sesuai, misalnya CSV dan JSON, atau XYZ dan gambar.
1155. Tuliskan interpretasi yang membedakan fakta komputasi, kesimpulan, dan keterbatasan.

Kriteria Produk

Produk dinyatakan lengkap apabila tidak bergantung pada variabel tersembunyi, seluruh pustaka dipasang atau diperiksa, parameter dapat diubah pada satu tempat, pesan kesalahan tidak disembunyikan, dan artefak dapat dibuka kembali. Nama berkas harus informatif dan tidak menimpa hasil sebelumnya tanpa peringatan. Grafik harus memiliki label dan satuan, sedangkan tabel harus mempunyai nama kolom yang jelas.

Rubrik Analitik

- Skor 4: notebook berjalan dari runtime bersih, validasi lengkap, keluaran tersimpan, dan interpretasi tepat serta kritis.
- Skor 3: notebook berjalan dan keluaran benar, tetapi dokumentasi, kasus uji, atau interpretasi belum lengkap.
- Skor 2: notebook hanya berjalan setelah intervensi manual atau masih bergantung pada state lama; validasi terbatas.
- Skor 1: kode belum berjalan konsisten, sumber kesalahan belum diidentifikasi, dan bukti hasil tidak tersedia.

Pertanyaan Refleksi

Jelaskan perubahan paling penting yang Anda lakukan setelah menemukan kesalahan. Sebutkan bukti yang menunjukkan proses berhasil, bagian yang masih memerlukan verifikasi eksternal, dan satu keputusan desain notebook yang meningkatkan reproduksibilitas. Refleksi harus merujuk pada sel, pesan, tabel, atau berkas tertentu, bukan hanya menyatakan bahwa tugas telah selesai.

F.30 Membuat pipeline baca-bersih-analisis-ekspor

Skenario

Anda diminta menyelesaikan aktivitas 'Membuat pipeline baca-bersih-analisis-ekspor' dalam sebuah notebook Google Colab yang dapat dijalankan dari awal sampai akhir. Notebook harus memuat identitas kegiatan, tujuan, teori singkat, parameter, kode, keluaran, validasi, interpretasi, dan refleksi. Seluruh variabel penting harus diberi nama yang bermakna. Bila aktivitas melibatkan data atau kimia komputasi, satuan, metode, basis set, muatan, spin, dan status konvergensi harus dicatat secara eksplisit.

Tugas yang Harus Dikerjakan

1156. Rancang alur input-proses-validasi-ekspor untuk membuat pipeline baca-bersih-analisis-ekspor.
1157. Sediakan minimal satu kasus uji benar dan satu kasus uji salah yang menghasilkan pesan kesalahan informatif.
1158. Jalankan notebook setelah Restart session menggunakan Run all, kemudian simpan bukti keluaran utama.
1159. Ekspor hasil dalam paling sedikit dua format yang sesuai, misalnya CSV dan JSON, atau XYZ dan gambar.
1160. Tuliskan interpretasi yang membedakan fakta komputasi, kesimpulan, dan keterbatasan.

Kriteria Produk

Produk dinyatakan lengkap apabila tidak bergantung pada variabel tersembunyi, seluruh pustaka dipasang atau diperiksa, parameter dapat diubah pada satu tempat, pesan kesalahan tidak disembunyikan, dan artefak dapat dibuka kembali. Nama berkas harus informatif dan tidak menimpa hasil sebelumnya tanpa peringatan. Grafik harus memiliki label dan satuan, sedangkan tabel harus mempunyai nama kolom yang jelas.

Rubrik Analitik

- Skor 4: notebook berjalan dari runtime bersih, validasi lengkap, keluaran tersimpan, dan interpretasi tepat serta kritis.
- Skor 3: notebook berjalan dan keluaran benar, tetapi dokumentasi, kasus uji, atau interpretasi belum lengkap.

- Skor 2: notebook hanya berjalan setelah intervensi manual atau masih bergantung pada state lama; validasi terbatas.
- Skor 1: kode belum berjalan konsisten, sumber kesalahan belum diidentifikasi, dan bukti hasil tidak tersedia.

Pertanyaan Refleksi

Jelaskan perubahan paling penting yang Anda lakukan setelah menemukan kesalahan. Sebutkan bukti yang menunjukkan proses berhasil, bagian yang masih memerlukan verifikasi eksternal, dan satu keputusan desain notebook yang meningkatkan reproduksibilitas. Refleksi harus merujuk pada sel, pesan, tabel, atau berkas tertentu, bukan hanya menyatakan bahwa tugas telah selesai.

F.4 Visualisasi

F.31 Membuat kurva energi dengan anotasi minimum

Skenario

Anda diminta menyelesaikan aktivitas 'Membuat kurva energi dengan anotasi minimum' dalam sebuah notebook Google Colab yang dapat dijalankan dari awal sampai akhir. Notebook harus memuat identitas kegiatan, tujuan, teori singkat, parameter, kode, keluaran, validasi, interpretasi, dan refleksi. Seluruh variabel penting harus diberi nama yang bermakna. Bila aktivitas melibatkan data atau kimia komputasi, satuan, metode, basis set, muatan, spin, dan status konvergensi harus dicatat secara eksplisit.

Tugas yang Harus Dikerjakan

1161. Rancang alur input-proses-validasi-ekspor untuk membuat kurva energi dengan anotasi minimum.
1162. Sediakan minimal satu kasus uji benar dan satu kasus uji salah yang menghasilkan pesan kesalahan informatif.
1163. Jalankan notebook setelah Restart session menggunakan Run all, kemudian simpan bukti keluaran utama.
1164. Ekspor hasil dalam paling sedikit dua format yang sesuai, misalnya CSV dan JSON, atau XYZ dan gambar.
1165. Tuliskan interpretasi yang membedakan fakta komputasi, kesimpulan, dan keterbatasan.

Kriteria Produk

Produk dinyatakan lengkap apabila tidak bergantung pada variabel tersembunyi, seluruh pustaka dipasang atau diperiksa, parameter dapat diubah pada satu tempat, pesan kesalahan tidak disembunyikan, dan artefak dapat dibuka kembali. Nama berkas harus informatif dan tidak menimpa hasil sebelumnya tanpa peringatan. Grafik harus memiliki label dan satuan, sedangkan tabel harus mempunyai nama kolom yang jelas.

Rubrik Analitik

- Skor 4: notebook berjalan dari runtime bersih, validasi lengkap, keluaran tersimpan, dan interpretasi tepat serta kritis.
- Skor 3: notebook berjalan dan keluaran benar, tetapi dokumentasi, kasus uji, atau interpretasi belum lengkap.
- Skor 2: notebook hanya berjalan setelah intervensi manual atau masih bergantung pada state lama; validasi terbatas.
- Skor 1: kode belum berjalan konsisten, sumber kesalahan belum diidentifikasi, dan bukti hasil tidak tersedia.

Pertanyaan Refleksi

Jelaskan perubahan paling penting yang Anda lakukan setelah menemukan kesalahan. Sebutkan bukti yang menunjukkan proses berhasil, bagian yang masih memerlukan verifikasi eksternal, dan satu keputusan desain notebook yang meningkatkan reproduksibilitas. Refleksi harus merujuk pada sel, pesan, tabel, atau berkas tertentu, bukan hanya menyatakan bahwa tugas telah selesai.

F.32 Membuat residual plot regresi

Skenario

Anda diminta menyelesaikan aktivitas 'Membuat residual plot regresi' dalam sebuah notebook Google Colab yang dapat dijalankan dari awal sampai akhir. Notebook harus memuat identitas kegiatan, tujuan, teori singkat, parameter, kode, keluaran, validasi, interpretasi, dan refleksi. Seluruh variabel penting harus diberi nama yang bermakna. Bila aktivitas melibatkan data atau kimia komputasi, satuan, metode, basis set, muatan, spin, dan status konvergensi harus dicatat secara eksplisit.

Tugas yang Harus Dikerjakan

1166. Rancang alur input-proses-validasi-ekspor untuk membuat residual plot regresi.
1167. Sediakan minimal satu kasus uji benar dan satu kasus uji salah yang menghasilkan pesan kesalahan informatif.
1168. Jalankan notebook setelah Restart session menggunakan Run all, kemudian simpan bukti keluaran utama.
1169. Ekspor hasil dalam paling sedikit dua format yang sesuai, misalnya CSV dan JSON, atau XYZ dan gambar.
1170. Tuliskan interpretasi yang membedakan fakta komputasi, kesimpulan, dan keterbatasan.

Kriteria Produk

Produk dinyatakan lengkap apabila tidak bergantung pada variabel tersembunyi, seluruh pustaka dipasang atau diperiksa, parameter dapat diubah pada satu tempat, pesan

kesalahan tidak disembunyikan, dan artefak dapat dibuka kembali. Nama berkas harus informatif dan tidak menimpa hasil sebelumnya tanpa peringatan. Grafik harus memiliki label dan satuan, sedangkan tabel harus mempunyai nama kolom yang jelas.

Rubrik Analitik

- Skor 4: notebook berjalan dari runtime bersih, validasi lengkap, keluaran tersimpan, dan interpretasi tepat serta kritis.
- Skor 3: notebook berjalan dan keluaran benar, tetapi dokumentasi, kasus uji, atau interpretasi belum lengkap.
- Skor 2: notebook hanya berjalan setelah intervensi manual atau masih bergantung pada state lama; validasi terbatas.
- Skor 1: kode belum berjalan konsisten, sumber kesalahan belum diidentifikasi, dan bukti hasil tidak tersedia.

Pertanyaan Refleksi

Jelaskan perubahan paling penting yang Anda lakukan setelah menemukan kesalahan. Sebutkan bukti yang menunjukkan proses berhasil, bagian yang masih memerlukan verifikasi eksternal, dan satu keputusan desain notebook yang meningkatkan reproduksibilitas. Refleksi harus merujuk pada sel, pesan, tabel, atau berkas tertentu, bukan hanya menyatakan bahwa tugas telah selesai.

F.33 Membandingkan histogram dengan bin berbeda

Skenario

Anda diminta menyelesaikan aktivitas 'Membandingkan histogram dengan bin berbeda' dalam sebuah notebook Google Colab yang dapat dijalankan dari awal sampai akhir. Notebook harus memuat identitas kegiatan, tujuan, teori singkat, parameter, kode, keluaran, validasi, interpretasi, dan refleksi. Seluruh variabel penting harus diberi nama yang bermakna. Bila aktivitas melibatkan data atau kimia komputasi, satuan, metode, basis set, muatan, spin, dan status konvergensi harus dicatat secara eksplisit.

Tugas yang Harus Dikerjakan

1171. Rancang alur input-proses-validasi-ekspor untuk membandingkan histogram dengan bin berbeda.
1172. Sediakan minimal satu kasus uji benar dan satu kasus uji salah yang menghasilkan pesan kesalahan informatif.
1173. Jalankan notebook setelah Restart session menggunakan Run all, kemudian simpan bukti keluaran utama.
1174. Ekspor hasil dalam paling sedikit dua format yang sesuai, misalnya CSV dan JSON, atau XYZ dan gambar.
1175. Tuliskan interpretasi yang membedakan fakta komputasi, kesimpulan, dan keterbatasan.

Kriteria Produk

Produk dinyatakan lengkap apabila tidak bergantung pada variabel tersembunyi, seluruh pustaka dipasang atau diperiksa, parameter dapat diubah pada satu tempat, pesan kesalahan tidak disembunyikan, dan artefak dapat dibuka kembali. Nama berkas harus informatif dan tidak menimpa hasil sebelumnya tanpa peringatan. Grafik harus memiliki label dan satuan, sedangkan tabel harus mempunyai nama kolom yang jelas.

Rubrik Analitik

- Skor 4: notebook berjalan dari runtime bersih, validasi lengkap, keluaran tersimpan, dan interpretasi tepat serta kritis.
- Skor 3: notebook berjalan dan keluaran benar, tetapi dokumentasi, kasus uji, atau interpretasi belum lengkap.
- Skor 2: notebook hanya berjalan setelah intervensi manual atau masih bergantung pada state lama; validasi terbatas.
- Skor 1: kode belum berjalan konsisten, sumber kesalahan belum diidentifikasi, dan bukti hasil tidak tersedia.

Pertanyaan Refleksi

Jelaskan perubahan paling penting yang Anda lakukan setelah menemukan kesalahan. Sebutkan bukti yang menunjukkan proses berhasil, bagian yang masih memerlukan verifikasi eksternal, dan satu keputusan desain notebook yang meningkatkan reproduksibilitas. Refleksi harus merujuk pada sel, pesan, tabel, atau berkas tertentu, bukan hanya menyatakan bahwa tugas telah selesai.

F.34 Membuat heatmap matriks korelasi

Skenario

Anda diminta menyelesaikan aktivitas 'Membuat heatmap matriks korelasi' dalam sebuah notebook Google Colab yang dapat dijalankan dari awal sampai akhir. Notebook harus memuat identitas kegiatan, tujuan, teori singkat, parameter, kode, keluaran, validasi, interpretasi, dan refleksi. Seluruh variabel penting harus diberi nama yang bermakna. Bila aktivitas melibatkan data atau kimia komputasi, satuan, metode, basis set, muatan, spin, dan status konvergensi harus dicatat secara eksplisit.

Tugas yang Harus Dikerjakan

1176. Rancang alur input-proses-validasi-ekspor untuk membuat heatmap matriks korelasi.
1177. Sediakan minimal satu kasus uji benar dan satu kasus uji salah yang menghasilkan pesan kesalahan informatif.
1178. Jalankan notebook setelah Restart session menggunakan Run all, kemudian simpan bukti keluaran utama.

1179. Ekspor hasil dalam paling sedikit dua format yang sesuai, misalnya CSV dan JSON, atau XYZ dan gambar.
1180. Tuliskan interpretasi yang membedakan fakta komputasi, kesimpulan, dan keterbatasan.

Kriteria Produk

Produk dinyatakan lengkap apabila tidak bergantung pada variabel tersembunyi, seluruh pustaka dipasang atau diperiksa, parameter dapat diubah pada satu tempat, pesan kesalahan tidak disembunyikan, dan artefak dapat dibuka kembali. Nama berkas harus informatif dan tidak menimpa hasil sebelumnya tanpa peringatan. Grafik harus memiliki label dan satuan, sedangkan tabel harus mempunyai nama kolom yang jelas.

Rubrik Analitik

- Skor 4: notebook berjalan dari runtime bersih, validasi lengkap, keluaran tersimpan, dan interpretasi tepat serta kritis.
- Skor 3: notebook berjalan dan keluaran benar, tetapi dokumentasi, kasus uji, atau interpretasi belum lengkap.
- Skor 2: notebook hanya berjalan setelah intervensi manual atau masih bergantung pada state lama; validasi terbatas.
- Skor 1: kode belum berjalan konsisten, sumber kesalahan belum diidentifikasi, dan bukti hasil tidak tersedia.

Pertanyaan Refleksi

Jelaskan perubahan paling penting yang Anda lakukan setelah menemukan kesalahan. Sebutkan bukti yang menunjukkan proses berhasil, bagian yang masih memerlukan verifikasi eksternal, dan satu keputusan desain notebook yang meningkatkan reproduksibilitas. Refleksi harus merujuk pada sel, pesan, tabel, atau berkas tertentu, bukan hanya menyatakan bahwa tugas telah selesai.

F.35 Mengekspor grafik ke PNG dan PDF

Skenario

Anda diminta menyelesaikan aktivitas 'Mengekspor grafik ke PNG dan PDF' dalam sebuah notebook Google Colab yang dapat dijalankan dari awal sampai akhir. Notebook harus memuat identitas kegiatan, tujuan, teori singkat, parameter, kode, keluaran, validasi, interpretasi, dan refleksi. Seluruh variabel penting harus diberi nama yang bermakna. Bila aktivitas melibatkan data atau kimia komputasi, satuan, metode, basis set, muatan, spin, dan status konvergensi harus dicatat secara eksplisit.

Tugas yang Harus Dikerjakan

1181. Rancang alur input-proses-validasi-ekspor untuk mengekspor grafik ke png dan pdf.
1182. Sediakan minimal satu kasus uji benar dan satu kasus uji salah yang menghasilkan pesan kesalahan informatif.
1183. Jalankan notebook setelah Restart session menggunakan Run all, kemudian simpan bukti keluaran utama.
1184. Ekspor hasil dalam paling sedikit dua format yang sesuai, misalnya CSV dan JSON, atau XYZ dan gambar.
1185. Tuliskan interpretasi yang membedakan fakta komputasi, kesimpulan, dan keterbatasan.

Kriteria Produk

Produk dinyatakan lengkap apabila tidak bergantung pada variabel tersembunyi, seluruh pustaka dipasang atau diperiksa, parameter dapat diubah pada satu tempat, pesan kesalahan tidak disembunyikan, dan artefak dapat dibuka kembali. Nama berkas harus informatif dan tidak menimpa hasil sebelumnya tanpa peringatan. Grafik harus memiliki label dan satuan, sedangkan tabel harus mempunyai nama kolom yang jelas.

Rubrik Analitik

- Skor 4: notebook berjalan dari runtime bersih, validasi lengkap, keluaran tersimpan, dan interpretasi tepat serta kritis.
- Skor 3: notebook berjalan dan keluaran benar, tetapi dokumentasi, kasus uji, atau interpretasi belum lengkap.
- Skor 2: notebook hanya berjalan setelah intervensi manual atau masih bergantung pada state lama; validasi terbatas.
- Skor 1: kode belum berjalan konsisten, sumber kesalahan belum diidentifikasi, dan bukti hasil tidak tersedia.

Pertanyaan Refleksi

Jelaskan perubahan paling penting yang Anda lakukan setelah menemukan kesalahan. Sebutkan bukti yang menunjukkan proses berhasil, bagian yang masih memerlukan verifikasi eksternal, dan satu keputusan desain notebook yang meningkatkan reproduksibilitas. Refleksi harus merujuk pada sel, pesan, tabel, atau berkas tertentu, bukan hanya menyatakan bahwa tugas telah selesai.

F.5 Google Colab

F.36 Mengelola unggah dan unduh berkas

Skenario

Anda diminta menyelesaikan aktivitas 'Mengelola unggah dan unduh berkas' dalam sebuah notebook Google Colab yang dapat dijalankan dari awal sampai akhir. Notebook harus memuat identitas kegiatan, tujuan, teori singkat, parameter, kode, keluaran, validasi, interpretasi, dan refleksi. Seluruh variabel penting harus diberi nama yang bermakna. Bila aktivitas melibatkan data atau kimia komputasi, satuan, metode, basis set, muatan, spin, dan status konvergensi harus dicatat secara eksplisit.

Tugas yang Harus Dikerjakan

1186. Rancang alur input-proses-validasi-ekspor untuk mengelola unggah dan unduh berkas.
1187. Sediakan minimal satu kasus uji benar dan satu kasus uji salah yang menghasilkan pesan kesalahan informatif.
1188. Jalankan notebook setelah Restart session menggunakan Run all, kemudian simpan bukti keluaran utama.
1189. Ekspor hasil dalam paling sedikit dua format yang sesuai, misalnya CSV dan JSON, atau XYZ dan gambar.
1190. Tuliskan interpretasi yang membedakan fakta komputasi, kesimpulan, dan keterbatasan.

Kriteria Produk

Produk dinyatakan lengkap apabila tidak bergantung pada variabel tersembunyi, seluruh pustaka dipasang atau diperiksa, parameter dapat diubah pada satu tempat, pesan kesalahan tidak disembunyikan, dan artefak dapat dibuka kembali. Nama berkas harus informatif dan tidak menimpa hasil sebelumnya tanpa peringatan. Grafik harus memiliki label dan satuan, sedangkan tabel harus mempunyai nama kolom yang jelas.

Rubrik Analitik

- Skor 4: notebook berjalan dari runtime bersih, validasi lengkap, keluaran tersimpan, dan interpretasi tepat serta kritis.
- Skor 3: notebook berjalan dan keluaran benar, tetapi dokumentasi, kasus uji, atau interpretasi belum lengkap.
- Skor 2: notebook hanya berjalan setelah intervensi manual atau masih bergantung pada state lama; validasi terbatas.
- Skor 1: kode belum berjalan konsisten, sumber kesalahan belum diidentifikasi, dan bukti hasil tidak tersedia.

Pertanyaan Refleksi

Jelaskan perubahan paling penting yang Anda lakukan setelah menemukan kesalahan. Sebutkan bukti yang menunjukkan proses berhasil, bagian yang masih memerlukan verifikasi eksternal, dan satu keputusan desain notebook yang meningkatkan reproduksibilitas. Refleksi harus merujuk pada sel, pesan, tabel, atau berkas tertentu, bukan hanya menyatakan bahwa tugas telah selesai.

F.37 Menyimpan hasil pada Google Drive

Skenario

Anda diminta menyelesaikan aktivitas 'Menyimpan hasil pada Google Drive' dalam sebuah notebook Google Colab yang dapat dijalankan dari awal sampai akhir. Notebook harus memuat identitas kegiatan, tujuan, teori singkat, parameter, kode, keluaran, validasi, interpretasi, dan refleksi. Seluruh variabel penting harus diberi nama yang bermakna. Bila aktivitas melibatkan data atau kimia komputasi, satuan, metode, basis set, muatan, spin, dan status konvergensi harus dicatat secara eksplisit.

Tugas yang Harus Dikerjakan

1191. Rancang alur input-proses-validasi-ekspor untuk menyimpan hasil pada google drive.
1192. Sediakan minimal satu kasus uji benar dan satu kasus uji salah yang menghasilkan pesan kesalahan informatif.
1193. Jalankan notebook setelah Restart session menggunakan Run all, kemudian simpan bukti keluaran utama.
1194. Ekspor hasil dalam paling sedikit dua format yang sesuai, misalnya CSV dan JSON, atau XYZ dan gambar.
1195. Tuliskan interpretasi yang membedakan fakta komputasi, kesimpulan, dan keterbatasan.

Kriteria Produk

Produk dinyatakan lengkap apabila tidak bergantung pada variabel tersembunyi, seluruh pustaka dipasang atau diperiksa, parameter dapat diubah pada satu tempat, pesan kesalahan tidak disembunyikan, dan artefak dapat dibuka kembali. Nama berkas harus informatif dan tidak menimpa hasil sebelumnya tanpa peringatan. Grafik harus memiliki label dan satuan, sedangkan tabel harus mempunyai nama kolom yang jelas.

Rubrik Analitik

- Skor 4: notebook berjalan dari runtime bersih, validasi lengkap, keluaran tersimpan, dan interpretasi tepat serta kritis.
- Skor 3: notebook berjalan dan keluaran benar, tetapi dokumentasi, kasus uji, atau interpretasi belum lengkap.

- Skor 2: notebook hanya berjalan setelah intervensi manual atau masih bergantung pada state lama; validasi terbatas.
- Skor 1: kode belum berjalan konsisten, sumber kesalahan belum diidentifikasi, dan bukti hasil tidak tersedia.

Pertanyaan Refleksi

Jelaskan perubahan paling penting yang Anda lakukan setelah menemukan kesalahan. Sebutkan bukti yang menunjukkan proses berhasil, bagian yang masih memerlukan verifikasi eksternal, dan satu keputusan desain notebook yang meningkatkan reproduksibilitas. Refleksi harus merujuk pada sel, pesan, tabel, atau berkas tertentu, bukan hanya menyatakan bahwa tugas telah selesai.

F.38 Mendeteksi CPU, RAM, disk, dan GPU

Skenario

Anda diminta menyelesaikan aktivitas 'Mendeteksi CPU, RAM, disk, dan GPU' dalam sebuah notebook Google Colab yang dapat dijalankan dari awal sampai akhir. Notebook harus memuat identitas kegiatan, tujuan, teori singkat, parameter, kode, keluaran, validasi, interpretasi, dan refleksi. Seluruh variabel penting harus diberi nama yang bermakna. Bila aktivitas melibatkan data atau kimia komputasi, satuan, metode, basis set, muatan, spin, dan status konvergensi harus dicatat secara eksplisit.

Tugas yang Harus Dikerjakan

1196. Rancang alur input-proses-validasi-ekspor untuk mendeteksi cpu, ram, disk, dan gpu.
1197. Sediakan minimal satu kasus uji benar dan satu kasus uji salah yang menghasilkan pesan kesalahan informatif.
1198. Jalankan notebook setelah Restart session menggunakan Run all, kemudian simpan bukti keluaran utama.
1199. Ekspor hasil dalam paling sedikit dua format yang sesuai, misalnya CSV dan JSON, atau XYZ dan gambar.
1200. Tuliskan interpretasi yang membedakan fakta komputasi, kesimpulan, dan keterbatasan.

Kriteria Produk

Produk dinyatakan lengkap apabila tidak bergantung pada variabel tersembunyi, seluruh pustaka dipasang atau diperiksa, parameter dapat diubah pada satu tempat, pesan kesalahan tidak disembunyikan, dan artefak dapat dibuka kembali. Nama berkas harus informatif dan tidak menimpa hasil sebelumnya tanpa peringatan. Grafik harus memiliki label dan satuan, sedangkan tabel harus mempunyai nama kolom yang jelas.

Rubrik Analitik

- Skor 4: notebook berjalan dari runtime bersih, validasi lengkap, keluaran tersimpan, dan interpretasi tepat serta kritis.
- Skor 3: notebook berjalan dan keluaran benar, tetapi dokumentasi, kasus uji, atau interpretasi belum lengkap.
- Skor 2: notebook hanya berjalan setelah intervensi manual atau masih bergantung pada state lama; validasi terbatas.
- Skor 1: kode belum berjalan konsisten, sumber kesalahan belum diidentifikasi, dan bukti hasil tidak tersedia.

Pertanyaan Refleksi

Jelaskan perubahan paling penting yang Anda lakukan setelah menemukan kesalahan. Sebutkan bukti yang menunjukkan proses berhasil, bagian yang masih memerlukan verifikasi eksternal, dan satu keputusan desain notebook yang meningkatkan reproduksibilitas. Refleksi harus merujuk pada sel, pesan, tabel, atau berkas tertentu, bukan hanya menyatakan bahwa tugas telah selesai.

F.39 Membuat mode auto CPU-GPU

Skenario

Anda diminta menyelesaikan aktivitas 'Membuat mode auto CPU-GPU' dalam sebuah notebook Google Colab yang dapat dijalankan dari awal sampai akhir. Notebook harus memuat identitas kegiatan, tujuan, teori singkat, parameter, kode, keluaran, validasi, interpretasi, dan refleksi. Seluruh variabel penting harus diberi nama yang bermakna. Bila aktivitas melibatkan data atau kimia komputasi, satuan, metode, basis set, muatan, spin, dan status konvergensi harus dicatat secara eksplisit.

Tugas yang Harus Dikerjakan

1201. Rancang alur input-proses-validasi-ekspor untuk membuat mode auto cpu-gpu.
1202. Sediakan minimal satu kasus uji benar dan satu kasus uji salah yang menghasilkan pesan kesalahan informatif.
1203. Jalankan notebook setelah Restart session menggunakan Run all, kemudian simpan bukti keluaran utama.
1204. Ekspor hasil dalam paling sedikit dua format yang sesuai, misalnya CSV dan JSON, atau XYZ dan gambar.
1205. Tuliskan interpretasi yang membedakan fakta komputasi, kesimpulan, dan keterbatasan.

Kriteria Produk

Produk dinyatakan lengkap apabila tidak bergantung pada variabel tersembunyi, seluruh pustaka dipasang atau diperiksa, parameter dapat diubah pada satu tempat, pesan

kesalahan tidak disembunyikan, dan artefak dapat dibuka kembali. Nama berkas harus informatif dan tidak menimpa hasil sebelumnya tanpa peringatan. Grafik harus memiliki label dan satuan, sedangkan tabel harus mempunyai nama kolom yang jelas.

Rubrik Analitik

- Skor 4: notebook berjalan dari runtime bersih, validasi lengkap, keluaran tersimpan, dan interpretasi tepat serta kritis.
- Skor 3: notebook berjalan dan keluaran benar, tetapi dokumentasi, kasus uji, atau interpretasi belum lengkap.
- Skor 2: notebook hanya berjalan setelah intervensi manual atau masih bergantung pada state lama; validasi terbatas.
- Skor 1: kode belum berjalan konsisten, sumber kesalahan belum diidentifikasi, dan bukti hasil tidak tersedia.

Pertanyaan Refleksi

Jelaskan perubahan paling penting yang Anda lakukan setelah menemukan kesalahan. Sebutkan bukti yang menunjukkan proses berhasil, bagian yang masih memerlukan verifikasi eksternal, dan satu keputusan desain notebook yang meningkatkan reproduksibilitas. Refleksi harus merujuk pada sel, pesan, tabel, atau berkas tertentu, bukan hanya menyatakan bahwa tugas telah selesai.

F.40 Merekam versi runtime dan pustaka

Skenario

Anda diminta menyelesaikan aktivitas 'Merekam versi runtime dan pustaka' dalam sebuah notebook Google Colab yang dapat dijalankan dari awal sampai akhir. Notebook harus memuat identitas kegiatan, tujuan, teori singkat, parameter, kode, keluaran, validasi, interpretasi, dan refleksi. Seluruh variabel penting harus diberi nama yang bermakna. Bila aktivitas melibatkan data atau kimia komputasi, satuan, metode, basis set, muatan, spin, dan status konvergensi harus dicatat secara eksplisit.

Tugas yang Harus Dikerjakan

1206. Rancang alur input-proses-validasi-ekspor untuk merekam versi runtime dan pustaka.
1207. Sediakan minimal satu kasus uji benar dan satu kasus uji salah yang menghasilkan pesan kesalahan informatif.
1208. Jalankan notebook setelah Restart session menggunakan Run all, kemudian simpan bukti keluaran utama.
1209. Ekspor hasil dalam paling sedikit dua format yang sesuai, misalnya CSV dan JSON, atau XYZ dan gambar.
1210. Tuliskan interpretasi yang membedakan fakta komputasi, kesimpulan, dan keterbatasan.

Kriteria Produk

Produk dinyatakan lengkap apabila tidak bergantung pada variabel tersembunyi, seluruh pustaka dipasang atau diperiksa, parameter dapat diubah pada satu tempat, pesan kesalahan tidak disembunyikan, dan artefak dapat dibuka kembali. Nama berkas harus informatif dan tidak menimpa hasil sebelumnya tanpa peringatan. Grafik harus memiliki label dan satuan, sedangkan tabel harus mempunyai nama kolom yang jelas.

Rubrik Analitik

- Skor 4: notebook berjalan dari runtime bersih, validasi lengkap, keluaran tersimpan, dan interpretasi tepat serta kritis.
- Skor 3: notebook berjalan dan keluaran benar, tetapi dokumentasi, kasus uji, atau interpretasi belum lengkap.
- Skor 2: notebook hanya berjalan setelah intervensi manual atau masih bergantung pada state lama; validasi terbatas.
- Skor 1: kode belum berjalan konsisten, sumber kesalahan belum diidentifikasi, dan bukti hasil tidak tersedia.

Pertanyaan Refleksi

Jelaskan perubahan paling penting yang Anda lakukan setelah menemukan kesalahan. Sebutkan bukti yang menunjukkan proses berhasil, bagian yang masih memerlukan verifikasi eksternal, dan satu keputusan desain notebook yang meningkatkan reproduksibilitas. Refleksi harus merujuk pada sel, pesan, tabel, atau berkas tertentu, bukan hanya menyatakan bahwa tugas telah selesai.

F.6 Kimia Komputasi

F.41 Menghitung kurva energi H₂

Skenario

Anda diminta menyelesaikan aktivitas 'Menghitung kurva energi H₂' dalam sebuah notebook Google Colab yang dapat dijalankan dari awal sampai akhir. Notebook harus memuat identitas kegiatan, tujuan, teori singkat, parameter, kode, keluaran, validasi, interpretasi, dan refleksi. Seluruh variabel penting harus diberi nama yang bermakna. Bila aktivitas melibatkan data atau kimia komputasi, satuan, metode, basis set, muatan, spin, dan status konvergensi harus dicatat secara eksplisit.

Tugas yang Harus Dikerjakan

1211. Rancang alur input-proses-validasi-ekspor untuk menghitung kurva energi h₂.
1212. Sediakan minimal satu kasus uji benar dan satu kasus uji salah yang menghasilkan pesan kesalahan informatif.

1213. Jalankan notebook setelah Restart session menggunakan Run all, kemudian simpan bukti keluaran utama.
1214. Ekspor hasil dalam paling sedikit dua format yang sesuai, misalnya CSV dan JSON, atau XYZ dan gambar.
1215. Tuliskan interpretasi yang membedakan fakta komputasi, kesimpulan, dan keterbatasan.

Kriteria Produk

Produk dinyatakan lengkap apabila tidak bergantung pada variabel tersembunyi, seluruh pustaka dipasang atau diperiksa, parameter dapat diubah pada satu tempat, pesan kesalahan tidak disembunyikan, dan artefak dapat dibuka kembali. Nama berkas harus informatif dan tidak menimpa hasil sebelumnya tanpa peringatan. Grafik harus memiliki label dan satuan, sedangkan tabel harus mempunyai nama kolom yang jelas.

Rubrik Analitik

- Skor 4: notebook berjalan dari runtime bersih, validasi lengkap, keluaran tersimpan, dan interpretasi tepat serta kritis.
- Skor 3: notebook berjalan dan keluaran benar, tetapi dokumentasi, kasus uji, atau interpretasi belum lengkap.
- Skor 2: notebook hanya berjalan setelah intervensi manual atau masih bergantung pada state lama; validasi terbatas.
- Skor 1: kode belum berjalan konsisten, sumber kesalahan belum diidentifikasi, dan bukti hasil tidak tersedia.

Pertanyaan Refleksi

Jelaskan perubahan paling penting yang Anda lakukan setelah menemukan kesalahan. Sebutkan bukti yang menunjukkan proses berhasil, bagian yang masih memerlukan verifikasi eksternal, dan satu keputusan desain notebook yang meningkatkan reproduksibilitas. Refleksi harus merujuk pada sel, pesan, tabel, atau berkas tertentu, bukan hanya menyatakan bahwa tugas telah selesai.

F.42 Membandingkan basis set H2

Skenario

Anda diminta menyelesaikan aktivitas 'Membandingkan basis set H2' dalam sebuah notebook Google Colab yang dapat dijalankan dari awal sampai akhir. Notebook harus memuat identitas kegiatan, tujuan, teori singkat, parameter, kode, keluaran, validasi, interpretasi, dan refleksi. Seluruh variabel penting harus diberi nama yang bermakna. Bila aktivitas melibatkan data atau kimia komputasi, satuan, metode, basis set, muatan, spin, dan status konvergensi harus dicatat secara eksplisit.

Tugas yang Harus Dikerjakan

1216. Rancang alur input-proses-validasi-ekspor untuk membandingkan basis set h2.
1217. Sediakan minimal satu kasus uji benar dan satu kasus uji salah yang menghasilkan pesan kesalahan informatif.
1218. Jalankan notebook setelah Restart session menggunakan Run all, kemudian simpan bukti keluaran utama.
1219. Ekspor hasil dalam paling sedikit dua format yang sesuai, misalnya CSV dan JSON, atau XYZ dan gambar.
1220. Tuliskan interpretasi yang membedakan fakta komputasi, kesimpulan, dan keterbatasan.

Kriteria Produk

Produk dinyatakan lengkap apabila tidak bergantung pada variabel tersembunyi, seluruh pustaka dipasang atau diperiksa, parameter dapat diubah pada satu tempat, pesan kesalahan tidak disembunyikan, dan artefak dapat dibuka kembali. Nama berkas harus informatif dan tidak menimpa hasil sebelumnya tanpa peringatan. Grafik harus memiliki label dan satuan, sedangkan tabel harus mempunyai nama kolom yang jelas.

Rubrik Analitik

- Skor 4: notebook berjalan dari runtime bersih, validasi lengkap, keluaran tersimpan, dan interpretasi tepat serta kritis.
- Skor 3: notebook berjalan dan keluaran benar, tetapi dokumentasi, kasus uji, atau interpretasi belum lengkap.
- Skor 2: notebook hanya berjalan setelah intervensi manual atau masih bergantung pada state lama; validasi terbatas.
- Skor 1: kode belum berjalan konsisten, sumber kesalahan belum diidentifikasi, dan bukti hasil tidak tersedia.

Pertanyaan Refleksi

Jelaskan perubahan paling penting yang Anda lakukan setelah menemukan kesalahan. Sebutkan bukti yang menunjukkan proses berhasil, bagian yang masih memerlukan verifikasi eksternal, dan satu keputusan desain notebook yang meningkatkan reproduksibilitas. Refleksi harus merujuk pada sel, pesan, tabel, atau berkas tertentu, bukan hanya menyatakan bahwa tugas telah selesai.

F.43 Menghitung energi H₂O dengan HF dan DFT

Skenario

Anda diminta menyelesaikan aktivitas 'Menghitung energi H₂O dengan HF dan DFT' dalam sebuah notebook Google Colab yang dapat dijalankan dari awal sampai akhir. Notebook harus memuat identitas kegiatan, tujuan, teori singkat, parameter, kode,

keluaran, validasi, interpretasi, dan refleksi. Seluruh variabel penting harus diberi nama yang bermakna. Bila aktivitas melibatkan data atau kimia komputasi, satuan, metode, basis set, muatan, spin, dan status konvergensi harus dicatat secara eksplisit.

Tugas yang Harus Dikerjakan

1221. Rancang alur input-proses-validasi-ekspor untuk menghitung energi h₂O dengan hf dan dft.
1222. Sediakan minimal satu kasus uji benar dan satu kasus uji salah yang menghasilkan pesan kesalahan informatif.
1223. Jalankan notebook setelah Restart session menggunakan Run all, kemudian simpan bukti keluaran utama.
1224. Ekspor hasil dalam paling sedikit dua format yang sesuai, misalnya CSV dan JSON, atau XYZ dan gambar.
1225. Tuliskan interpretasi yang membedakan fakta komputasi, kesimpulan, dan keterbatasan.

Kriteria Produk

Produk dinyatakan lengkap apabila tidak bergantung pada variabel tersembunyi, seluruh pustaka dipasang atau diperiksa, parameter dapat diubah pada satu tempat, pesan kesalahan tidak disembunyikan, dan artefak dapat dibuka kembali. Nama berkas harus informatif dan tidak menimpa hasil sebelumnya tanpa peringatan. Grafik harus memiliki label dan satuan, sedangkan tabel harus mempunyai nama kolom yang jelas.

Rubrik Analitik

- Skor 4: notebook berjalan dari runtime bersih, validasi lengkap, keluaran tersimpan, dan interpretasi tepat serta kritis.
- Skor 3: notebook berjalan dan keluaran benar, tetapi dokumentasi, kasus uji, atau interpretasi belum lengkap.
- Skor 2: notebook hanya berjalan setelah intervensi manual atau masih bergantung pada state lama; validasi terbatas.
- Skor 1: kode belum berjalan konsisten, sumber kesalahan belum diidentifikasi, dan bukti hasil tidak tersedia.

Pertanyaan Refleksi

Jelaskan perubahan paling penting yang Anda lakukan setelah menemukan kesalahan. Sebutkan bukti yang menunjukkan proses berhasil, bagian yang masih memerlukan verifikasi eksternal, dan satu keputusan desain notebook yang meningkatkan reproduksibilitas. Refleksi harus merujuk pada sel, pesan, tabel, atau berkas tertentu, bukan hanya menyatakan bahwa tugas telah selesai.

F.44 Mengoptimasi geometri H₂O

Skenario

Anda diminta menyelesaikan aktivitas 'Mengoptimasi geometri H₂O' dalam sebuah notebook Google Colab yang dapat dijalankan dari awal sampai akhir. Notebook harus memuat identitas kegiatan, tujuan, teori singkat, parameter, kode, keluaran, validasi, interpretasi, dan refleksi. Seluruh variabel penting harus diberi nama yang bermakna. Bila aktivitas melibatkan data atau kimia komputasi, satuan, metode, basis set, muatan, spin, dan status konvergensi harus dicatat secara eksplisit.

Tugas yang Harus Dikerjakan

1226. Rancang alur input-proses-validasi-ekspor untuk mengoptimasi geometri h₂o.
1227. Sediakan minimal satu kasus uji benar dan satu kasus uji salah yang menghasilkan pesan kesalahan informatif.
1228. Jalankan notebook setelah Restart session menggunakan Run all, kemudian simpan bukti keluaran utama.
1229. Ekspor hasil dalam paling sedikit dua format yang sesuai, misalnya CSV dan JSON, atau XYZ dan gambar.
1230. Tuliskan interpretasi yang membedakan fakta komputasi, kesimpulan, dan keterbatasan.

Kriteria Produk

Produk dinyatakan lengkap apabila tidak bergantung pada variabel tersembunyi, seluruh pustaka dipasang atau diperiksa, parameter dapat diubah pada satu tempat, pesan kesalahan tidak disembunyikan, dan artefak dapat dibuka kembali. Nama berkas harus informatif dan tidak menimpa hasil sebelumnya tanpa peringatan. Grafik harus memiliki label dan satuan, sedangkan tabel harus mempunyai nama kolom yang jelas.

Rubrik Analitik

- Skor 4: notebook berjalan dari runtime bersih, validasi lengkap, keluaran tersimpan, dan interpretasi tepat serta kritis.
- Skor 3: notebook berjalan dan keluaran benar, tetapi dokumentasi, kasus uji, atau interpretasi belum lengkap.
- Skor 2: notebook hanya berjalan setelah intervensi manual atau masih bergantung pada state lama; validasi terbatas.
- Skor 1: kode belum berjalan konsisten, sumber kesalahan belum diidentifikasi, dan bukti hasil tidak tersedia.

Pertanyaan Refleksi

Jelaskan perubahan paling penting yang Anda lakukan setelah menemukan kesalahan. Sebutkan bukti yang menunjukkan proses berhasil, bagian yang masih memerlukan

verifikasi eksternal, dan satu keputusan desain notebook yang meningkatkan reproduksibilitas. Refleksi harus merujuk pada sel, pesan, tabel, atau berkas tertentu, bukan hanya menyatakan bahwa tugas telah selesai.

F.45 Menghitung panjang dan sudut ikatan

Skenario

Anda diminta menyelesaikan aktivitas 'Menghitung panjang dan sudut ikatan' dalam sebuah notebook Google Colab yang dapat dijalankan dari awal sampai akhir. Notebook harus memuat identitas kegiatan, tujuan, teori singkat, parameter, kode, keluaran, validasi, interpretasi, dan refleksi. Seluruh variabel penting harus diberi nama yang bermakna. Bila aktivitas melibatkan data atau kimia komputasi, satuan, metode, basis set, muatan, spin, dan status konvergensi harus dicatat secara eksplisit.

Tugas yang Harus Dikerjakan

1231. Rancang alur input-proses-validasi-ekspor untuk menghitung panjang dan sudut ikatan.
1232. Sediakan minimal satu kasus uji benar dan satu kasus uji salah yang menghasilkan pesan kesalahan informatif.
1233. Jalankan notebook setelah Restart session menggunakan Run all, kemudian simpan bukti keluaran utama.
1234. Ekspor hasil dalam paling sedikit dua format yang sesuai, misalnya CSV dan JSON, atau XYZ dan gambar.
1235. Tuliskan interpretasi yang membedakan fakta komputasi, kesimpulan, dan keterbatasan.

Kriteria Produk

Produk dinyatakan lengkap apabila tidak bergantung pada variabel tersembunyi, seluruh pustaka dipasang atau diperiksa, parameter dapat diubah pada satu tempat, pesan kesalahan tidak disembunyikan, dan artefak dapat dibuka kembali. Nama berkas harus informatif dan tidak menimpa hasil sebelumnya tanpa peringatan. Grafik harus memiliki label dan satuan, sedangkan tabel harus mempunyai nama kolom yang jelas.

Rubrik Analitik

- Skor 4: notebook berjalan dari runtime bersih, validasi lengkap, keluaran tersimpan, dan interpretasi tepat serta kritis.
- Skor 3: notebook berjalan dan keluaran benar, tetapi dokumentasi, kasus uji, atau interpretasi belum lengkap.
- Skor 2: notebook hanya berjalan setelah intervensi manual atau masih bergantung pada state lama; validasi terbatas.
- Skor 1: kode belum berjalan konsisten, sumber kesalahan belum diidentifikasi, dan bukti hasil tidak tersedia.

Pertanyaan Refleksi

Jelaskan perubahan paling penting yang Anda lakukan setelah menemukan kesalahan. Sebutkan bukti yang menunjukkan proses berhasil, bagian yang masih memerlukan verifikasi eksternal, dan satu keputusan desain notebook yang meningkatkan reproduksibilitas. Refleksi harus merujuk pada sel, pesan, tabel, atau berkas tertentu, bukan hanya menyatakan bahwa tugas telah selesai.

F.46 Menganalisis HOMO, LUMO, dan gap

Skenario

Anda diminta menyelesaikan aktivitas 'Menganalisis HOMO, LUMO, dan gap' dalam sebuah notebook Google Colab yang dapat dijalankan dari awal sampai akhir. Notebook harus memuat identitas kegiatan, tujuan, teori singkat, parameter, kode, keluaran, validasi, interpretasi, dan refleksi. Seluruh variabel penting harus diberi nama yang bermakna. Bila aktivitas melibatkan data atau kimia komputasi, satuan, metode, basis set, muatan, spin, dan status konvergensi harus dicatat secara eksplisit.

Tugas yang Harus Dikerjakan

1236. Rancang alur input-proses-validasi-ekspor untuk menganalisis homo, lumo, dan gap.
1237. Sediakan minimal satu kasus uji benar dan satu kasus uji salah yang menghasilkan pesan kesalahan informatif.
1238. Jalankan notebook setelah Restart session menggunakan Run all, kemudian simpan bukti keluaran utama.
1239. Ekspor hasil dalam paling sedikit dua format yang sesuai, misalnya CSV dan JSON, atau XYZ dan gambar.
1240. Tuliskan interpretasi yang membedakan fakta komputasi, kesimpulan, dan keterbatasan.

Kriteria Produk

Produk dinyatakan lengkap apabila tidak bergantung pada variabel tersembunyi, seluruh pustaka dipasang atau diperiksa, parameter dapat diubah pada satu tempat, pesan kesalahan tidak disembunyikan, dan artefak dapat dibuka kembali. Nama berkas harus informatif dan tidak menimpa hasil sebelumnya tanpa peringatan. Grafik harus memiliki label dan satuan, sedangkan tabel harus mempunyai nama kolom yang jelas.

Rubrik Analitik

- Skor 4: notebook berjalan dari runtime bersih, validasi lengkap, keluaran tersimpan, dan interpretasi tepat serta kritis.
- Skor 3: notebook berjalan dan keluaran benar, tetapi dokumentasi, kasus uji, atau interpretasi belum lengkap.

- Skor 2: notebook hanya berjalan setelah intervensi manual atau masih bergantung pada state lama; validasi terbatas.
- Skor 1: kode belum berjalan konsisten, sumber kesalahan belum diidentifikasi, dan bukti hasil tidak tersedia.

Pertanyaan Refleksi

Jelaskan perubahan paling penting yang Anda lakukan setelah menemukan kesalahan. Sebutkan bukti yang menunjukkan proses berhasil, bagian yang masih memerlukan verifikasi eksternal, dan satu keputusan desain notebook yang meningkatkan reproduksibilitas. Refleksi harus merujuk pada sel, pesan, tabel, atau berkas tertentu, bukan hanya menyatakan bahwa tugas telah selesai.

F.47 Menghitung momen dipol

Skenario

Anda diminta menyelesaikan aktivitas 'Menghitung momen dipol' dalam sebuah notebook Google Colab yang dapat dijalankan dari awal sampai akhir. Notebook harus memuat identitas kegiatan, tujuan, teori singkat, parameter, kode, keluaran, validasi, interpretasi, dan refleksi. Seluruh variabel penting harus diberi nama yang bermakna. Bila aktivitas melibatkan data atau kimia komputasi, satuan, metode, basis set, muatan, spin, dan status konvergensi harus dicatat secara eksplisit.

Tugas yang Harus Dikerjakan

1241. Rancang alur input-proses-validasi-ekspor untuk menghitung momen dipol.
1242. Sediakan minimal satu kasus uji benar dan satu kasus uji salah yang menghasilkan pesan kesalahan informatif.
1243. Jalankan notebook setelah Restart session menggunakan Run all, kemudian simpan bukti keluaran utama.
1244. Ekspor hasil dalam paling sedikit dua format yang sesuai, misalnya CSV dan JSON, atau XYZ dan gambar.
1245. Tuliskan interpretasi yang membedakan fakta komputasi, kesimpulan, dan keterbatasan.

Kriteria Produk

Produk dinyatakan lengkap apabila tidak bergantung pada variabel tersembunyi, seluruh pustaka dipasang atau diperiksa, parameter dapat diubah pada satu tempat, pesan kesalahan tidak disembunyikan, dan artefak dapat dibuka kembali. Nama berkas harus informatif dan tidak menimpa hasil sebelumnya tanpa peringatan. Grafik harus memiliki label dan satuan, sedangkan tabel harus mempunyai nama kolom yang jelas.

Rubrik Analitik

- Skor 4: notebook berjalan dari runtime bersih, validasi lengkap, keluaran tersimpan, dan interpretasi tepat serta kritis.
- Skor 3: notebook berjalan dan keluaran benar, tetapi dokumentasi, kasus uji, atau interpretasi belum lengkap.
- Skor 2: notebook hanya berjalan setelah intervensi manual atau masih bergantung pada state lama; validasi terbatas.
- Skor 1: kode belum berjalan konsisten, sumber kesalahan belum diidentifikasi, dan bukti hasil tidak tersedia.

Pertanyaan Refleksi

Jelaskan perubahan paling penting yang Anda lakukan setelah menemukan kesalahan. Sebutkan bukti yang menunjukkan proses berhasil, bagian yang masih memerlukan verifikasi eksternal, dan satu keputusan desain notebook yang meningkatkan reproduksibilitas. Refleksi harus merujuk pada sel, pesan, tabel, atau berkas tertentu, bukan hanya menyatakan bahwa tugas telah selesai.

F.48 Membuat konformer etanol dengan RDKit

Skenario

Anda diminta menyelesaikan aktivitas 'Membuat konformer etanol dengan RDKit' dalam sebuah notebook Google Colab yang dapat dijalankan dari awal sampai akhir. Notebook harus memuat identitas kegiatan, tujuan, teori singkat, parameter, kode, keluaran, validasi, interpretasi, dan refleksi. Seluruh variabel penting harus diberi nama yang bermakna. Bila aktivitas melibatkan data atau kimia komputasi, satuan, metode, basis set, muatan, spin, dan status konvergensi harus dicatat secara eksplisit.

Tugas yang Harus Dikerjakan

1246. Rancang alur input-proses-validasi-ekspor untuk membuat konformer etanol dengan rdkit.
1247. Sediakan minimal satu kasus uji benar dan satu kasus uji salah yang menghasilkan pesan kesalahan informatif.
1248. Jalankan notebook setelah Restart session menggunakan Run all, kemudian simpan bukti keluaran utama.
1249. Ekspor hasil dalam paling sedikit dua format yang sesuai, misalnya CSV dan JSON, atau XYZ dan gambar.
1250. Tuliskan interpretasi yang membedakan fakta komputasi, kesimpulan, dan keterbatasan.

Kriteria Produk

Produk dinyatakan lengkap apabila tidak bergantung pada variabel tersembunyi, seluruh pustaka dipasang atau diperiksa, parameter dapat diubah pada satu tempat, pesan kesalahan tidak disembunyikan, dan artefak dapat dibuka kembali. Nama berkas harus informatif dan tidak menimpa hasil sebelumnya tanpa peringatan. Grafik harus memiliki label dan satuan, sedangkan tabel harus mempunyai nama kolom yang jelas.

Rubrik Analitik

- Skor 4: notebook berjalan dari runtime bersih, validasi lengkap, keluaran tersimpan, dan interpretasi tepat serta kritis.
- Skor 3: notebook berjalan dan keluaran benar, tetapi dokumentasi, kasus uji, atau interpretasi belum lengkap.
- Skor 2: notebook hanya berjalan setelah intervensi manual atau masih bergantung pada state lama; validasi terbatas.
- Skor 1: kode belum berjalan konsisten, sumber kesalahan belum diidentifikasi, dan bukti hasil tidak tersedia.

Pertanyaan Refleksi

Jelaskan perubahan paling penting yang Anda lakukan setelah menemukan kesalahan. Sebutkan bukti yang menunjukkan proses berhasil, bagian yang masih memerlukan verifikasi eksternal, dan satu keputusan desain notebook yang meningkatkan reproduksibilitas. Refleksi harus merujuk pada sel, pesan, tabel, atau berkas tertentu, bukan hanya menyatakan bahwa tugas telah selesai.

F.49 Memvisualisasikan struktur dengan py3Dmol

Skenario

Anda diminta menyelesaikan aktivitas 'Memvisualisasikan struktur dengan py3Dmol' dalam sebuah notebook Google Colab yang dapat dijalankan dari awal sampai akhir. Notebook harus memuat identitas kegiatan, tujuan, teori singkat, parameter, kode, keluaran, validasi, interpretasi, dan refleksi. Seluruh variabel penting harus diberi nama yang bermakna. Bila aktivitas melibatkan data atau kimia komputasi, satuan, metode, basis set, muatan, spin, dan status konvergensi harus dicatat secara eksplisit.

Tugas yang Harus Dikerjakan

1251. Rancang alur input-proses-validasi-ekspor untuk memvisualisasikan struktur dengan py3dmol.
1252. Sediakan minimal satu kasus uji benar dan satu kasus uji salah yang menghasilkan pesan kesalahan informatif.
1253. Jalankan notebook setelah Restart session menggunakan Run all, kemudian simpan bukti keluaran utama.

1254. Ekspor hasil dalam paling sedikit dua format yang sesuai, misalnya CSV dan JSON, atau XYZ dan gambar.
1255. Tuliskan interpretasi yang membedakan fakta komputasi, kesimpulan, dan keterbatasan.

Kriteria Produk

Produk dinyatakan lengkap apabila tidak bergantung pada variabel tersembunyi, seluruh pustaka dipasang atau diperiksa, parameter dapat diubah pada satu tempat, pesan kesalahan tidak disembunyikan, dan artefak dapat dibuka kembali. Nama berkas harus informatif dan tidak menimpa hasil sebelumnya tanpa peringatan. Grafik harus memiliki label dan satuan, sedangkan tabel harus mempunyai nama kolom yang jelas.

Rubrik Analitik

- Skor 4: notebook berjalan dari runtime bersih, validasi lengkap, keluaran tersimpan, dan interpretasi tepat serta kritis.
- Skor 3: notebook berjalan dan keluaran benar, tetapi dokumentasi, kasus uji, atau interpretasi belum lengkap.
- Skor 2: notebook hanya berjalan setelah intervensi manual atau masih bergantung pada state lama; validasi terbatas.
- Skor 1: kode belum berjalan konsisten, sumber kesalahan belum diidentifikasi, dan bukti hasil tidak tersedia.

Pertanyaan Refleksi

Jelaskan perubahan paling penting yang Anda lakukan setelah menemukan kesalahan. Sebutkan bukti yang menunjukkan proses berhasil, bagian yang masih memerlukan verifikasi eksternal, dan satu keputusan desain notebook yang meningkatkan reproduksibilitas. Refleksi harus merujuk pada sel, pesan, tabel, atau berkas tertentu, bukan hanya menyatakan bahwa tugas telah selesai.

F.50 Menyusun proyek lengkap parasetamol

Skenario

Anda diminta menyelesaikan aktivitas 'Menyusun proyek lengkap parasetamol' dalam sebuah notebook Google Colab yang dapat dijalankan dari awal sampai akhir. Notebook harus memuat identitas kegiatan, tujuan, teori singkat, parameter, kode, keluaran, validasi, interpretasi, dan refleksi. Seluruh variabel penting harus diberi nama yang bermakna. Bila aktivitas melibatkan data atau kimia komputasi, satuan, metode, basis set, muatan, spin, dan status konvergensi harus dicatat secara eksplisit.

Tugas yang Harus Dikerjakan

1256. Rancang alur input-proses-validasi-ekspor untuk menyusun proyek lengkap parasetamol.
1257. Sediakan minimal satu kasus uji benar dan satu kasus uji salah yang menghasilkan pesan kesalahan informatif.
1258. Jalankan notebook setelah Restart session menggunakan Run all, kemudian simpan bukti keluaran utama.
1259. Ekspor hasil dalam paling sedikit dua format yang sesuai, misalnya CSV dan JSON, atau XYZ dan gambar.
1260. Tuliskan interpretasi yang membedakan fakta komputasi, kesimpulan, dan keterbatasan.

Kriteria Produk

Produk dinyatakan lengkap apabila tidak bergantung pada variabel tersembunyi, seluruh pustaka dipasang atau diperiksa, parameter dapat diubah pada satu tempat, pesan kesalahan tidak disembunyikan, dan artefak dapat dibuka kembali. Nama berkas harus informatif dan tidak menimpa hasil sebelumnya tanpa peringatan. Grafik harus memiliki label dan satuan, sedangkan tabel harus mempunyai nama kolom yang jelas.

Rubrik Analitik

- Skor 4: notebook berjalan dari runtime bersih, validasi lengkap, keluaran tersimpan, dan interpretasi tepat serta kritis.
- Skor 3: notebook berjalan dan keluaran benar, tetapi dokumentasi, kasus uji, atau interpretasi belum lengkap.
- Skor 2: notebook hanya berjalan setelah intervensi manual atau masih bergantung pada state lama; validasi terbatas.
- Skor 1: kode belum berjalan konsisten, sumber kesalahan belum diidentifikasi, dan bukti hasil tidak tersedia.

Pertanyaan Refleksi

Jelaskan perubahan paling penting yang Anda lakukan setelah menemukan kesalahan. Sebutkan bukti yang menunjukkan proses berhasil, bagian yang masih memerlukan verifikasi eksternal, dan satu keputusan desain notebook yang meningkatkan reproduksibilitas. Refleksi harus merujuk pada sel, pesan, tabel, atau berkas tertentu, bukan hanya menyatakan bahwa tugas telah selesai.

LAMPIRAN G - 40 SKENARIO TROUBLESHOOTING DAN PEMULIHAN

Troubleshooting yang baik dimulai dari bukti, bukan tebakan. Simpan pesan kesalahan lengkap, identifikasi sel pertama yang gagal, bedakan gejala dari penyebab, dan ubah satu hal pada satu waktu. Setelah perbaikan, ulangi dari runtime bersih agar keberhasilan tidak bergantung pada state lama. Empat puluh skenario berikut merangkum masalah yang sering muncul pada Google Colab, Python, pengelolaan data, RDKit, PySCF, dan penggunaan GPU.

G.01 ModuleNotFoundError

Gejala dan Bukti Awal

Gejala yang terlihat adalah `modulenotfounderror`. Jangan langsung mengubah banyak sel. Salin pesan kesalahan lengkap, catat nomor sel dan waktu kejadian, serta periksa apakah masalah dapat diulang setelah Restart session. Bukti awal yang berguna meliputi versi Python dan paket, direktori kerja, daftar berkas, shape dan dtype data, parameter kimia, status CPU/GPU, serta bagian traceback yang menunjuk baris pertama dari kode pengguna.

Penyebab yang Paling Mungkin

Penyebab utama yang perlu diuji adalah: Paket belum terpasang pada runtime baru. Penyebab tersebut harus diperlakukan sebagai hipotesis. Periksa prasyarat paling sederhana terlebih dahulu, lalu bergerak ke dependensi dan algoritma. Apabila beberapa penyebab mungkin, buat uji minimal yang mengisolasi satu komponen, misalnya hanya mengimpor modul, hanya membaca satu berkas, atau menjalankan molekul kecil.

Langkah Pemulihan

1261. Simpan notebook dan artefak yang masih tersedia sebelum melakukan restart atau instalasi ulang.
1262. Buat uji minimal untuk memastikan gejala dapat direproduksi tanpa sel yang tidak berkaitan.
1263. Jalankan `%pip` pada sel instalasi, periksa nama distribusi, lalu restart bila paket inti berubah.
1264. Jalankan kembali seluruh notebook dari sesi bersih dan bandingkan status serta keluaran dengan catatan sebelum perbaikan.
1265. Dokumentasikan akar masalah, perubahan yang dilakukan, versi, dan tindakan pencegahan.

Pencegahan

Pencegahan dilakukan dengan memusatkan parameter, menggunakan validasi eksplisit, mencatat versi, menyimpan checkpoint, dan menyediakan pesan kesalahan yang spesifik. Notebook yang baik tidak hanya bekerja pada kondisi ideal, tetapi juga gagal secara aman ketika input atau lingkungan tidak memenuhi syarat. Tambahkan satu sel diagnostik pada awal notebook dan satu sel manifest pada akhir proyek.

G.02 ImportError setelah upgrade

Gejala dan Bukti Awal

Gejala yang terlihat adalah importerror setelah upgrade. Jangan langsung mengubah banyak sel. Salin pesan kesalahan lengkap, catat nomor sel dan waktu kejadian, serta periksa apakah masalah dapat diulang setelah Restart session. Bukti awal yang berguna meliputi versi Python dan paket, direktori kerja, daftar berkas, shape dan dtype data, parameter kimia, status CPU/GPU, serta bagian traceback yang menunjuk baris pertama dari kode pengguna.

Penyebab yang Paling Mungkin

Penyebab utama yang perlu diuji adalah: Versi dependensi tidak kompatibel atau modul lama masih berada di memori. Penyebab tersebut harus diperlakukan sebagai hipotesis. Periksa prasyarat paling sederhana terlebih dahulu, lalu bergerak ke dependensi dan algoritma. Apabila beberapa penyebab mungkin, buat uji minimal yang mengisolasi satu komponen, misalnya hanya mengimpor modul, hanya membaca satu berkas, atau menjalankan molekul kecil.

Langkah Pemulihan

1266. Simpan notebook dan artefak yang masih tersedia sebelum melakukan restart atau instalasi ulang.
1267. Buat uji minimal untuk memastikan gejala dapat direproduksi tanpa sel yang tidak berkaitan.
1268. Catat versi, restart session, pasang versi yang kompatibel, dan uji impor minimal.
1269. Jalankan kembali seluruh notebook dari sesi bersih dan bandingkan status serta keluaran dengan catatan sebelum perbaikan.
1270. Dokumentasikan akar masalah, perubahan yang dilakukan, versi, dan tindakan pencegahan.

Pencegahan

Pencegahan dilakukan dengan memusatkan parameter, menggunakan validasi eksplisit, mencatat versi, menyimpan checkpoint, dan menyediakan pesan kesalahan yang spesifik. Notebook yang baik tidak hanya bekerja pada kondisi ideal, tetapi juga gagal secara aman

ketika input atau lingkungan tidak memenuhi syarat. Tambahkan satu sel diagnostik pada awal notebook dan satu sel manifest pada akhir proyek.

G.03 Runtime terputus

Gejala dan Bukti Awal

Gejala yang terlihat adalah runtime terputus. Jangan langsung mengubah banyak sel. Salin pesan kesalahan lengkap, catat nomor sel dan waktu kejadian, serta periksa apakah masalah dapat diulang setelah Restart session. Bukti awal yang berguna meliputi versi Python dan paket, direktori kerja, daftar berkas, shape dan dtype data, parameter kimia, status CPU/GPU, serta bagian traceback yang menunjuk baris pertama dari kode pengguna.

Penyebab yang Paling Mungkin

Penyebab utama yang perlu diuji adalah: Batas sumber daya, idle, jaringan, atau proses berat. Penyebab tersebut harus diperlakukan sebagai hipotesis. Periksa prasyarat paling sederhana terlebih dahulu, lalu bergerak ke dependensi dan algoritma. Apabila beberapa penyebab mungkin, buat uji minimal yang mengisolasi satu komponen, misalnya hanya mengimpor modul, hanya membaca satu berkas, atau menjalankan molekul kecil.

Langkah Pemulihan

1271. Simpan notebook dan artefak yang masih tersedia sebelum melakukan restart atau instalasi ulang.
1272. Buat uji minimal untuk memastikan gejala dapat direproduksi tanpa sel yang tidak berkaitan.
1273. Simpan checkpoint ke Drive, kurangi beban, dan pecah proses menjadi tahap yang dapat dilanjutkan.
1274. Jalankan kembali seluruh notebook dari sesi bersih dan bandingkan status serta keluaran dengan catatan sebelum perbaikan.
1275. Dokumentasikan akar masalah, perubahan yang dilakukan, versi, dan tindakan pencegahan.

Pencegahan

Pencegahan dilakukan dengan memusatkan parameter, menggunakan validasi eksplisit, mencatat versi, menyimpan checkpoint, dan menyediakan pesan kesalahan yang spesifik. Notebook yang baik tidak hanya bekerja pada kondisi ideal, tetapi juga gagal secara aman ketika input atau lingkungan tidak memenuhi syarat. Tambahkan satu sel diagnostik pada awal notebook dan satu sel manifest pada akhir proyek.

G.04 Berkas di /content hilang

Gejala dan Bukti Awal

Gejala yang terlihat adalah berkas di /content hilang. Jangan langsung mengubah banyak sel. Salin pesan kesalahan lengkap, catat nomor sel dan waktu kejadian, serta periksa apakah masalah dapat diulang setelah Restart session. Bukti awal yang berguna meliputi versi Python dan paket, direktori kerja, daftar berkas, shape dan dtype data, parameter kimia, status CPU/GPU, serta bagian traceback yang menunjuk baris pertama dari kode pengguna.

Penyebab yang Paling Mungkin

Penyebab utama yang perlu diuji adalah: Penyimpanan runtime bersifat sementara. Penyebab tersebut harus diperlakukan sebagai hipotesis. Periksa prasyarat paling sederhana terlebih dahulu, lalu bergerak ke dependensi dan algoritma. Apabila beberapa penyebab mungkin, buat uji minimal yang mengisolasi satu komponen, misalnya hanya mengimpor modul, hanya membaca satu berkas, atau menjalankan molekul kecil.

Langkah Pemulihan

1276. Simpan notebook dan artefak yang masih tersedia sebelum melakukan restart atau instalasi ulang.
1277. Buat uji minimal untuk memastikan gejala dapat direproduksi tanpa sel yang tidak berkaitan.
1278. Pulihkan dari Drive atau unggah ulang; tambahkan sel ekspor otomatis.
1279. Jalankan kembali seluruh notebook dari sesi bersih dan bandingkan status serta keluaran dengan catatan sebelum perbaikan.
1280. Dokumentasikan akar masalah, perubahan yang dilakukan, versi, dan tindakan pencegahan.

Pencegahan

Pencegahan dilakukan dengan memusatkan parameter, menggunakan validasi eksplisit, mencatat versi, menyimpan checkpoint, dan menyediakan pesan kesalahan yang spesifik. Notebook yang baik tidak hanya bekerja pada kondisi ideal, tetapi juga gagal secara aman ketika input atau lingkungan tidak memenuhi syarat. Tambahkan satu sel diagnostik pada awal notebook dan satu sel manifest pada akhir proyek.

G.05 PermissionError pada Drive

Gejala dan Bukti Awal

Gejala yang terlihat adalah permissionerror pada drive. Jangan langsung mengubah banyak sel. Salin pesan kesalahan lengkap, catat nomor sel dan waktu kejadian, serta periksa apakah masalah dapat diulang setelah Restart session. Bukti awal yang berguna

meliputi versi Python dan paket, direktori kerja, daftar berkas, shape dan dtype data, parameter kimia, status CPU/GPU, serta bagian traceback yang menunjuk baris pertama dari kode pengguna.

Penyebab yang Paling Mungkin

Penyebab utama yang perlu diuji adalah: Mount belum aktif atau jalur tidak dapat ditulis. Penyebab tersebut harus diperlakukan sebagai hipotesis. Periksa prasyarat paling sederhana terlebih dahulu, lalu bergerak ke dependensi dan algoritma. Apabila beberapa penyebab mungkin, buat uji minimal yang mengisolasi satu komponen, misalnya hanya mengimpor modul, hanya membaca satu berkas, atau menjalankan molekul kecil.

Langkah Pemulihan

1281. Simpan notebook dan artefak yang masih tersedia sebelum melakukan restart atau instalasi ulang.
1282. Buat uji minimal untuk memastikan gejala dapat direproduksi tanpa sel yang tidak berkaitan.
1283. Mount ulang, periksa jalur MyDrive, dan uji menulis berkas kecil.
1284. Jalankan kembali seluruh notebook dari sesi bersih dan bandingkan status serta keluaran dengan catatan sebelum perbaikan.
1285. Dokumentasikan akar masalah, perubahan yang dilakukan, versi, dan tindakan pencegahan.

Pencegahan

Pencegahan dilakukan dengan memusatkan parameter, menggunakan validasi eksplisit, mencatat versi, menyimpan checkpoint, dan menyediakan pesan kesalahan yang spesifik. Notebook yang baik tidak hanya bekerja pada kondisi ideal, tetapi juga gagal secara aman ketika input atau lingkungan tidak memenuhi syarat. Tambahkan satu sel diagnostik pada awal notebook dan satu sel manifest pada akhir proyek.

G.06 FileNotFoundError

Gejala dan Bukti Awal

Gejala yang terlihat adalah `FileNotFoundError`. Jangan langsung mengubah banyak sel. Salin pesan kesalahan lengkap, catat nomor sel dan waktu kejadian, serta periksa apakah masalah dapat diulang setelah Restart session. Bukti awal yang berguna meliputi versi Python dan paket, direktori kerja, daftar berkas, shape dan dtype data, parameter kimia, status CPU/GPU, serta bagian traceback yang menunjuk baris pertama dari kode pengguna.

Penyebab yang Paling Mungkin

Penyebab utama yang perlu diuji adalah: Jalur relatif salah atau berkas belum diunggah. Penyebab tersebut harus diperlakukan sebagai hipotesis. Periksa prasyarat paling sederhana terlebih dahulu, lalu bergerak ke dependensi dan algoritma. Apabila beberapa penyebab mungkin, buat uji minimal yang mengisolasi satu komponen, misalnya hanya mengimpor modul, hanya membaca satu berkas, atau menjalankan molekul kecil.

Langkah Pemulihan

1286. Simpan notebook dan artefak yang masih tersedia sebelum melakukan restart atau instalasi ulang.
1287. Buat uji minimal untuk memastikan gejala dapat direproduksi tanpa sel yang tidak berkaitan.
1288. Cetak `Path.cwd()`, daftar isi folder, dan gunakan `pathlib` untuk membangun jalur.
1289. Jalankan kembali seluruh notebook dari sesi bersih dan bandingkan status serta keluaran dengan catatan sebelum perbaikan.
1290. Dokumentasikan akar masalah, perubahan yang dilakukan, versi, dan tindakan pencegahan.

Pencegahan

Pencegahan dilakukan dengan memusatkan parameter, menggunakan validasi eksplisit, mencatat versi, menyimpan checkpoint, dan menyediakan pesan kesalahan yang spesifik. Notebook yang baik tidak hanya bekerja pada kondisi ideal, tetapi juga gagal secara aman ketika input atau lingkungan tidak memenuhi syarat. Tambahkan satu sel diagnostik pada awal notebook dan satu sel manifest pada akhir proyek.

G.07 UnicodeDecodeError

Gejala dan Bukti Awal

Gejala yang terlihat adalah `unicodedecodeerror`. Jangan langsung mengubah banyak sel. Salin pesan kesalahan lengkap, catat nomor sel dan waktu kejadian, serta periksa apakah masalah dapat diulang setelah Restart session. Bukti awal yang berguna meliputi versi Python dan paket, direktori kerja, daftar berkas, shape dan dtype data, parameter kimia, status CPU/GPU, serta bagian traceback yang menunjuk baris pertama dari kode pengguna.

Penyebab yang Paling Mungkin

Penyebab utama yang perlu diuji adalah: Encoding berkas tidak sesuai. Penyebab tersebut harus diperlakukan sebagai hipotesis. Periksa prasyarat paling sederhana terlebih dahulu, lalu bergerak ke dependensi dan algoritma. Apabila beberapa penyebab mungkin, buat uji minimal yang mengisolasi satu komponen, misalnya hanya mengimpor modul, hanya membaca satu berkas, atau menjalankan molekul kecil.

Langkah Pemulihan

1291. Simpan notebook dan artefak yang masih tersedia sebelum melakukan restart atau instalasi ulang.
1292. Buat uji minimal untuk memastikan gejala dapat direproduksi tanpa sel yang tidak berkaitan.
1293. Deteksi atau tentukan encoding yang benar; hindari mengabaikan karakter tanpa catatan.
1294. Jalankan kembali seluruh notebook dari sesi bersih dan bandingkan status serta keluaran dengan catatan sebelum perbaikan.
1295. Dokumentasikan akar masalah, perubahan yang dilakukan, versi, dan tindakan pencegahan.

Pencegahan

Pencegahan dilakukan dengan memusatkan parameter, menggunakan validasi eksplisit, mencatat versi, menyimpan checkpoint, dan menyediakan pesan kesalahan yang spesifik. Notebook yang baik tidak hanya bekerja pada kondisi ideal, tetapi juga gagal secara aman ketika input atau lingkungan tidak memenuhi syarat. Tambahkan satu sel diagnostik pada awal notebook dan satu sel manifest pada akhir proyek.

G.08 CSV terbaca satu kolom

Gejala dan Bukti Awal

Gejala yang terlihat adalah csv terbaca satu kolom. Jangan langsung mengubah banyak sel. Salin pesan kesalahan lengkap, catat nomor sel dan waktu kejadian, serta periksa apakah masalah dapat diulang setelah Restart session. Bukti awal yang berguna meliputi versi Python dan paket, direktori kerja, daftar berkas, shape dan dtype data, parameter kimia, status CPU/GPU, serta bagian traceback yang menunjuk baris pertama dari kode pengguna.

Penyebab yang Paling Mungkin

Penyebab utama yang perlu diuji adalah: Delimiter atau locale tidak sesuai. Penyebab tersebut harus diperlakukan sebagai hipotesis. Periksa prasyarat paling sederhana terlebih dahulu, lalu bergerak ke dependensi dan algoritma. Apabila beberapa penyebab mungkin, buat uji minimal yang mengisolasi satu komponen, misalnya hanya mengimpor modul, hanya membaca satu berkas, atau menjalankan molekul kecil.

Langkah Pemulihan

1296. Simpan notebook dan artefak yang masih tersedia sebelum melakukan restart atau instalasi ulang.
1297. Buat uji minimal untuk memastikan gejala dapat direproduksi tanpa sel yang tidak berkaitan.

1298. Periksa beberapa baris mentah dan tentukan sep, decimal, serta encoding.
1299. Jalankan kembali seluruh notebook dari sesi bersih dan bandingkan status serta keluaran dengan catatan sebelum perbaikan.
1300. Dokumentasikan akar masalah, perubahan yang dilakukan, versi, dan tindakan pencegahan.

Pencegahan

Pencegahan dilakukan dengan memusatkan parameter, menggunakan validasi eksplisit, mencatat versi, menyimpan checkpoint, dan menyediakan pesan kesalahan yang spesifik. Notebook yang baik tidak hanya bekerja pada kondisi ideal, tetapi juga gagal secara aman ketika input atau lingkungan tidak memenuhi syarat. Tambahkan satu sel diagnostik pada awal notebook dan satu sel manifest pada akhir proyek.

G.09 Kolom numerik menjadi object

Gejala dan Bukti Awal

Gejala yang terlihat adalah kolom numerik menjadi object. Jangan langsung mengubah banyak sel. Salin pesan kesalahan lengkap, catat nomor sel dan waktu kejadian, serta periksa apakah masalah dapat diulang setelah Restart session. Bukti awal yang berguna meliputi versi Python dan paket, direktori kerja, daftar berkas, shape dan dtype data, parameter kimia, status CPU/GPU, serta bagian traceback yang menunjuk baris pertama dari kode pengguna.

Penyebab yang Paling Mungkin

Penyebab utama yang perlu diuji adalah: Ada simbol, spasi, atau pemisah desimal campuran. Penyebab tersebut harus diperlakukan sebagai hipotesis. Periksa prasyarat paling sederhana terlebih dahulu, lalu bergerak ke dependensi dan algoritma. Apabila beberapa penyebab mungkin, buat uji minimal yang mengisolasi satu komponen, misalnya hanya mengimpor modul, hanya membaca satu berkas, atau menjalankan molekul kecil.

Langkah Pemulihan

1301. Simpan notebook dan artefak yang masih tersedia sebelum melakukan restart atau instalasi ulang.
1302. Buat uji minimal untuk memastikan gejala dapat direproduksi tanpa sel yang tidak berkaitan.
1303. Gunakan `to_numeric` dengan audit nilai yang gagal dikonversi.
1304. Jalankan kembali seluruh notebook dari sesi bersih dan bandingkan status serta keluaran dengan catatan sebelum perbaikan.
1305. Dokumentasikan akar masalah, perubahan yang dilakukan, versi, dan tindakan pencegahan.

Pencegahan

Pencegahan dilakukan dengan memusatkan parameter, menggunakan validasi eksplisit, mencatat versi, menyimpan checkpoint, dan menyediakan pesan kesalahan yang spesifik. Notebook yang baik tidak hanya bekerja pada kondisi ideal, tetapi juga gagal secara aman ketika input atau lingkungan tidak memenuhi syarat. Tambahkan satu sel diagnostik pada awal notebook dan satu sel manifest pada akhir proyek.

G.10 DataFrame kosong setelah filter

Gejala dan Bukti Awal

Gejala yang terlihat adalah dataframe kosong setelah filter. Jangan langsung mengubah banyak sel. Salin pesan kesalahan lengkap, catat nomor sel dan waktu kejadian, serta periksa apakah masalah dapat diulang setelah Restart session. Bukti awal yang berguna meliputi versi Python dan paket, direktori kerja, daftar berkas, shape dan dtype data, parameter kimia, status CPU/GPU, serta bagian traceback yang menunjuk baris pertama dari kode pengguna.

Penyebab yang Paling Mungkin

Penyebab utama yang perlu diuji adalah: Kondisi terlalu ketat atau tipe kolom salah. Penyebab tersebut harus diperlakukan sebagai hipotesis. Periksa prasyarat paling sederhana terlebih dahulu, lalu bergerak ke dependensi dan algoritma. Apabila beberapa penyebab mungkin, buat uji minimal yang mengisolasi satu komponen, misalnya hanya mengimpor modul, hanya membaca satu berkas, atau menjalankan molekul kecil.

Langkah Pemulihan

1306. Simpan notebook dan artefak yang masih tersedia sebelum melakukan restart atau instalasi ulang.
1307. Buat uji minimal untuk memastikan gejala dapat direproduksi tanpa sel yang tidak berkaitan.
1308. Cetak jumlah baris tiap tahap dan uji kondisi satu per satu.
1309. Jalankan kembali seluruh notebook dari sesi bersih dan bandingkan status serta keluaran dengan catatan sebelum perbaikan.
1310. Dokumentasikan akar masalah, perubahan yang dilakukan, versi, dan tindakan pencegahan.

Pencegahan

Pencegahan dilakukan dengan memusatkan parameter, menggunakan validasi eksplisit, mencatat versi, menyimpan checkpoint, dan menyediakan pesan kesalahan yang spesifik. Notebook yang baik tidak hanya bekerja pada kondisi ideal, tetapi juga gagal secara aman ketika input atau lingkungan tidak memenuhi syarat. Tambahkan satu sel diagnostik pada awal notebook dan satu sel manifest pada akhir proyek.

G.11 SettingWithCopyWarning

Gejala dan Bukti Awal

Gejala yang terlihat adalah `settingwithcopywarning`. Jangan langsung mengubah banyak sel. Salin pesan kesalahan lengkap, catat nomor sel dan waktu kejadian, serta periksa apakah masalah dapat diulang setelah Restart session. Bukti awal yang berguna meliputi versi Python dan paket, direktori kerja, daftar berkas, shape dan dtype data, parameter kimia, status CPU/GPU, serta bagian traceback yang menunjuk baris pertama dari kode pengguna.

Penyebab yang Paling Mungkin

Penyebab utama yang perlu diuji adalah: Modifikasi dilakukan pada view. Penyebab tersebut harus diperlakukan sebagai hipotesis. Periksa prasyarat paling sederhana terlebih dahulu, lalu bergerak ke dependensi dan algoritma. Apabila beberapa penyebab mungkin, buat uji minimal yang mengisolasi satu komponen, misalnya hanya mengimpor modul, hanya membaca satu berkas, atau menjalankan molekul kecil.

Langkah Pemulihan

1311. Simpan notebook dan artefak yang masih tersedia sebelum melakukan restart atau instalasi ulang.
1312. Buat uji minimal untuk memastikan gejala dapat direproduksi tanpa sel yang tidak berkaitan.
1313. Gunakan `.loc` dan `.copy()` sebelum mengubah subset.
1314. Jalankan kembali seluruh notebook dari sesi bersih dan bandingkan status serta keluaran dengan catatan sebelum perbaikan.
1315. Dokumentasikan akar masalah, perubahan yang dilakukan, versi, dan tindakan pencegahan.

Pencegahan

Pencegahan dilakukan dengan memusatkan parameter, menggunakan validasi eksplisit, mencatat versi, menyimpan checkpoint, dan menyediakan pesan kesalahan yang spesifik. Notebook yang baik tidak hanya bekerja pada kondisi ideal, tetapi juga gagal secara aman ketika input atau lingkungan tidak memenuhi syarat. Tambahkan satu sel diagnostik pada awal notebook dan satu sel manifest pada akhir proyek.

G.12 ValueError shape mismatch

Gejala dan Bukti Awal

Gejala yang terlihat adalah `valueerror shape mismatch`. Jangan langsung mengubah banyak sel. Salin pesan kesalahan lengkap, catat nomor sel dan waktu kejadian, serta periksa apakah masalah dapat diulang setelah Restart session. Bukti awal yang berguna

meliputi versi Python dan paket, direktori kerja, daftar berkas, shape dan dtype data, parameter kimia, status CPU/GPU, serta bagian traceback yang menunjuk baris pertama dari kode pengguna.

Penyebab yang Paling Mungkin

Penyebab utama yang perlu diuji adalah: Ukuran array tidak konsisten. Penyebab tersebut harus diperlakukan sebagai hipotesis. Periksa prasyarat paling sederhana terlebih dahulu, lalu bergerak ke dependensi dan algoritma. Apabila beberapa penyebab mungkin, buat uji minimal yang mengisolasi satu komponen, misalnya hanya mengimpor modul, hanya membaca satu berkas, atau menjalankan molekul kecil.

Langkah Pemulihan

1316. Simpan notebook dan artefak yang masih tersedia sebelum melakukan restart atau instalasi ulang.
1317. Buat uji minimal untuk memastikan gejala dapat direproduksi tanpa sel yang tidak berkaitan.
1318. Cetak shape setiap input dan tambahkan assert sebelum operasi.
1319. Jalankan kembali seluruh notebook dari sesi bersih dan bandingkan status serta keluaran dengan catatan sebelum perbaikan.
1320. Dokumentasikan akar masalah, perubahan yang dilakukan, versi, dan tindakan pencegahan.

Pencegahan

Pencegahan dilakukan dengan memusatkan parameter, menggunakan validasi eksplisit, mencatat versi, menyimpan checkpoint, dan menyediakan pesan kesalahan yang spesifik. Notebook yang baik tidak hanya bekerja pada kondisi ideal, tetapi juga gagal secara aman ketika input atau lingkungan tidak memenuhi syarat. Tambahkan satu sel diagnostik pada awal notebook dan satu sel manifest pada akhir proyek.

G.13 NaN atau inf muncul

Gejala dan Bukti Awal

Gejala yang terlihat adalah nan atau inf muncul. Jangan langsung mengubah banyak sel. Salin pesan kesalahan lengkap, catat nomor sel dan waktu kejadian, serta periksa apakah masalah dapat diulang setelah Restart session. Bukti awal yang berguna meliputi versi Python dan paket, direktori kerja, daftar berkas, shape dan dtype data, parameter kimia, status CPU/GPU, serta bagian traceback yang menunjuk baris pertama dari kode pengguna.

Penyebab yang Paling Mungkin

Penyebab utama yang perlu diuji adalah: Pembagian nol, log domain salah, atau perhitungan gagal. Penyebab tersebut harus diperlakukan sebagai hipotesis. Periksa prasyarat paling sederhana terlebih dahulu, lalu bergerak ke dependensi dan algoritma. Apabila beberapa penyebab mungkin, buat uji minimal yang mengisolasi satu komponen, misalnya hanya mengimpor modul, hanya membaca satu berkas, atau menjalankan molekul kecil.

Langkah Pemulihan

1321. Simpan notebook dan artefak yang masih tersedia sebelum melakukan restart atau instalasi ulang.
1322. Buat uji minimal untuk memastikan gejala dapat direproduksi tanpa sel yang tidak berkaitan.
1323. Gunakan isfinite, telusuri titik pertama, dan hentikan sebelum ekspor.
1324. Jalankan kembali seluruh notebook dari sesi bersih dan bandingkan status serta keluaran dengan catatan sebelum perbaikan.
1325. Dokumentasikan akar masalah, perubahan yang dilakukan, versi, dan tindakan pencegahan.

Pencegahan

Pencegahan dilakukan dengan memusatkan parameter, menggunakan validasi eksplisit, mencatat versi, menyimpan checkpoint, dan menyediakan pesan kesalahan yang spesifik. Notebook yang baik tidak hanya bekerja pada kondisi ideal, tetapi juga gagal secara aman ketika input atau lingkungan tidak memenuhi syarat. Tambahkan satu sel diagnostik pada awal notebook dan satu sel manifest pada akhir proyek.

G.14 Grafik kosong

Gejala dan Bukti Awal

Gejala yang terlihat adalah grafik kosong. Jangan langsung mengubah banyak sel. Salin pesan kesalahan lengkap, catat nomor sel dan waktu kejadian, serta periksa apakah masalah dapat diulang setelah Restart session. Bukti awal yang berguna meliputi versi Python dan paket, direktori kerja, daftar berkas, shape dan dtype data, parameter kimia, status CPU/GPU, serta bagian traceback yang menunjuk baris pertama dari kode pengguna.

Penyebab yang Paling Mungkin

Penyebab utama yang perlu diuji adalah: Data kosong, semua NaN, atau rentang sumbu keliru. Penyebab tersebut harus diperlakukan sebagai hipotesis. Periksa prasyarat paling sederhana terlebih dahulu, lalu bergerak ke dependensi dan algoritma. Apabila beberapa

penyebab mungkin, buat uji minimal yang mengisolasi satu komponen, misalnya hanya mengimpor modul, hanya membaca satu berkas, atau menjalankan molekul kecil.

Langkah Pemulihan

1326. Simpan notebook dan artefak yang masih tersedia sebelum melakukan restart atau instalasi ulang.
1327. Buat uji minimal untuk memastikan gejala dapat direproduksi tanpa sel yang tidak berkaitan.
1328. Cetak statistik data dan biarkan Matplotlib menentukan batas awal.
1329. Jalankan kembali seluruh notebook dari sesi bersih dan bandingkan status serta keluaran dengan catatan sebelum perbaikan.
1330. Dokumentasikan akar masalah, perubahan yang dilakukan, versi, dan tindakan pencegahan.

Pencegahan

Pencegahan dilakukan dengan memusatkan parameter, menggunakan validasi eksplisit, mencatat versi, menyimpan checkpoint, dan menyediakan pesan kesalahan yang spesifik. Notebook yang baik tidak hanya bekerja pada kondisi ideal, tetapi juga gagal secara aman ketika input atau lingkungan tidak memenuhi syarat. Tambahkan satu sel diagnostik pada awal notebook dan satu sel manifest pada akhir proyek.

G.15 Label grafik terpotong

Gejala dan Bukti Awal

Gejala yang terlihat adalah label grafik terpotong. Jangan langsung mengubah banyak sel. Salin pesan kesalahan lengkap, catat nomor sel dan waktu kejadian, serta periksa apakah masalah dapat diulang setelah Restart session. Bukti awal yang berguna meliputi versi Python dan paket, direktori kerja, daftar berkas, shape dan dtype data, parameter kimia, status CPU/GPU, serta bagian traceback yang menunjuk baris pertama dari kode pengguna.

Penyebab yang Paling Mungkin

Penyebab utama yang perlu diuji adalah: Layout dan ukuran gambar tidak sesuai. Penyebab tersebut harus diperlakukan sebagai hipotesis. Periksa prasyarat paling sederhana terlebih dahulu, lalu bergerak ke dependensi dan algoritma. Apabila beberapa penyebab mungkin, buat uji minimal yang mengisolasi satu komponen, misalnya hanya mengimpor modul, hanya membaca satu berkas, atau menjalankan molekul kecil.

Langkah Pemulihan

1331. Simpan notebook dan artefak yang masih tersedia sebelum melakukan restart atau instalasi ulang.

1332. Buat uji minimal untuk memastikan gejala dapat direproduksi tanpa sel yang tidak berkaitan.
1333. Gunakan `tight_layout` atau `constrained_layout` dan periksa file ekspor.
1334. Jalankan kembali seluruh notebook dari sesi bersih dan bandingkan status serta keluaran dengan catatan sebelum perbaikan.
1335. Dokumentasikan akar masalah, perubahan yang dilakukan, versi, dan tindakan pencegahan.

Pencegahan

Pencegahan dilakukan dengan memusatkan parameter, menggunakan validasi eksplisit, mencatat versi, menyimpan checkpoint, dan menyediakan pesan kesalahan yang spesifik. Notebook yang baik tidak hanya bekerja pada kondisi ideal, tetapi juga gagal secara aman ketika input atau lingkungan tidak memenuhi syarat. Tambahkan satu sel diagnostik pada awal notebook dan satu sel manifest pada akhir proyek.

G.16 Unggah berkas tidak muncul

Gejala dan Bukti Awal

Gejala yang terlihat adalah unggah berkas tidak muncul. Jangan langsung mengubah banyak sel. Salin pesan kesalahan lengkap, catat nomor sel dan waktu kejadian, serta periksa apakah masalah dapat diulang setelah Restart session. Bukti awal yang berguna meliputi versi Python dan paket, direktori kerja, daftar berkas, shape dan dtype data, parameter kimia, status CPU/GPU, serta bagian traceback yang menunjuk baris pertama dari kode pengguna.

Penyebab yang Paling Mungkin

Penyebab utama yang perlu diuji adalah: Dialog dibatalkan atau kode tidak berjalan di Colab. Penyebab tersebut harus diperlakukan sebagai hipotesis. Periksa prasyarat paling sederhana terlebih dahulu, lalu bergerak ke dependensi dan algoritma. Apabila beberapa penyebab mungkin, buat uji minimal yang mengisolasi satu komponen, misalnya hanya mengimpor modul, hanya membaca satu berkas, atau menjalankan molekul kecil.

Langkah Pemulihan

1336. Simpan notebook dan artefak yang masih tersedia sebelum melakukan restart atau instalasi ulang.
1337. Buat uji minimal untuk memastikan gejala dapat direproduksi tanpa sel yang tidak berkaitan.
1338. Periksa uploaded keys dan sediakan jalur alternatif.
1339. Jalankan kembali seluruh notebook dari sesi bersih dan bandingkan status serta keluaran dengan catatan sebelum perbaikan.
1340. Dokumentasikan akar masalah, perubahan yang dilakukan, versi, dan tindakan pencegahan.

Pencegahan

Pencegahan dilakukan dengan memusatkan parameter, menggunakan validasi eksplisit, mencatat versi, menyimpan checkpoint, dan menyediakan pesan kesalahan yang spesifik. Notebook yang baik tidak hanya bekerja pada kondisi ideal, tetapi juga gagal secara aman ketika input atau lingkungan tidak memenuhi syarat. Tambahkan satu sel diagnostik pada awal notebook dan satu sel manifest pada akhir proyek.

G.17 Unduhan tidak dimulai

Gejala dan Bukti Awal

Gejala yang terlihat adalah unduhan tidak dimulai. Jangan langsung mengubah banyak sel. Salin pesan kesalahan lengkap, catat nomor sel dan waktu kejadian, serta periksa apakah masalah dapat diulang setelah Restart session. Bukti awal yang berguna meliputi versi Python dan paket, direktori kerja, daftar berkas, shape dan dtype data, parameter kimia, status CPU/GPU, serta bagian traceback yang menunjuk baris pertama dari kode pengguna.

Penyebab yang Paling Mungkin

Penyebab utama yang perlu diuji adalah: Berkas belum dibuat atau terlalu besar. Penyebab tersebut harus diperlakukan sebagai hipotesis. Periksa prasyarat paling sederhana terlebih dahulu, lalu bergerak ke dependensi dan algoritma. Apabila beberapa penyebab mungkin, buat uji minimal yang mengisolasi satu komponen, misalnya hanya mengimpor modul, hanya membaca satu berkas, atau menjalankan molekul kecil.

Langkah Pemulihan

1341. Simpan notebook dan artefak yang masih tersedia sebelum melakukan restart atau instalasi ulang.
1342. Buat uji minimal untuk memastikan gejala dapat direproduksi tanpa sel yang tidak berkaitan.
1343. Periksa keberadaan dan ukuran, lalu kompres atau pindahkan ke Drive.
1344. Jalankan kembali seluruh notebook dari sesi bersih dan bandingkan status serta keluaran dengan catatan sebelum perbaikan.
1345. Dokumentasikan akar masalah, perubahan yang dilakukan, versi, dan tindakan pencegahan.

Pencegahan

Pencegahan dilakukan dengan memusatkan parameter, menggunakan validasi eksplisit, mencatat versi, menyimpan checkpoint, dan menyediakan pesan kesalahan yang spesifik. Notebook yang baik tidak hanya bekerja pada kondisi ideal, tetapi juga gagal secara aman ketika input atau lingkungan tidak memenuhi syarat. Tambahkan satu sel diagnostik pada awal notebook dan satu sel manifest pada akhir proyek.

G.18 nvidia-smi tidak ditemukan

Gejala dan Bukti Awal

Gejala yang terlihat adalah nvidia-smi tidak ditemukan. Jangan langsung mengubah banyak sel. Salin pesan kesalahan lengkap, catat nomor sel dan waktu kejadian, serta periksa apakah masalah dapat diulang setelah Restart session. Bukti awal yang berguna meliputi versi Python dan paket, direktori kerja, daftar berkas, shape dan dtype data, parameter kimia, status CPU/GPU, serta bagian traceback yang menunjuk baris pertama dari kode pengguna.

Penyebab yang Paling Mungkin

Penyebab utama yang perlu diuji adalah: Runtime bukan GPU. Penyebab tersebut harus diperlakukan sebagai hipotesis. Periksa prasyarat paling sederhana terlebih dahulu, lalu bergerak ke dependensi dan algoritma. Apabila beberapa penyebab mungkin, buat uji minimal yang mengisolasi satu komponen, misalnya hanya mengimpor modul, hanya membaca satu berkas, atau menjalankan molekul kecil.

Langkah Pemulihan

1346. Simpan notebook dan artefak yang masih tersedia sebelum melakukan restart atau instalasi ulang.
1347. Buat uji minimal untuk memastikan gejala dapat direproduksi tanpa sel yang tidak berkaitan.
1348. Pilih akselerator GPU atau gunakan fallback CPU.
1349. Jalankan kembali seluruh notebook dari sesi bersih dan bandingkan status serta keluaran dengan catatan sebelum perbaikan.
1350. Dokumentasikan akar masalah, perubahan yang dilakukan, versi, dan tindakan pencegahan.

Pencegahan

Pencegahan dilakukan dengan memusatkan parameter, menggunakan validasi eksplisit, mencatat versi, menyimpan checkpoint, dan menyediakan pesan kesalahan yang spesifik. Notebook yang baik tidak hanya bekerja pada kondisi ideal, tetapi juga gagal secara aman ketika input atau lingkungan tidak memenuhi syarat. Tambahkan satu sel diagnostik pada awal notebook dan satu sel manifest pada akhir proyek.

G.19 GPU terdeteksi tetapi kode tetap CPU

Gejala dan Bukti Awal

Gejala yang terlihat adalah gpu terdeteksi tetapi kode tetap cpu. Jangan langsung mengubah banyak sel. Salin pesan kesalahan lengkap, catat nomor sel dan waktu kejadian, serta periksa apakah masalah dapat diulang setelah Restart session. Bukti awal

yang berguna meliputi versi Python dan paket, direktori kerja, daftar berkas, shape dan dtype data, parameter kimia, status CPU/GPU, serta bagian traceback yang menunjuk baris pertama dari kode pengguna.

Penyebab yang Paling Mungkin

Penyebab utama yang perlu diuji adalah: Pustaka tidak memakai backend GPU. Penyebab tersebut harus diperlakukan sebagai hipotesis. Periksa prasyarat paling sederhana terlebih dahulu, lalu bergerak ke dependensi dan algoritma. Apabila beberapa penyebab mungkin, buat uji minimal yang mengisolasi satu komponen, misalnya hanya mengimpor modul, hanya membaca satu berkas, atau menjalankan molekul kecil.

Langkah Pemulihan

1351. Simpan notebook dan artefak yang masih tersedia sebelum melakukan restart atau instalasi ulang.
1352. Buat uji minimal untuk memastikan gejala dapat direproduksi tanpa sel yang tidak berkaitan.
1353. Gunakan implementasi GPU yang kompatibel dan ukur backend secara eksplisit.
1354. Jalankan kembali seluruh notebook dari sesi bersih dan bandingkan status serta keluaran dengan catatan sebelum perbaikan.
1355. Dokumentasikan akar masalah, perubahan yang dilakukan, versi, dan tindakan pencegahan.

Pencegahan

Pencegahan dilakukan dengan memusatkan parameter, menggunakan validasi eksplisit, mencatat versi, menyimpan checkpoint, dan menyediakan pesan kesalahan yang spesifik. Notebook yang baik tidak hanya bekerja pada kondisi ideal, tetapi juga gagal secara aman ketika input atau lingkungan tidak memenuhi syarat. Tambahkan satu sel diagnostik pada awal notebook dan satu sel manifest pada akhir proyek.

G.20 CUDA mismatch

Gejala dan Bukti Awal

Gejala yang terlihat adalah cuda mismatch. Jangan langsung mengubah banyak sel. Salin pesan kesalahan lengkap, catat nomor sel dan waktu kejadian, serta periksa apakah masalah dapat diulang setelah Restart session. Bukti awal yang berguna meliputi versi Python dan paket, direktori kerja, daftar berkas, shape dan dtype data, parameter kimia, status CPU/GPU, serta bagian traceback yang menunjuk baris pertama dari kode pengguna.

Penyebab yang Paling Mungkin

Penyebab utama yang perlu diuji adalah: Versi driver, toolkit, dan wheel tidak cocok. Penyebab tersebut harus diperlakukan sebagai hipotesis. Periksa prasyarat paling sederhana terlebih dahulu, lalu bergerak ke dependensi dan algoritma. Apabila beberapa penyebab mungkin, buat uji minimal yang mengisolasi satu komponen, misalnya hanya mengimpor modul, hanya membaca satu berkas, atau menjalankan molekul kecil.

Langkah Pemulihan

1356. Simpan notebook dan artefak yang masih tersedia sebelum melakukan restart atau instalasi ulang.
1357. Buat uji minimal untuk memastikan gejala dapat direproduksi tanpa sel yang tidak berkaitan.
1358. Ikuti matriks kompatibilitas paket, restart, dan jangan mencampur wheel acak.
1359. Jalankan kembali seluruh notebook dari sesi bersih dan bandingkan status serta keluaran dengan catatan sebelum perbaikan.
1360. Dokumentasikan akar masalah, perubahan yang dilakukan, versi, dan tindakan pencegahan.

Pencegahan

Pencegahan dilakukan dengan memusatkan parameter, menggunakan validasi eksplisit, mencatat versi, menyimpan checkpoint, dan menyediakan pesan kesalahan yang spesifik. Notebook yang baik tidak hanya bekerja pada kondisi ideal, tetapi juga gagal secara aman ketika input atau lingkungan tidak memenuhi syarat. Tambahkan satu sel diagnostik pada awal notebook dan satu sel manifest pada akhir proyek.

G.21 RDKit MolFromSmiles menghasilkan None

Gejala dan Bukti Awal

Gejala yang terlihat adalah rdkit molfromsmiles menghasilkan none. Jangan langsung mengubah banyak sel. Salin pesan kesalahan lengkap, catat nomor sel dan waktu kejadian, serta periksa apakah masalah dapat diulang setelah Restart session. Bukti awal yang berguna meliputi versi Python dan paket, direktori kerja, daftar berkas, shape dan dtype data, parameter kimia, status CPU/GPU, serta bagian traceback yang menunjuk baris pertama dari kode pengguna.

Penyebab yang Paling Mungkin

Penyebab utama yang perlu diuji adalah: SMILES tidak valid. Penyebab tersebut harus diperlakukan sebagai hipotesis. Periksa prasyarat paling sederhana terlebih dahulu, lalu bergerak ke dependensi dan algoritma. Apabila beberapa penyebab mungkin, buat uji minimal yang mengisolasi satu komponen, misalnya hanya mengimpor modul, hanya membaca satu berkas, atau menjalankan molekul kecil.

Langkah Pemulihan

1361. Simpan notebook dan artefak yang masih tersedia sebelum melakukan restart atau instalasi ulang.
1362. Buat uji minimal untuk memastikan gejala dapat direproduksi tanpa sel yang tidak berkaitan.
1363. Periksa sintaks, muatan, stereokimia, dan sanitasi.
1364. Jalankan kembali seluruh notebook dari sesi bersih dan bandingkan status serta keluaran dengan catatan sebelum perbaikan.
1365. Dokumentasikan akar masalah, perubahan yang dilakukan, versi, dan tindakan pencegahan.

Pencegahan

Pencegahan dilakukan dengan memusatkan parameter, menggunakan validasi eksplisit, mencatat versi, menyimpan checkpoint, dan menyediakan pesan kesalahan yang spesifik. Notebook yang baik tidak hanya bekerja pada kondisi ideal, tetapi juga gagal secara aman ketika input atau lingkungan tidak memenuhi syarat. Tambahkan satu sel diagnostik pada awal notebook dan satu sel manifest pada akhir proyek.

G.22 EmbedMolecule gagal

Gejala dan Bukti Awal

Gejala yang terlihat adalah embedmolecule gagal. Jangan langsung mengubah banyak sel. Salin pesan kesalahan lengkap, catat nomor sel dan waktu kejadian, serta periksa apakah masalah dapat diulang setelah Restart session. Bukti awal yang berguna meliputi versi Python dan paket, direktori kerja, daftar berkas, shape dan dtype data, parameter kimia, status CPU/GPU, serta bagian traceback yang menunjuk baris pertama dari kode pengguna.

Penyebab yang Paling Mungkin

Penyebab utama yang perlu diuji adalah: Struktur sulit, parameter embedding, atau seed. Penyebab tersebut harus diperlakukan sebagai hipotesis. Periksa prasyarat paling sederhana terlebih dahulu, lalu bergerak ke dependensi dan algoritma. Apabila beberapa penyebab mungkin, buat uji minimal yang mengisolasi satu komponen, misalnya hanya mengimpor modul, hanya membaca satu berkas, atau menjalankan molekul kecil.

Langkah Pemulihan

1366. Simpan notebook dan artefak yang masih tersedia sebelum melakukan restart atau instalasi ulang.
1367. Buat uji minimal untuk memastikan gejala dapat direproduksi tanpa sel yang tidak berkaitan.
1368. Gunakan ETKDGV3, tambah percobaan, dan periksa struktur input.

1369. Jalankan kembali seluruh notebook dari sesi bersih dan bandingkan status serta keluaran dengan catatan sebelum perbaikan.
1370. Dokumentasikan akar masalah, perubahan yang dilakukan, versi, dan tindakan pencegahan.

Pencegahan

Pencegahan dilakukan dengan memusatkan parameter, menggunakan validasi eksplisit, mencatat versi, menyimpan checkpoint, dan menyediakan pesan kesalahan yang spesifik. Notebook yang baik tidak hanya bekerja pada kondisi ideal, tetapi juga gagal secara aman ketika input atau lingkungan tidak memenuhi syarat. Tambahkan satu sel diagnostik pada awal notebook dan satu sel manifest pada akhir proyek.

G.23 MMFF tidak tersedia

Gejala dan Bukti Awal

Gejala yang terlihat adalah mmff tidak tersedia. Jangan langsung mengubah banyak sel. Salin pesan kesalahan lengkap, catat nomor sel dan waktu kejadian, serta periksa apakah masalah dapat diulang setelah Restart session. Bukti awal yang berguna meliputi versi Python dan paket, direktori kerja, daftar berkas, shape dan dtype data, parameter kimia, status CPU/GPU, serta bagian traceback yang menunjuk baris pertama dari kode pengguna.

Penyebab yang Paling Mungkin

Penyebab utama yang perlu diuji adalah: Parameter atom tidak tercakup. Penyebab tersebut harus diperlakukan sebagai hipotesis. Periksa prasyarat paling sederhana terlebih dahulu, lalu bergerak ke dependensi dan algoritma. Apabila beberapa penyebab mungkin, buat uji minimal yang mengisolasi satu komponen, misalnya hanya mengimpor modul, hanya membaca satu berkas, atau menjalankan molekul kecil.

Langkah Pemulihan

1371. Simpan notebook dan artefak yang masih tersedia sebelum melakukan restart atau instalasi ulang.
1372. Buat uji minimal untuk memastikan gejala dapat direproduksi tanpa sel yang tidak berkaitan.
1373. Gunakan UFF sebagai fallback dan catat perubahan metode.
1374. Jalankan kembali seluruh notebook dari sesi bersih dan bandingkan status serta keluaran dengan catatan sebelum perbaikan.
1375. Dokumentasikan akar masalah, perubahan yang dilakukan, versi, dan tindakan pencegahan.

Pencegahan

Pencegahan dilakukan dengan memusatkan parameter, menggunakan validasi eksplisit, mencatat versi, menyimpan checkpoint, dan menyediakan pesan kesalahan yang spesifik. Notebook yang baik tidak hanya bekerja pada kondisi ideal, tetapi juga gagal secara aman ketika input atau lingkungan tidak memenuhi syarat. Tambahkan satu sel diagnostik pada awal notebook dan satu sel manifest pada akhir proyek.

G.24 Koordinat XYZ kosong

Gejala dan Bukti Awal

Gejala yang terlihat adalah koordinat xyz kosong. Jangan langsung mengubah banyak sel. Salin pesan kesalahan lengkap, catat nomor sel dan waktu kejadian, serta periksa apakah masalah dapat diulang setelah Restart session. Bukti awal yang berguna meliputi versi Python dan paket, direktori kerja, daftar berkas, shape dan dtype data, parameter kimia, status CPU/GPU, serta bagian traceback yang menunjuk baris pertama dari kode pengguna.

Penyebab yang Paling Mungkin

Penyebab utama yang perlu diuji adalah: Konformer belum dibuat. Penyebab tersebut harus diperlakukan sebagai hipotesis. Periksa prasyarat paling sederhana terlebih dahulu, lalu bergerak ke dependensi dan algoritma. Apabila beberapa penyebab mungkin, buat uji minimal yang mengisolasi satu komponen, misalnya hanya mengimpor modul, hanya membaca satu berkas, atau menjalankan molekul kecil.

Langkah Pemulihan

1376. Simpan notebook dan artefak yang masih tersedia sebelum melakukan restart atau instalasi ulang.
1377. Buat uji minimal untuk memastikan gejala dapat direproduksi tanpa sel yang tidak berkaitan.
1378. Pastikan EmbedMolecule sukses dan conformer tersedia.
1379. Jalankan kembali seluruh notebook dari sesi bersih dan bandingkan status serta keluaran dengan catatan sebelum perbaikan.
1380. Dokumentasikan akar masalah, perubahan yang dilakukan, versi, dan tindakan pencegahan.

Pencegahan

Pencegahan dilakukan dengan memusatkan parameter, menggunakan validasi eksplisit, mencatat versi, menyimpan checkpoint, dan menyediakan pesan kesalahan yang spesifik. Notebook yang baik tidak hanya bekerja pada kondisi ideal, tetapi juga gagal secara aman ketika input atau lingkungan tidak memenuhi syarat. Tambahkan satu sel diagnostik pada awal notebook dan satu sel manifest pada akhir proyek.

G.25 py3Dmol tidak menampilkan struktur

Gejala dan Bukti Awal

Gejala yang terlihat adalah py3dmol tidak menampilkan struktur. Jangan langsung mengubah banyak sel. Salin pesan kesalahan lengkap, catat nomor sel dan waktu kejadian, serta periksa apakah masalah dapat diulang setelah Restart session. Bukti awal yang berguna meliputi versi Python dan paket, direktori kerja, daftar berkas, shape dan dtype data, parameter kimia, status CPU/GPU, serta bagian traceback yang menunjuk baris pertama dari kode pengguna.

Penyebab yang Paling Mungkin

Penyebab utama yang perlu diuji adalah: Format model atau teks XYZ salah. Penyebab tersebut harus diperlakukan sebagai hipotesis. Periksa prasyarat paling sederhana terlebih dahulu, lalu bergerak ke dependensi dan algoritma. Apabila beberapa penyebab mungkin, buat uji minimal yang mengisolasi satu komponen, misalnya hanya mengimpor modul, hanya membaca satu berkas, atau menjalankan molekul kecil.

Langkah Pemulihan

1381. Simpan notebook dan artefak yang masih tersedia sebelum melakukan restart atau instalasi ulang.
1382. Buat uji minimal untuk memastikan gejala dapat direproduksi tanpa sel yang tidak berkaitan.
1383. Periksa jumlah atom, baris koordinat, dan jenis model.
1384. Jalankan kembali seluruh notebook dari sesi bersih dan bandingkan status serta keluaran dengan catatan sebelum perbaikan.
1385. Dokumentasikan akar masalah, perubahan yang dilakukan, versi, dan tindakan pencegahan.

Pencegahan

Pencegahan dilakukan dengan memusatkan parameter, menggunakan validasi eksplisit, mencatat versi, menyimpan checkpoint, dan menyediakan pesan kesalahan yang spesifik. Notebook yang baik tidak hanya bekerja pada kondisi ideal, tetapi juga gagal secara aman ketika input atau lingkungan tidak memenuhi syarat. Tambahkan satu sel diagnostik pada awal notebook dan satu sel manifest pada akhir proyek.

G.26 PySCF gagal membangun molekul

Gejala dan Bukti Awal

Gejala yang terlihat adalah pyscf gagal membangun molekul. Jangan langsung mengubah banyak sel. Salin pesan kesalahan lengkap, catat nomor sel dan waktu kejadian, serta periksa apakah masalah dapat diulang setelah Restart session. Bukti awal yang berguna

meliputi versi Python dan paket, direktori kerja, daftar berkas, shape dan dtype data, parameter kimia, status CPU/GPU, serta bagian traceback yang menunjuk baris pertama dari kode pengguna.

Penyebab yang Paling Mungkin

Penyebab utama yang perlu diuji adalah: Koordinat, basis, charge, atau spin tidak konsisten. Penyebab tersebut harus diperlakukan sebagai hipotesis. Periksa prasyarat paling sederhana terlebih dahulu, lalu bergerak ke dependensi dan algoritma. Apabila beberapa penyebab mungkin, buat uji minimal yang mengisolasi satu komponen, misalnya hanya mengimpor modul, hanya membaca satu berkas, atau menjalankan molekul kecil.

Langkah Pemulihan

1386. Simpan notebook dan artefak yang masih tersedia sebelum melakukan restart atau instalasi ulang.
1387. Buat uji minimal untuk memastikan gejala dapat direproduksi tanpa sel yang tidak berkaitan.
1388. Cetak jumlah elektron, periksa simbol atom, dan validasi parity spin.
1389. Jalankan kembali seluruh notebook dari sesi bersih dan bandingkan status serta keluaran dengan catatan sebelum perbaikan.
1390. Dokumentasikan akar masalah, perubahan yang dilakukan, versi, dan tindakan pencegahan.

Pencegahan

Pencegahan dilakukan dengan memusatkan parameter, menggunakan validasi eksplisit, mencatat versi, menyimpan checkpoint, dan menyediakan pesan kesalahan yang spesifik. Notebook yang baik tidak hanya bekerja pada kondisi ideal, tetapi juga gagal secara aman ketika input atau lingkungan tidak memenuhi syarat. Tambahkan satu sel diagnostik pada awal notebook dan satu sel manifest pada akhir proyek.

G.27 SCF tidak konvergen

Gejala dan Bukti Awal

Gejala yang terlihat adalah scf tidak konvergen. Jangan langsung mengubah banyak sel. Salin pesan kesalahan lengkap, catat nomor sel dan waktu kejadian, serta periksa apakah masalah dapat diulang setelah Restart session. Bukti awal yang berguna meliputi versi Python dan paket, direktori kerja, daftar berkas, shape dan dtype data, parameter kimia, status CPU/GPU, serta bagian traceback yang menunjuk baris pertama dari kode pengguna.

Penyebab yang Paling Mungkin

Penyebab utama yang perlu diuji adalah: Tebakan awal buruk atau sistem sulit. Penyebab tersebut harus diperlakukan sebagai hipotesis. Periksa prasyarat paling sederhana terlebih dahulu, lalu bergerak ke dependensi dan algoritma. Apabila beberapa penyebab mungkin, buat uji minimal yang mengisolasi satu komponen, misalnya hanya mengimpor modul, hanya membaca satu berkas, atau menjalankan molekul kecil.

Langkah Pemulihan

1391. Simpan notebook dan artefak yang masih tersedia sebelum melakukan restart atau instalasi ulang.
1392. Buat uji minimal untuk memastikan gejala dapat direproduksi tanpa sel yang tidak berkaitan.
1393. Naikkan `max_cycle`, ubah `initial guess`, gunakan `damping/level shift` atau `Newton`.
1394. Jalankan kembali seluruh notebook dari sesi bersih dan bandingkan status serta keluaran dengan catatan sebelum perbaikan.
1395. Dokumentasikan akar masalah, perubahan yang dilakukan, versi, dan tindakan pencegahan.

Pencegahan

Pencegahan dilakukan dengan memusatkan parameter, menggunakan validasi eksplisit, mencatat versi, menyimpan checkpoint, dan menyediakan pesan kesalahan yang spesifik. Notebook yang baik tidak hanya bekerja pada kondisi ideal, tetapi juga gagal secara aman ketika input atau lingkungan tidak memenuhi syarat. Tambahkan satu sel diagnostik pada awal notebook dan satu sel manifest pada akhir proyek.

G.28 Energi tampak tidak masuk akal

Gejala dan Bukti Awal

Gejala yang terlihat adalah energi tampak tidak masuk akal. Jangan langsung mengubah banyak sel. Salin pesan kesalahan lengkap, catat nomor sel dan waktu kejadian, serta periksa apakah masalah dapat diulang setelah Restart session. Bukti awal yang berguna meliputi versi Python dan paket, direktori kerja, daftar berkas, shape dan dtype data, parameter kimia, status CPU/GPU, serta bagian traceback yang menunjuk baris pertama dari kode pengguna.

Penyebab yang Paling Mungkin

Penyebab utama yang perlu diuji adalah: Satuan, geometri, charge, spin, atau metode salah. Penyebab tersebut harus diperlakukan sebagai hipotesis. Periksa prasyarat paling sederhana terlebih dahulu, lalu bergerak ke dependensi dan algoritma. Apabila beberapa penyebab mungkin, buat uji minimal yang mengisolasi satu komponen, misalnya hanya mengimpor modul, hanya membaca satu berkas, atau menjalankan molekul kecil.

Langkah Pemulihan

1396. Simpan notebook dan artefak yang masih tersedia sebelum melakukan restart atau instalasi ulang.
1397. Buat uji minimal untuk memastikan gejala dapat direproduksi tanpa sel yang tidak berkaitan.
1398. Cetak seluruh parameter dan bandingkan dengan kasus minimal.
1399. Jalankan kembali seluruh notebook dari sesi bersih dan bandingkan status serta keluaran dengan catatan sebelum perbaikan.
1400. Dokumentasikan akar masalah, perubahan yang dilakukan, versi, dan tindakan pencegahan.

Pencegahan

Pencegahan dilakukan dengan memusatkan parameter, menggunakan validasi eksplisit, mencatat versi, menyimpan checkpoint, dan menyediakan pesan kesalahan yang spesifik. Notebook yang baik tidak hanya bekerja pada kondisi ideal, tetapi juga gagal secara aman ketika input atau lingkungan tidak memenuhi syarat. Tambahkan satu sel diagnostik pada awal notebook dan satu sel manifest pada akhir proyek.

G.29 Optimasi geometri berhenti

Gejala dan Bukti Awal

Gejala yang terlihat adalah optimasi geometri berhenti. Jangan langsung mengubah banyak sel. Salin pesan kesalahan lengkap, catat nomor sel dan waktu kejadian, serta periksa apakah masalah dapat diulang setelah Restart session. Bukti awal yang berguna meliputi versi Python dan paket, direktori kerja, daftar berkas, shape dan dtype data, parameter kimia, status CPU/GPU, serta bagian traceback yang menunjuk baris pertama dari kode pengguna.

Penyebab yang Paling Mungkin

Penyebab utama yang perlu diuji adalah: SCF pada salah satu langkah gagal atau maxsteps habis. Penyebab tersebut harus diperlakukan sebagai hipotesis. Periksa prasyarat paling sederhana terlebih dahulu, lalu bergerak ke dependensi dan algoritma. Apabila beberapa penyebab mungkin, buat uji minimal yang mengisolasi satu komponen, misalnya hanya mengimpor modul, hanya membaca satu berkas, atau menjalankan molekul kecil.

Langkah Pemulihan

1401. Simpan notebook dan artefak yang masih tersedia sebelum melakukan restart atau instalasi ulang.
1402. Buat uji minimal untuk memastikan gejala dapat direproduksi tanpa sel yang tidak berkaitan.

1403. Periksa log langkah, perbaiki struktur awal, dan tingkatkan batas secara wajar.
1404. Jalankan kembali seluruh notebook dari sesi bersih dan bandingkan status serta keluaran dengan catatan sebelum perbaikan.
1405. Dokumentasikan akar masalah, perubahan yang dilakukan, versi, dan tindakan pencegahan.

Pencegahan

Pencegahan dilakukan dengan memusatkan parameter, menggunakan validasi eksplisit, mencatat versi, menyimpan checkpoint, dan menyediakan pesan kesalahan yang spesifik. Notebook yang baik tidak hanya bekerja pada kondisi ideal, tetapi juga gagal secara aman ketika input atau lingkungan tidak memenuhi syarat. Tambahkan satu sel diagnostik pada awal notebook dan satu sel manifest pada akhir proyek.

G.30 Optimasi selesai tetapi geometri aneh

Gejala dan Bukti Awal

Gejala yang terlihat adalah optimasi selesai tetapi geometri aneh. Jangan langsung mengubah banyak sel. Salin pesan kesalahan lengkap, catat nomor sel dan waktu kejadian, serta periksa apakah masalah dapat diulang setelah Restart session. Bukti awal yang berguna meliputi versi Python dan paket, direktori kerja, daftar berkas, shape dan dtype data, parameter kimia, status CPU/GPU, serta bagian traceback yang menunjuk baris pertama dari kode pengguna.

Penyebab yang Paling Mungkin

Penyebab utama yang perlu diuji adalah: Minimum lokal buruk atau input tidak realistis. Penyebab tersebut harus diperlakukan sebagai hipotesis. Periksa prasyarat paling sederhana terlebih dahulu, lalu bergerak ke dependensi dan algoritma. Apabila beberapa penyebab mungkin, buat uji minimal yang mengisolasi satu komponen, misalnya hanya mengimpor modul, hanya membaca satu berkas, atau menjalankan molekul kecil.

Langkah Pemulihan

1406. Simpan notebook dan artefak yang masih tersedia sebelum melakukan restart atau instalasi ulang.
1407. Buat uji minimal untuk memastikan gejala dapat direproduksi tanpa sel yang tidak berkaitan.
1408. Visualisasikan, periksa panjang ikatan, dan mulai dari konformer lebih baik.
1409. Jalankan kembali seluruh notebook dari sesi bersih dan bandingkan status serta keluaran dengan catatan sebelum perbaikan.
1410. Dokumentasikan akar masalah, perubahan yang dilakukan, versi, dan tindakan pencegahan.

Pencegahan

Pencegahan dilakukan dengan memusatkan parameter, menggunakan validasi eksplisit, mencatat versi, menyimpan checkpoint, dan menyediakan pesan kesalahan yang spesifik. Notebook yang baik tidak hanya bekerja pada kondisi ideal, tetapi juga gagal secara aman ketika input atau lingkungan tidak memenuhi syarat. Tambahkan satu sel diagnostik pada awal notebook dan satu sel manifest pada akhir proyek.

G.31 Frekuensi imajiner

Gejala dan Bukti Awal

Gejala yang terlihat adalah frekuensi imajiner. Jangan langsung mengubah banyak sel. Salin pesan kesalahan lengkap, catat nomor sel dan waktu kejadian, serta periksa apakah masalah dapat diulang setelah Restart session. Bukti awal yang berguna meliputi versi Python dan paket, direktori kerja, daftar berkas, shape dan dtype data, parameter kimia, status CPU/GPU, serta bagian traceback yang menunjuk baris pertama dari kode pengguna.

Penyebab yang Paling Mungkin

Penyebab utama yang perlu diuji adalah: Struktur bukan minimum atau mode translasi/rotasi numerik. Penyebab tersebut harus diperlakukan sebagai hipotesis. Periksa prasyarat paling sederhana terlebih dahulu, lalu bergerak ke dependensi dan algoritma. Apabila beberapa penyebab mungkin, buat uji minimal yang mengisolasi satu komponen, misalnya hanya mengimpor modul, hanya membaca satu berkas, atau menjalankan molekul kecil.

Langkah Pemulihan

1411. Simpan notebook dan artefak yang masih tersedia sebelum melakukan restart atau instalasi ulang.
1412. Buat uji minimal untuk memastikan gejala dapat direproduksi tanpa sel yang tidak berkaitan.
1413. Periksa besar frekuensi, visualisasikan mode, dan optimasi ulang dengan kriteria lebih ketat.
1414. Jalankan kembali seluruh notebook dari sesi bersih dan bandingkan status serta keluaran dengan catatan sebelum perbaikan.
1415. Dokumentasikan akar masalah, perubahan yang dilakukan, versi, dan tindakan pencegahan.

Pencegahan

Pencegahan dilakukan dengan memusatkan parameter, menggunakan validasi eksplisit, mencatat versi, menyimpan checkpoint, dan menyediakan pesan kesalahan yang spesifik. Notebook yang baik tidak hanya bekerja pada kondisi ideal, tetapi juga gagal secara aman

ketika input atau lingkungan tidak memenuhi syarat. Tambahkan satu sel diagnostik pada awal notebook dan satu sel manifest pada akhir proyek.

G.32 HOMO/LUMO salah indeks

Gejala dan Bukti Awal

Gejala yang terlihat adalah homo/lumo salah indeks. Jangan langsung mengubah banyak sel. Salin pesan kesalahan lengkap, catat nomor sel dan waktu kejadian, serta periksa apakah masalah dapat diulang setelah Restart session. Bukti awal yang berguna meliputi versi Python dan paket, direktori kerja, daftar berkas, shape dan dtype data, parameter kimia, status CPU/GPU, serta bagian traceback yang menunjuk baris pertama dari kode pengguna.

Penyebab yang Paling Mungkin

Penyebab utama yang perlu diuji adalah: Okupansi orbital tidak diperiksa. Penyebab tersebut harus diperlakukan sebagai hipotesis. Periksa prasyarat paling sederhana terlebih dahulu, lalu bergerak ke dependensi dan algoritma. Apabila beberapa penyebab mungkin, buat uji minimal yang mengisolasi satu komponen, misalnya hanya mengimpor modul, hanya membaca satu berkas, atau menjalankan molekul kecil.

Langkah Pemulihan

1416. Simpan notebook dan artefak yang masih tersedia sebelum melakukan restart atau instalasi ulang.
1417. Buat uji minimal untuk memastikan gejala dapat direproduksi tanpa sel yang tidak berkaitan.
1418. Tentukan indeks dari `mo_occ`, terutama untuk open-shell.
1419. Jalankan kembali seluruh notebook dari sesi bersih dan bandingkan status serta keluaran dengan catatan sebelum perbaikan.
1420. Dokumentasikan akar masalah, perubahan yang dilakukan, versi, dan tindakan pencegahan.

Pencegahan

Pencegahan dilakukan dengan memusatkan parameter, menggunakan validasi eksplisit, mencatat versi, menyimpan checkpoint, dan menyediakan pesan kesalahan yang spesifik. Notebook yang baik tidak hanya bekerja pada kondisi ideal, tetapi juga gagal secara aman ketika input atau lingkungan tidak memenuhi syarat. Tambahkan satu sel diagnostik pada awal notebook dan satu sel manifest pada akhir proyek.

G.33 Momen dipol berubah tanda

Gejala dan Bukti Awal

Gejala yang terlihat adalah momen dipol berubah tanda. Jangan langsung mengubah banyak sel. Salin pesan kesalahan lengkap, catat nomor sel dan waktu kejadian, serta periksa apakah masalah dapat diulang setelah Restart session. Bukti awal yang berguna meliputi versi Python dan paket, direktori kerja, daftar berkas, shape dan dtype data, parameter kimia, status CPU/GPU, serta bagian traceback yang menunjuk baris pertama dari kode pengguna.

Penyebab yang Paling Mungkin

Penyebab utama yang perlu diuji adalah: Orientasi koordinat berubah. Penyebab tersebut harus diperlakukan sebagai hipotesis. Periksa prasyarat paling sederhana terlebih dahulu, lalu bergerak ke dependensi dan algoritma. Apabila beberapa penyebab mungkin, buat uji minimal yang mengisolasi satu komponen, misalnya hanya mengimpor modul, hanya membaca satu berkas, atau menjalankan molekul kecil.

Langkah Pemulihan

1421. Simpan notebook dan artefak yang masih tersedia sebelum melakukan restart atau instalasi ulang.
1422. Buat uji minimal untuk memastikan gejala dapat direproduksi tanpa sel yang tidak berkaitan.
1423. Bandingkan besar vektor, bukan tanda komponen saja.
1424. Jalankan kembali seluruh notebook dari sesi bersih dan bandingkan status serta keluaran dengan catatan sebelum perbaikan.
1425. Dokumentasikan akar masalah, perubahan yang dilakukan, versi, dan tindakan pencegahan.

Pencegahan

Pencegahan dilakukan dengan memusatkan parameter, menggunakan validasi eksplisit, mencatat versi, menyimpan checkpoint, dan menyediakan pesan kesalahan yang spesifik. Notebook yang baik tidak hanya bekerja pada kondisi ideal, tetapi juga gagal secara aman ketika input atau lingkungan tidak memenuhi syarat. Tambahkan satu sel diagnostik pada awal notebook dan satu sel manifest pada akhir proyek.

G.34 Hasil CPU dan GPU sedikit berbeda

Gejala dan Bukti Awal

Gejala yang terlihat adalah hasil cpu dan gpu sedikit berbeda. Jangan langsung mengubah banyak sel. Salin pesan kesalahan lengkap, catat nomor sel dan waktu kejadian, serta periksa apakah masalah dapat diulang setelah Restart session. Bukti awal yang berguna

meliputi versi Python dan paket, direktori kerja, daftar berkas, shape dan dtype data, parameter kimia, status CPU/GPU, serta bagian traceback yang menunjuk baris pertama dari kode pengguna.

Penyebab yang Paling Mungkin

Penyebab utama yang perlu diuji adalah: Urutan operasi dan presisi numerik berbeda. Penyebab tersebut harus diperlakukan sebagai hipotesis. Periksa prasyarat paling sederhana terlebih dahulu, lalu bergerak ke dependensi dan algoritma. Apabila beberapa penyebab mungkin, buat uji minimal yang mengisolasi satu komponen, misalnya hanya mengimpor modul, hanya membaca satu berkas, atau menjalankan molekul kecil.

Langkah Pemulihan

1426. Simpan notebook dan artefak yang masih tersedia sebelum melakukan restart atau instalasi ulang.
1427. Buat uji minimal untuk memastikan gejala dapat direproduksi tanpa sel yang tidak berkaitan.
1428. Bandingkan toleransi, digit signifikan, dan kriteria konvergensi.
1429. Jalankan kembali seluruh notebook dari sesi bersih dan bandingkan status serta keluaran dengan catatan sebelum perbaikan.
1430. Dokumentasikan akar masalah, perubahan yang dilakukan, versi, dan tindakan pencegahan.

Pencegahan

Pencegahan dilakukan dengan memusatkan parameter, menggunakan validasi eksplisit, mencatat versi, menyimpan checkpoint, dan menyediakan pesan kesalahan yang spesifik. Notebook yang baik tidak hanya bekerja pada kondisi ideal, tetapi juga gagal secara aman ketika input atau lingkungan tidak memenuhi syarat. Tambahkan satu sel diagnostik pada awal notebook dan satu sel manifest pada akhir proyek.

G.35 Notebook hanya berjalan sebagian

Gejala dan Bukti Awal

Gejala yang terlihat adalah notebook hanya berjalan sebagian. Jangan langsung mengubah banyak sel. Salin pesan kesalahan lengkap, catat nomor sel dan waktu kejadian, serta periksa apakah masalah dapat diulang setelah Restart session. Bukti awal yang berguna meliputi versi Python dan paket, direktori kerja, daftar berkas, shape dan dtype data, parameter kimia, status CPU/GPU, serta bagian traceback yang menunjuk baris pertama dari kode pengguna.

Penyebab yang Paling Mungkin

Penyebab utama yang perlu diuji adalah: Sel tidak berurutan atau variabel tersembunyi. Penyebab tersebut harus diperlakukan sebagai hipotesis. Periksa prasyarat paling sederhana terlebih dahulu, lalu bergerak ke dependensi dan algoritma. Apabila beberapa penyebab mungkin, buat uji minimal yang mengisolasi satu komponen, misalnya hanya mengimpor modul, hanya membaca satu berkas, atau menjalankan molekul kecil.

Langkah Pemulihan

1431. Simpan notebook dan artefak yang masih tersedia sebelum melakukan restart atau instalasi ulang.
1432. Buat uji minimal untuk memastikan gejala dapat direproduksi tanpa sel yang tidak berkaitan.
1433. Restart dan Run all; pindahkan konfigurasi ke awal.
1434. Jalankan kembali seluruh notebook dari sesi bersih dan bandingkan status serta keluaran dengan catatan sebelum perbaikan.
1435. Dokumentasikan akar masalah, perubahan yang dilakukan, versi, dan tindakan pencegahan.

Pencegahan

Pencegahan dilakukan dengan memusatkan parameter, menggunakan validasi eksplisit, mencatat versi, menyimpan checkpoint, dan menyediakan pesan kesalahan yang spesifik. Notebook yang baik tidak hanya bekerja pada kondisi ideal, tetapi juga gagal secara aman ketika input atau lingkungan tidak memenuhi syarat. Tambahkan satu sel diagnostik pada awal notebook dan satu sel manifest pada akhir proyek.

G.36 Output lama tertinggal

Gejala dan Bukti Awal

Gejala yang terlihat adalah output lama tertinggal. Jangan langsung mengubah banyak sel. Salin pesan kesalahan lengkap, catat nomor sel dan waktu kejadian, serta periksa apakah masalah dapat diulang setelah Restart session. Bukti awal yang berguna meliputi versi Python dan paket, direktori kerja, daftar berkas, shape dan dtype data, parameter kimia, status CPU/GPU, serta bagian traceback yang menunjuk baris pertama dari kode berguna.

Penyebab yang Paling Mungkin

Penyebab utama yang perlu diuji adalah: Sel belum dijalankan ulang atau output tidak dibersihkan. Penyebab tersebut harus diperlakukan sebagai hipotesis. Periksa prasyarat paling sederhana terlebih dahulu, lalu bergerak ke dependensi dan algoritma. Apabila beberapa penyebab mungkin, buat uji minimal yang mengisolasi satu komponen,

misalnya hanya mengimpor modul, hanya membaca satu berkas, atau menjalankan molekul kecil.

Langkah Pemulihan

1436. Simpan notebook dan artefak yang masih tersedia sebelum melakukan restart atau instalasi ulang.
1437. Buat uji minimal untuk memastikan gejala dapat direproduksi tanpa sel yang tidak berkaitan.
1438. Clear all outputs dan jalankan dari awal.
1439. Jalankan kembali seluruh notebook dari sesi bersih dan bandingkan status serta keluaran dengan catatan sebelum perbaikan.
1440. Dokumentasikan akar masalah, perubahan yang dilakukan, versi, dan tindakan pencegahan.

Pencegahan

Pencegahan dilakukan dengan memusatkan parameter, menggunakan validasi eksplisit, mencatat versi, menyimpan checkpoint, dan menyediakan pesan kesalahan yang spesifik. Notebook yang baik tidak hanya bekerja pada kondisi ideal, tetapi juga gagal secara aman ketika input atau lingkungan tidak memenuhi syarat. Tambahkan satu sel diagnostik pada awal notebook dan satu sel manifest pada akhir proyek.

G.37 Versi notebook tidak jelas

Gejala dan Bukti Awal

Gejala yang terlihat adalah versi notebook tidak jelas. Jangan langsung mengubah banyak sel. Salin pesan kesalahan lengkap, catat nomor sel dan waktu kejadian, serta periksa apakah masalah dapat diulang setelah Restart session. Bukti awal yang berguna meliputi versi Python dan paket, direktori kerja, daftar berkas, shape dan dtype data, parameter kimia, status CPU/GPU, serta bagian traceback yang menunjuk baris pertama dari kode pengguna.

Penyebab yang Paling Mungkin

Penyebab utama yang perlu diuji adalah: Metadata tidak dicatat. Penyebab tersebut harus diperlakukan sebagai hipotesis. Periksa prasyarat paling sederhana terlebih dahulu, lalu bergerak ke dependensi dan algoritma. Apabila beberapa penyebab mungkin, buat uji minimal yang mengisolasi satu komponen, misalnya hanya mengimpor modul, hanya membaca satu berkas, atau menjalankan molekul kecil.

Langkah Pemulihan

1441. Simpan notebook dan artefak yang masih tersedia sebelum melakukan restart atau instalasi ulang.

1442. Buat uji minimal untuk memastikan gejala dapat direproduksi tanpa sel yang tidak berkaitan.
1443. Cetak versi Python, paket, commit atau tanggal, dan parameter.
1444. Jalankan kembali seluruh notebook dari sesi bersih dan bandingkan status serta keluaran dengan catatan sebelum perbaikan.
1445. Dokumentasikan akar masalah, perubahan yang dilakukan, versi, dan tindakan pencegahan.

Pencegahan

Pencegahan dilakukan dengan memusatkan parameter, menggunakan validasi eksplisit, mencatat versi, menyimpan checkpoint, dan menyediakan pesan kesalahan yang spesifik. Notebook yang baik tidak hanya bekerja pada kondisi ideal, tetapi juga gagal secara aman ketika input atau lingkungan tidak memenuhi syarat. Tambahkan satu sel diagnostik pada awal notebook dan satu sel manifest pada akhir proyek.

G.38 ZIP tidak memuat semua hasil

Gejala dan Bukti Awal

Gejala yang terlihat adalah zip tidak memuat semua hasil. Jangan langsung mengubah banyak sel. Salin pesan kesalahan lengkap, catat nomor sel dan waktu kejadian, serta periksa apakah masalah dapat diulang setelah Restart session. Bukti awal yang berguna meliputi versi Python dan paket, direktori kerja, daftar berkas, shape dan dtype data, parameter kimia, status CPU/GPU, serta bagian traceback yang menunjuk baris pertama dari kode pengguna.

Penyebab yang Paling Mungkin

Penyebab utama yang perlu diuji adalah: Pola glob atau direktori salah. Penyebab tersebut harus diperlakukan sebagai hipotesis. Periksa prasyarat paling sederhana terlebih dahulu, lalu bergerak ke dependensi dan algoritma. Apabila beberapa penyebab mungkin, buat uji minimal yang mengisolasi satu komponen, misalnya hanya mengimpor modul, hanya membaca satu berkas, atau menjalankan molekul kecil.

Langkah Pemulihan

1446. Simpan notebook dan artefak yang masih tersedia sebelum melakukan restart atau instalasi ulang.
1447. Buat uji minimal untuk memastikan gejala dapat direproduksi tanpa sel yang tidak berkaitan.
1448. Cetak daftar berkas sebelum kompresi dan gunakan manifest.
1449. Jalankan kembali seluruh notebook dari sesi bersih dan bandingkan status serta keluaran dengan catatan sebelum perbaikan.
1450. Dokumentasikan akar masalah, perubahan yang dilakukan, versi, dan tindakan pencegahan.

Pencegahan

Pencegahan dilakukan dengan memusatkan parameter, menggunakan validasi eksplisit, mencatat versi, menyimpan checkpoint, dan menyediakan pesan kesalahan yang spesifik. Notebook yang baik tidak hanya bekerja pada kondisi ideal, tetapi juga gagal secara aman ketika input atau lingkungan tidak memenuhi syarat. Tambahkan satu sel diagnostik pada awal notebook dan satu sel manifest pada akhir proyek.

G.39 JSON gagal disimpan

Gejala dan Bukti Awal

Gejala yang terlihat adalah json gagal disimpan. Jangan langsung mengubah banyak sel. Salin pesan kesalahan lengkap, catat nomor sel dan waktu kejadian, serta periksa apakah masalah dapat diulang setelah Restart session. Bukti awal yang berguna meliputi versi Python dan paket, direktori kerja, daftar berkas, shape dan dtype data, parameter kimia, status CPU/GPU, serta bagian traceback yang menunjuk baris pertama dari kode pengguna.

Penyebab yang Paling Mungkin

Penyebab utama yang perlu diuji adalah: Ada tipe NumPy atau objek non-serializable. Penyebab tersebut harus diperlakukan sebagai hipotesis. Periksa prasyarat paling sederhana terlebih dahulu, lalu bergerak ke dependensi dan algoritma. Apabila beberapa penyebab mungkin, buat uji minimal yang mengisolasi satu komponen, misalnya hanya mengimpor modul, hanya membaca satu berkas, atau menjalankan molekul kecil.

Langkah Pemulihan

1451. Simpan notebook dan artefak yang masih tersedia sebelum melakukan restart atau instalasi ulang.
1452. Buat uji minimal untuk memastikan gejala dapat direproduksi tanpa sel yang tidak berkaitan.
1453. Konversi ke int, float, list, atau string secara eksplisit.
1454. Jalankan kembali seluruh notebook dari sesi bersih dan bandingkan status serta keluaran dengan catatan sebelum perbaikan.
1455. Dokumentasikan akar masalah, perubahan yang dilakukan, versi, dan tindakan pencegahan.

Pencegahan

Pencegahan dilakukan dengan memusatkan parameter, menggunakan validasi eksplisit, mencatat versi, menyimpan checkpoint, dan menyediakan pesan kesalahan yang spesifik. Notebook yang baik tidak hanya bekerja pada kondisi ideal, tetapi juga gagal secara aman ketika input atau lingkungan tidak memenuhi syarat. Tambahkan satu sel diagnostik pada awal notebook dan satu sel manifest pada akhir proyek.

G.40 Waktu eksekusi sangat berubah

Gejala dan Bukti Awal

Gejala yang terlihat adalah waktu eksekusi sangat berubah. Jangan langsung mengubah banyak sel. Salin pesan kesalahan lengkap, catat nomor sel dan waktu kejadian, serta periksa apakah masalah dapat diulang setelah Restart session. Bukti awal yang berguna meliputi versi Python dan paket, direktori kerja, daftar berkas, shape dan dtype data, parameter kimia, status CPU/GPU, serta bagian traceback yang menunjuk baris pertama dari kode pengguna.

Penyebab yang Paling Mungkin

Penyebab utama yang perlu diuji adalah: Beban runtime bersama, cache, atau backend berbeda. Penyebab tersebut harus diperlakukan sebagai hipotesis. Periksa prasyarat paling sederhana terlebih dahulu, lalu bergerak ke dependensi dan algoritma. Apabila beberapa penyebab mungkin, buat uji minimal yang mengisolasi satu komponen, misalnya hanya mengimpor modul, hanya membaca satu berkas, atau menjalankan molekul kecil.

Langkah Pemulihan

1456. Simpan notebook dan artefak yang masih tersedia sebelum melakukan restart atau instalasi ulang.
1457. Buat uji minimal untuk memastikan gejala dapat direproduksi tanpa sel yang tidak berkaitan.
1458. Lakukan pengulangan, gunakan median, dan catat kondisi perangkat.
1459. Jalankan kembali seluruh notebook dari sesi bersih dan bandingkan status serta keluaran dengan catatan sebelum perbaikan.
1460. Dokumentasikan akar masalah, perubahan yang dilakukan, versi, dan tindakan pencegahan.

Pencegahan

Pencegahan dilakukan dengan memusatkan parameter, menggunakan validasi eksplisit, mencatat versi, menyimpan checkpoint, dan menyediakan pesan kesalahan yang spesifik. Notebook yang baik tidak hanya bekerja pada kondisi ideal, tetapi juga gagal secara aman ketika input atau lingkungan tidak memenuhi syarat. Tambahkan satu sel diagnostik pada awal notebook dan satu sel manifest pada akhir proyek.

LAMPIRAN H - KAMUS 200 PERINTAH DAN POLA KODE PENTING

Kamus ini berfungsi sebagai rujukan cepat. Setiap entri memuat fungsi utama, contoh singkat, dan catatan penggunaan. Contoh sengaja dibuat ringkas agar pembaca dapat menggabungkannya ke dalam notebook yang lebih lengkap. Sebelum menggunakan perintah pada data nyata, periksa dokumentasi versi pustaka yang sedang terpasang dan uji dengan input kecil.

H.Python

H.001 print()

Fungsi utama: Menampilkan teks atau nilai ke output sel. Perintah ini sebaiknya ditempatkan pada tahap notebook yang jelas dan disertai pemeriksaan input. Contoh minimal: `print("Status: berhasil")`. Setelah menjalankan contoh, periksa tipe, nilai, status, atau berkas yang dihasilkan. Jangan menafsirkan keluaran di luar konteks data dan metode.

Catatan penggunaan: tuliskan alasan pemakaian perintah, bukan hanya sintaksnya. Apabila perintah bergantung pada paket tambahan, catat versi paket dan sediakan sel instalasi. Bila perintah dapat gagal, tangkap pengecualian yang spesifik atau periksa nilai balik. Untuk proses yang menghasilkan berkas, verifikasi keberadaan, ukuran, encoding, dan kemampuan pembacaan ulang.

H.002 type()

Fungsi utama: Memeriksa tipe objek. Perintah ini sebaiknya ditempatkan pada tahap notebook yang jelas dan disertai pemeriksaan input. Contoh minimal: `print(type(nilai))`. Setelah menjalankan contoh, periksa tipe, nilai, status, atau berkas yang dihasilkan. Jangan menafsirkan keluaran di luar konteks data dan metode.

Catatan penggunaan: tuliskan alasan pemakaian perintah, bukan hanya sintaksnya. Apabila perintah bergantung pada paket tambahan, catat versi paket dan sediakan sel instalasi. Bila perintah dapat gagal, tangkap pengecualian yang spesifik atau periksa nilai balik. Untuk proses yang menghasilkan berkas, verifikasi keberadaan, ukuran, encoding, dan kemampuan pembacaan ulang.

H.003 isinstance()

Fungsi utama: Menguji apakah objek termasuk tipe tertentu. Perintah ini sebaiknya ditempatkan pada tahap notebook yang jelas dan disertai pemeriksaan input. Contoh minimal: `assert isinstance(spin, int)`. Setelah menjalankan contoh, periksa tipe, nilai, status, atau berkas yang dihasilkan. Jangan menafsirkan keluaran di luar konteks data dan metode.

Catatan penggunaan: tuliskan alasan pemakaian perintah, bukan hanya sintaksnya. Apabila perintah bergantung pada paket tambahan, catat versi paket dan sediakan sel instalasi. Bila perintah dapat gagal, tangkap pengecualian yang spesifik atau periksa nilai balik. Untuk proses yang menghasilkan berkas, verifikasi keberadaan, ukuran, encoding, dan kemampuan pembacaan ulang.

H.004 len()

Fungsi utama: Menghitung jumlah elemen. Perintah ini sebaiknya ditempatkan pada tahap notebook yang jelas dan disertai pemeriksaan input. Contoh minimal: `n = len(data)`. Setelah menjalankan contoh, periksa tipe, nilai, status, atau berkas yang dihasilkan. Jangan menafsirkan keluaran di luar konteks data dan metode.

Catatan penggunaan: tuliskan alasan pemakaian perintah, bukan hanya sintaksnya. Apabila perintah bergantung pada paket tambahan, catat versi paket dan sediakan sel instalasi. Bila perintah dapat gagal, tangkap pengecualian yang spesifik atau periksa nilai balik. Untuk proses yang menghasilkan berkas, verifikasi keberadaan, ukuran, encoding, dan kemampuan pembacaan ulang.

H.005 range()

Fungsi utama: Membuat rentang integer untuk perulangan. Perintah ini sebaiknya ditempatkan pada tahap notebook yang jelas dan disertai pemeriksaan input. Contoh minimal: `for i in range(5): pass`. Setelah menjalankan contoh, periksa tipe, nilai, status, atau berkas yang dihasilkan. Jangan menafsirkan keluaran di luar konteks data dan metode.

Catatan penggunaan: tuliskan alasan pemakaian perintah, bukan hanya sintaksnya. Apabila perintah bergantung pada paket tambahan, catat versi paket dan sediakan sel instalasi. Bila perintah dapat gagal, tangkap pengecualian yang spesifik atau periksa nilai balik. Untuk proses yang menghasilkan berkas, verifikasi keberadaan, ukuran, encoding, dan kemampuan pembacaan ulang.

H.006 enumerate()

Fungsi utama: Memberi indeks saat mengiterasi. Perintah ini sebaiknya ditempatkan pada tahap notebook yang jelas dan disertai pemeriksaan input. Contoh minimal: `for i, x in enumerate(data): pass`. Setelah menjalankan contoh, periksa tipe, nilai, status, atau berkas yang dihasilkan. Jangan menafsirkan keluaran di luar konteks data dan metode.

Catatan penggunaan: tuliskan alasan pemakaian perintah, bukan hanya sintaksnya. Apabila perintah bergantung pada paket tambahan, catat versi paket dan sediakan sel instalasi. Bila perintah dapat gagal, tangkap pengecualian yang spesifik atau periksa nilai balik. Untuk proses yang menghasilkan berkas, verifikasi keberadaan, ukuran, encoding, dan kemampuan pembacaan ulang.

H.007 zip()

Fungsi utama: Menggabungkan beberapa iterable sejajar. Perintah ini sebaiknya ditempatkan pada tahap notebook yang jelas dan disertai pemeriksaan input. Contoh minimal: `for x, y in zip(xs, ys): pass`. Setelah menjalankan contoh, periksa tipe, nilai, status, atau berkas yang dihasilkan. Jangan menafsirkan keluaran di luar konteks data dan metode.

Catatan penggunaan: tuliskan alasan pemakaian perintah, bukan hanya sintaksnya. Apabila perintah bergantung pada paket tambahan, catat versi paket dan sediakan sel instalasi. Bila perintah dapat gagal, tangkap pengecualian yang spesifik atau periksa nilai balik. Untuk proses yang menghasilkan berkas, verifikasi keberadaan, ukuran, encoding, dan kemampuan pembacaan ulang.

H.008 sum()

Fungsi utama: Menjumlahkan elemen. Perintah ini sebaiknya ditempatkan pada tahap notebook yang jelas dan disertai pemeriksaan input. Contoh minimal: `total = sum(data)`. Setelah menjalankan contoh, periksa tipe, nilai, status, atau berkas yang dihasilkan. Jangan menafsirkan keluaran di luar konteks data dan metode.

Catatan penggunaan: tuliskan alasan pemakaian perintah, bukan hanya sintaksnya. Apabila perintah bergantung pada paket tambahan, catat versi paket dan sediakan sel instalasi. Bila perintah dapat gagal, tangkap pengecualian yang spesifik atau periksa nilai balik. Untuk proses yang menghasilkan berkas, verifikasi keberadaan, ukuran, encoding, dan kemampuan pembacaan ulang.

H.009 min()

Fungsi utama: Mengambil nilai minimum. Perintah ini sebaiknya ditempatkan pada tahap notebook yang jelas dan disertai pemeriksaan input. Contoh minimal: `nilai_min = min(data)`. Setelah menjalankan contoh, periksa tipe, nilai, status, atau berkas yang dihasilkan. Jangan menafsirkan keluaran di luar konteks data dan metode.

Catatan penggunaan: tuliskan alasan pemakaian perintah, bukan hanya sintaksnya. Apabila perintah bergantung pada paket tambahan, catat versi paket dan sediakan sel instalasi. Bila perintah dapat gagal, tangkap pengecualian yang spesifik atau periksa nilai balik. Untuk proses yang menghasilkan berkas, verifikasi keberadaan, ukuran, encoding, dan kemampuan pembacaan ulang.

H.010 max()

Fungsi utama: Mengambil nilai maksimum. Perintah ini sebaiknya ditempatkan pada tahap notebook yang jelas dan disertai pemeriksaan input. Contoh minimal: `nilai_max =`

`max(data)`. Setelah menjalankan contoh, periksa tipe, nilai, status, atau berkas yang dihasilkan. Jangan menafsirkan keluaran di luar konteks data dan metode.

Catatan penggunaan: tuliskan alasan pemakaian perintah, bukan hanya sintaksnya. Apabila perintah bergantung pada paket tambahan, catat versi paket dan sediakan sel instalasi. Bila perintah dapat gagal, tangkap pengecualian yang spesifik atau periksa nilai balik. Untuk proses yang menghasilkan berkas, verifikasi keberadaan, ukuran, encoding, dan kemampuan pembacaan ulang.

H.011 sorted()

Fungsi utama: Menghasilkan urutan baru. Perintah ini sebaiknya ditempatkan pada tahap notebook yang jelas dan disertai pemeriksaan input. Contoh minimal: `urut = sorted(data)`. Setelah menjalankan contoh, periksa tipe, nilai, status, atau berkas yang dihasilkan. Jangan menafsirkan keluaran di luar konteks data dan metode.

Catatan penggunaan: tuliskan alasan pemakaian perintah, bukan hanya sintaksnya. Apabila perintah bergantung pada paket tambahan, catat versi paket dan sediakan sel instalasi. Bila perintah dapat gagal, tangkap pengecualian yang spesifik atau periksa nilai balik. Untuk proses yang menghasilkan berkas, verifikasi keberadaan, ukuran, encoding, dan kemampuan pembacaan ulang.

H.012 any()

Fungsi utama: Benar jika minimal satu kondisi benar. Perintah ini sebaiknya ditempatkan pada tahap notebook yang jelas dan disertai pemeriksaan input. Contoh minimal: `if any(mask): pass`. Setelah menjalankan contoh, periksa tipe, nilai, status, atau berkas yang dihasilkan. Jangan menafsirkan keluaran di luar konteks data dan metode.

Catatan penggunaan: tuliskan alasan pemakaian perintah, bukan hanya sintaksnya. Apabila perintah bergantung pada paket tambahan, catat versi paket dan sediakan sel instalasi. Bila perintah dapat gagal, tangkap pengecualian yang spesifik atau periksa nilai balik. Untuk proses yang menghasilkan berkas, verifikasi keberadaan, ukuran, encoding, dan kemampuan pembacaan ulang.

H.013 all()

Fungsi utama: Benar jika seluruh kondisi benar. Perintah ini sebaiknya ditempatkan pada tahap notebook yang jelas dan disertai pemeriksaan input. Contoh minimal: `assert all(valid)`. Setelah menjalankan contoh, periksa tipe, nilai, status, atau berkas yang dihasilkan. Jangan menafsirkan keluaran di luar konteks data dan metode.

Catatan penggunaan: tuliskan alasan pemakaian perintah, bukan hanya sintaksnya. Apabila perintah bergantung pada paket tambahan, catat versi paket dan sediakan sel instalasi. Bila perintah dapat gagal, tangkap pengecualian yang spesifik atau periksa nilai

balik. Untuk proses yang menghasilkan berkas, verifikasi keberadaan, ukuran, encoding, dan kemampuan pembacaan ulang.

H.014 round()

Fungsi utama: Membulatkan nilai untuk tampilan. Perintah ini sebaiknya ditempatkan pada tahap notebook yang jelas dan disertai pemeriksaan input. Contoh minimal: `x = round(nilai, 6)`. Setelah menjalankan contoh, periksa tipe, nilai, status, atau berkas yang dihasilkan. Jangan menafsirkan keluaran di luar konteks data dan metode.

Catatan penggunaan: tuliskan alasan pemakaian perintah, bukan hanya sintaksnya. Apabila perintah bergantung pada paket tambahan, catat versi paket dan sediakan sel instalasi. Bila perintah dapat gagal, tangkap pengecualian yang spesifik atau periksa nilai balik. Untuk proses yang menghasilkan berkas, verifikasi keberadaan, ukuran, encoding, dan kemampuan pembacaan ulang.

H.015 abs()

Fungsi utama: Mengambil nilai absolut. Perintah ini sebaiknya ditempatkan pada tahap notebook yang jelas dan disertai pemeriksaan input. Contoh minimal: `galat = abs(a-b)`. Setelah menjalankan contoh, periksa tipe, nilai, status, atau berkas yang dihasilkan. Jangan menafsirkan keluaran di luar konteks data dan metode.

Catatan penggunaan: tuliskan alasan pemakaian perintah, bukan hanya sintaksnya. Apabila perintah bergantung pada paket tambahan, catat versi paket dan sediakan sel instalasi. Bila perintah dapat gagal, tangkap pengecualian yang spesifik atau periksa nilai balik. Untuk proses yang menghasilkan berkas, verifikasi keberadaan, ukuran, encoding, dan kemampuan pembacaan ulang.

H.016 open()

Fungsi utama: Membuka berkas dengan mode dan encoding. Perintah ini sebaiknya ditempatkan pada tahap notebook yang jelas dan disertai pemeriksaan input. Contoh minimal: `with open("a.txt","r",encoding="utf-8") as f: pass`. Setelah menjalankan contoh, periksa tipe, nilai, status, atau berkas yang dihasilkan. Jangan menafsirkan keluaran di luar konteks data dan metode.

Catatan penggunaan: tuliskan alasan pemakaian perintah, bukan hanya sintaksnya. Apabila perintah bergantung pada paket tambahan, catat versi paket dan sediakan sel instalasi. Bila perintah dapat gagal, tangkap pengecualian yang spesifik atau periksa nilai balik. Untuk proses yang menghasilkan berkas, verifikasi keberadaan, ukuran, encoding, dan kemampuan pembacaan ulang.

H.017 with

Fungsi utama: Mengelola resource secara aman. Perintah ini sebaiknya ditempatkan pada tahap notebook yang jelas dan disertai pemeriksaan input. Contoh minimal: `with open("a.txt") as f: teks=f.read()`. Setelah menjalankan contoh, periksa tipe, nilai, status, atau berkas yang dihasilkan. Jangan menafsirkan keluaran di luar konteks data dan metode.

Catatan penggunaan: tuliskan alasan pemakaian perintah, bukan hanya sintaksnya. Apabila perintah bergantung pada paket tambahan, catat versi paket dan sediakan sel instalasi. Bila perintah dapat gagal, tangkap pengecualian yang spesifik atau periksa nilai balik. Untuk proses yang menghasilkan berkas, verifikasi keberadaan, ukuran, encoding, dan kemampuan pembacaan ulang.

H.018 try-except

Fungsi utama: Menangani pengecualian spesifik. Perintah ini sebaiknya ditempatkan pada tahap notebook yang jelas dan disertai pemeriksaan input. Contoh minimal: `try: proses()`
`except ValueError as e: print(e)`. Setelah menjalankan contoh, periksa tipe, nilai, status, atau berkas yang dihasilkan. Jangan menafsirkan keluaran di luar konteks data dan metode.

Catatan penggunaan: tuliskan alasan pemakaian perintah, bukan hanya sintaksnya. Apabila perintah bergantung pada paket tambahan, catat versi paket dan sediakan sel instalasi. Bila perintah dapat gagal, tangkap pengecualian yang spesifik atau periksa nilai balik. Untuk proses yang menghasilkan berkas, verifikasi keberadaan, ukuran, encoding, dan kemampuan pembacaan ulang.

H.019 raise

Fungsi utama: Menghentikan proses dengan kesalahan jelas. Perintah ini sebaiknya ditempatkan pada tahap notebook yang jelas dan disertai pemeriksaan input. Contoh minimal: `raise ValueError("input tidak valid")`. Setelah menjalankan contoh, periksa tipe, nilai, status, atau berkas yang dihasilkan. Jangan menafsirkan keluaran di luar konteks data dan metode.

Catatan penggunaan: tuliskan alasan pemakaian perintah, bukan hanya sintaksnya. Apabila perintah bergantung pada paket tambahan, catat versi paket dan sediakan sel instalasi. Bila perintah dapat gagal, tangkap pengecualian yang spesifik atau periksa nilai balik. Untuk proses yang menghasilkan berkas, verifikasi keberadaan, ukuran, encoding, dan kemampuan pembacaan ulang.

H.020 assert

Fungsi utama: Memeriksa asumsi pengembangan. Perintah ini sebaiknya ditempatkan pada tahap notebook yang jelas dan disertai pemeriksaan input. Contoh minimal: `assert xyz.shape[1] == 3`. Setelah menjalankan contoh, periksa tipe, nilai, status, atau berkas yang dihasilkan. Jangan menafsirkan keluaran di luar konteks data dan metode.

Catatan penggunaan: tuliskan alasan pemakaian perintah, bukan hanya sintaksnya. Apabila perintah bergantung pada paket tambahan, catat versi paket dan sediakan sel instalasi. Bila perintah dapat gagal, tangkap pengecualian yang spesifik atau periksa nilai balik. Untuk proses yang menghasilkan berkas, verifikasi keberadaan, ukuran, encoding, dan kemampuan pembacaan ulang.

H.Pathlib

H.021 Path()

Fungsi utama: Membuat objek jalur portabel. Perintah ini sebaiknya ditempatkan pada tahap notebook yang jelas dan disertai pemeriksaan input. Contoh minimal: `p = Path("output")`. Setelah menjalankan contoh, periksa tipe, nilai, status, atau berkas yang dihasilkan. Jangan menafsirkan keluaran di luar konteks data dan metode.

Catatan penggunaan: tuliskan alasan pemakaian perintah, bukan hanya sintaksnya. Apabila perintah bergantung pada paket tambahan, catat versi paket dan sediakan sel instalasi. Bila perintah dapat gagal, tangkap pengecualian yang spesifik atau periksa nilai balik. Untuk proses yang menghasilkan berkas, verifikasi keberadaan, ukuran, encoding, dan kemampuan pembacaan ulang.

H.022 Path.exists()

Fungsi utama: Memeriksa keberadaan jalur. Perintah ini sebaiknya ditempatkan pada tahap notebook yang jelas dan disertai pemeriksaan input. Contoh minimal: `if p.exists(): print(p)`. Setelah menjalankan contoh, periksa tipe, nilai, status, atau berkas yang dihasilkan. Jangan menafsirkan keluaran di luar konteks data dan metode.

Catatan penggunaan: tuliskan alasan pemakaian perintah, bukan hanya sintaksnya. Apabila perintah bergantung pada paket tambahan, catat versi paket dan sediakan sel instalasi. Bila perintah dapat gagal, tangkap pengecualian yang spesifik atau periksa nilai balik. Untuk proses yang menghasilkan berkas, verifikasi keberadaan, ukuran, encoding, dan kemampuan pembacaan ulang.

H.023 Path.mkdir()

Fungsi utama: Membuat direktori. Perintah ini sebaiknya ditempatkan pada tahap notebook yang jelas dan disertai pemeriksaan input. Contoh minimal:

`p.mkdir(parents=True, exist_ok=True)`. Setelah menjalankan contoh, periksa tipe, nilai, status, atau berkas yang dihasilkan. Jangan menafsirkan keluaran di luar konteks data dan metode.

Catatan penggunaan: tuliskan alasan pemakaian perintah, bukan hanya sintaksnya. Apabila perintah bergantung pada paket tambahan, catat versi paket dan sediakan sel instalasi. Bila perintah dapat gagal, tangkap pengecualian yang spesifik atau periksa nilai balik. Untuk proses yang menghasilkan berkas, verifikasi keberadaan, ukuran, encoding, dan kemampuan pembacaan ulang.

H.024 Path.read_text()

Fungsi utama: Membaca teks dengan encoding. Perintah ini sebaiknya ditempatkan pada tahap notebook yang jelas dan disertai pemeriksaan input. Contoh minimal: `teks = p.read_text(encoding="utf-8")`. Setelah menjalankan contoh, periksa tipe, nilai, status, atau berkas yang dihasilkan. Jangan menafsirkan keluaran di luar konteks data dan metode.

Catatan penggunaan: tuliskan alasan pemakaian perintah, bukan hanya sintaksnya. Apabila perintah bergantung pada paket tambahan, catat versi paket dan sediakan sel instalasi. Bila perintah dapat gagal, tangkap pengecualian yang spesifik atau periksa nilai balik. Untuk proses yang menghasilkan berkas, verifikasi keberadaan, ukuran, encoding, dan kemampuan pembacaan ulang.

H.025 Path.write_text()

Fungsi utama: Menulis teks dengan encoding. Perintah ini sebaiknya ditempatkan pada tahap notebook yang jelas dan disertai pemeriksaan input. Contoh minimal: `p.write_text(teks, encoding="utf-8")`. Setelah menjalankan contoh, periksa tipe, nilai, status, atau berkas yang dihasilkan. Jangan menafsirkan keluaran di luar konteks data dan metode.

Catatan penggunaan: tuliskan alasan pemakaian perintah, bukan hanya sintaksnya. Apabila perintah bergantung pada paket tambahan, catat versi paket dan sediakan sel instalasi. Bila perintah dapat gagal, tangkap pengecualian yang spesifik atau periksa nilai balik. Untuk proses yang menghasilkan berkas, verifikasi keberadaan, ukuran, encoding, dan kemampuan pembacaan ulang.

H.026 Path.glob()

Fungsi utama: Mencari berkas berdasarkan pola. Perintah ini sebaiknya ditempatkan pada tahap notebook yang jelas dan disertai pemeriksaan input. Contoh minimal: `files = list(Path(".").glob("*.csv"))`. Setelah menjalankan contoh, periksa tipe, nilai, status, atau berkas yang dihasilkan. Jangan menafsirkan keluaran di luar konteks data dan metode.

Catatan penggunaan: tuliskan alasan pemakaian perintah, bukan hanya sintaksnya. Apabila perintah bergantung pada paket tambahan, catat versi paket dan sediakan sel instalasi. Bila perintah dapat gagal, tangkap pengecualian yang spesifik atau periksa nilai balik. Untuk proses yang menghasilkan berkas, verifikasi keberadaan, ukuran, encoding, dan kemampuan pembacaan ulang.

H.027 Path.resolve()

Fungsi utama: Mendapatkan jalur absolut. Perintah ini sebaiknya ditempatkan pada tahap notebook yang jelas dan disertai pemeriksaan input. Contoh minimal: `print(p.resolve())`. Setelah menjalankan contoh, periksa tipe, nilai, status, atau berkas yang dihasilkan. Jangan menafsirkan keluaran di luar konteks data dan metode.

Catatan penggunaan: tuliskan alasan pemakaian perintah, bukan hanya sintaksnya. Apabila perintah bergantung pada paket tambahan, catat versi paket dan sediakan sel instalasi. Bila perintah dapat gagal, tangkap pengecualian yang spesifik atau periksa nilai balik. Untuk proses yang menghasilkan berkas, verifikasi keberadaan, ukuran, encoding, dan kemampuan pembacaan ulang.

H.028 Path.stat()

Fungsi utama: Membaca metadata ukuran/waktu. Perintah ini sebaiknya ditempatkan pada tahap notebook yang jelas dan disertai pemeriksaan input. Contoh minimal: `ukuran = p.stat().st_size`. Setelah menjalankan contoh, periksa tipe, nilai, status, atau berkas yang dihasilkan. Jangan menafsirkan keluaran di luar konteks data dan metode.

Catatan penggunaan: tuliskan alasan pemakaian perintah, bukan hanya sintaksnya. Apabila perintah bergantung pada paket tambahan, catat versi paket dan sediakan sel instalasi. Bila perintah dapat gagal, tangkap pengecualian yang spesifik atau periksa nilai balik. Untuk proses yang menghasilkan berkas, verifikasi keberadaan, ukuran, encoding, dan kemampuan pembacaan ulang.

H.JSON

H.029 json.dumps()

Fungsi utama: Mengubah objek menjadi teks JSON. Perintah ini sebaiknya ditempatkan pada tahap notebook yang jelas dan disertai pemeriksaan input. Contoh minimal: `teks = json.dumps(data, indent=2)`. Setelah menjalankan contoh, periksa tipe, nilai, status, atau berkas yang dihasilkan. Jangan menafsirkan keluaran di luar konteks data dan metode.

Catatan penggunaan: tuliskan alasan pemakaian perintah, bukan hanya sintaksnya. Apabila perintah bergantung pada paket tambahan, catat versi paket dan sediakan sel instalasi. Bila perintah dapat gagal, tangkap pengecualian yang spesifik atau periksa nilai

balik. Untuk proses yang menghasilkan berkas, verifikasi keberadaan, ukuran, encoding, dan kemampuan pembacaan ulang.

H.030 json.loads()

Fungsi utama: Mengubah teks JSON menjadi objek. Perintah ini sebaiknya ditempatkan pada tahap notebook yang jelas dan disertai pemeriksaan input. Contoh minimal: `data = json.loads(teks)`. Setelah menjalankan contoh, periksa tipe, nilai, status, atau berkas yang dihasilkan. Jangan menafsirkan keluaran di luar konteks data dan metode.

Catatan penggunaan: tuliskan alasan pemakaian perintah, bukan hanya sintaksnya. Apabila perintah bergantung pada paket tambahan, catat versi paket dan sediakan sel instalasi. Bila perintah dapat gagal, tangkap pengecualian yang spesifik atau periksa nilai balik. Untuk proses yang menghasilkan berkas, verifikasi keberadaan, ukuran, encoding, dan kemampuan pembacaan ulang.

H.031 json.dump()

Fungsi utama: Menulis JSON ke file object. Perintah ini sebaiknya ditempatkan pada tahap notebook yang jelas dan disertai pemeriksaan input. Contoh minimal: `json.dump(data, f, indent=2)`. Setelah menjalankan contoh, periksa tipe, nilai, status, atau berkas yang dihasilkan. Jangan menafsirkan keluaran di luar konteks data dan metode.

Catatan penggunaan: tuliskan alasan pemakaian perintah, bukan hanya sintaksnya. Apabila perintah bergantung pada paket tambahan, catat versi paket dan sediakan sel instalasi. Bila perintah dapat gagal, tangkap pengecualian yang spesifik atau periksa nilai balik. Untuk proses yang menghasilkan berkas, verifikasi keberadaan, ukuran, encoding, dan kemampuan pembacaan ulang.

H.032 json.load()

Fungsi utama: Membaca JSON dari file object. Perintah ini sebaiknya ditempatkan pada tahap notebook yang jelas dan disertai pemeriksaan input. Contoh minimal: `data = json.load(f)`. Setelah menjalankan contoh, periksa tipe, nilai, status, atau berkas yang dihasilkan. Jangan menafsirkan keluaran di luar konteks data dan metode.

Catatan penggunaan: tuliskan alasan pemakaian perintah, bukan hanya sintaksnya. Apabila perintah bergantung pada paket tambahan, catat versi paket dan sediakan sel instalasi. Bila perintah dapat gagal, tangkap pengecualian yang spesifik atau periksa nilai balik. Untuk proses yang menghasilkan berkas, verifikasi keberadaan, ukuran, encoding, dan kemampuan pembacaan ulang.

H.NumPy

H.033 np.array()

Fungsi utama: Membuat array numerik. Perintah ini sebaiknya ditempatkan pada tahap notebook yang jelas dan disertai pemeriksaan input. Contoh minimal: `a = np.array([1,2,3], dtype=float)`. Setelah menjalankan contoh, periksa tipe, nilai, status, atau berkas yang dihasilkan. Jangan menafsirkan keluaran di luar konteks data dan metode.

Catatan penggunaan: tuliskan alasan pemakaian perintah, bukan hanya sintaksnya. Apabila perintah bergantung pada paket tambahan, catat versi paket dan sediakan sel instalasi. Bila perintah dapat gagal, tangkap pengecualian yang spesifik atau periksa nilai balik. Untuk proses yang menghasilkan berkas, verifikasi keberadaan, ukuran, encoding, dan kemampuan pembacaan ulang.

H.034 np.asarray()

Fungsi utama: Mengubah input menjadi array tanpa salinan bila mungkin. Perintah ini sebaiknya ditempatkan pada tahap notebook yang jelas dan disertai pemeriksaan input. Contoh minimal: `a = np.asarray(data)`. Setelah menjalankan contoh, periksa tipe, nilai, status, atau berkas yang dihasilkan. Jangan menafsirkan keluaran di luar konteks data dan metode.

Catatan penggunaan: tuliskan alasan pemakaian perintah, bukan hanya sintaksnya. Apabila perintah bergantung pada paket tambahan, catat versi paket dan sediakan sel instalasi. Bila perintah dapat gagal, tangkap pengecualian yang spesifik atau periksa nilai balik. Untuk proses yang menghasilkan berkas, verifikasi keberadaan, ukuran, encoding, dan kemampuan pembacaan ulang.

H.035 np.arange()

Fungsi utama: Membuat rentang dengan interval. Perintah ini sebaiknya ditempatkan pada tahap notebook yang jelas dan disertai pemeriksaan input. Contoh minimal: `x = np.arange(0.4,2.01,0.1)`. Setelah menjalankan contoh, periksa tipe, nilai, status, atau berkas yang dihasilkan. Jangan menafsirkan keluaran di luar konteks data dan metode.

Catatan penggunaan: tuliskan alasan pemakaian perintah, bukan hanya sintaksnya. Apabila perintah bergantung pada paket tambahan, catat versi paket dan sediakan sel instalasi. Bila perintah dapat gagal, tangkap pengecualian yang spesifik atau periksa nilai balik. Untuk proses yang menghasilkan berkas, verifikasi keberadaan, ukuran, encoding, dan kemampuan pembacaan ulang.

H.036 np.linspace()

Fungsi utama: Membuat titik merata termasuk ujung. Perintah ini sebaiknya ditempatkan pada tahap notebook yang jelas dan disertai pemeriksaan input. Contoh minimal: `x = np.linspace(0,1,101)`. Setelah menjalankan contoh, periksa tipe, nilai, status, atau berkas yang dihasilkan. Jangan menafsirkan keluaran di luar konteks data dan metode.

Catatan penggunaan: tuliskan alasan pemakaian perintah, bukan hanya sintaksnya. Apabila perintah bergantung pada paket tambahan, catat versi paket dan sediakan sel instalasi. Bila perintah dapat gagal, tangkap pengecualian yang spesifik atau periksa nilai balik. Untuk proses yang menghasilkan berkas, verifikasi keberadaan, ukuran, encoding, dan kemampuan pembacaan ulang.

H.037 np.zeros()

Fungsi utama: Membuat array nol. Perintah ini sebaiknya ditempatkan pada tahap notebook yang jelas dan disertai pemeriksaan input. Contoh minimal: `a = np.zeros((3,3))`. Setelah menjalankan contoh, periksa tipe, nilai, status, atau berkas yang dihasilkan. Jangan menafsirkan keluaran di luar konteks data dan metode.

Catatan penggunaan: tuliskan alasan pemakaian perintah, bukan hanya sintaksnya. Apabila perintah bergantung pada paket tambahan, catat versi paket dan sediakan sel instalasi. Bila perintah dapat gagal, tangkap pengecualian yang spesifik atau periksa nilai balik. Untuk proses yang menghasilkan berkas, verifikasi keberadaan, ukuran, encoding, dan kemampuan pembacaan ulang.

H.038 np.ones()

Fungsi utama: Membuat array satu. Perintah ini sebaiknya ditempatkan pada tahap notebook yang jelas dan disertai pemeriksaan input. Contoh minimal: `a = np.ones(5)`. Setelah menjalankan contoh, periksa tipe, nilai, status, atau berkas yang dihasilkan. Jangan menafsirkan keluaran di luar konteks data dan metode.

Catatan penggunaan: tuliskan alasan pemakaian perintah, bukan hanya sintaksnya. Apabila perintah bergantung pada paket tambahan, catat versi paket dan sediakan sel instalasi. Bila perintah dapat gagal, tangkap pengecualian yang spesifik atau periksa nilai balik. Untuk proses yang menghasilkan berkas, verifikasi keberadaan, ukuran, encoding, dan kemampuan pembacaan ulang.

H.039 np.eye()

Fungsi utama: Membuat matriks identitas. Perintah ini sebaiknya ditempatkan pada tahap notebook yang jelas dan disertai pemeriksaan input. Contoh minimal: `I = np.eye(3)`. Setelah menjalankan contoh, periksa tipe, nilai, status, atau berkas yang dihasilkan. Jangan menafsirkan keluaran di luar konteks data dan metode.

Catatan penggunaan: tuliskan alasan pemakaian perintah, bukan hanya sintaksnya. Apabila perintah bergantung pada paket tambahan, catat versi paket dan sediakan sel instalasi. Bila perintah dapat gagal, tangkap pengecualian yang spesifik atau periksa nilai balik. Untuk proses yang menghasilkan berkas, verifikasi keberadaan, ukuran, encoding, dan kemampuan pembacaan ulang.

H.040 np.isfinite()

Fungsi utama: Mendeteksi nilai hingga. Perintah ini sebaiknya ditempatkan pada tahap notebook yang jelas dan disertai pemeriksaan input. Contoh minimal: `mask = np.isfinite(a)`. Setelah menjalankan contoh, periksa tipe, nilai, status, atau berkas yang dihasilkan. Jangan menafsirkan keluaran di luar konteks data dan metode.

Catatan penggunaan: tuliskan alasan pemakaian perintah, bukan hanya sintaksnya. Apabila perintah bergantung pada paket tambahan, catat versi paket dan sediakan sel instalasi. Bila perintah dapat gagal, tangkap pengecualian yang spesifik atau periksa nilai balik. Untuk proses yang menghasilkan berkas, verifikasi keberadaan, ukuran, encoding, dan kemampuan pembacaan ulang.

H.041 np.isnan()

Fungsi utama: Mendeteksi NaN. Perintah ini sebaiknya ditempatkan pada tahap notebook yang jelas dan disertai pemeriksaan input. Contoh minimal: `mask = np.isnan(a)`. Setelah menjalankan contoh, periksa tipe, nilai, status, atau berkas yang dihasilkan. Jangan menafsirkan keluaran di luar konteks data dan metode.

Catatan penggunaan: tuliskan alasan pemakaian perintah, bukan hanya sintaksnya. Apabila perintah bergantung pada paket tambahan, catat versi paket dan sediakan sel instalasi. Bila perintah dapat gagal, tangkap pengecualian yang spesifik atau periksa nilai balik. Untuk proses yang menghasilkan berkas, verifikasi keberadaan, ukuran, encoding, dan kemampuan pembacaan ulang.

H.042 np.allclose()

Fungsi utama: Membandingkan array dengan toleransi. Perintah ini sebaiknya ditempatkan pada tahap notebook yang jelas dan disertai pemeriksaan input. Contoh minimal: `assert np.allclose(a,b)`. Setelah menjalankan contoh, periksa tipe, nilai, status, atau berkas yang dihasilkan. Jangan menafsirkan keluaran di luar konteks data dan metode.

Catatan penggunaan: tuliskan alasan pemakaian perintah, bukan hanya sintaksnya. Apabila perintah bergantung pada paket tambahan, catat versi paket dan sediakan sel instalasi. Bila perintah dapat gagal, tangkap pengecualian yang spesifik atau periksa nilai balik. Untuk proses yang menghasilkan berkas, verifikasi keberadaan, ukuran, encoding, dan kemampuan pembacaan ulang.

H.043 np.dot()

Fungsi utama: Menghitung dot product. Perintah ini sebaiknya ditempatkan pada tahap notebook yang jelas dan disertai pemeriksaan input. Contoh minimal: `d = np.dot(v1,v2)`. Setelah menjalankan contoh, periksa tipe, nilai, status, atau berkas yang dihasilkan. Jangan menafsirkan keluaran di luar konteks data dan metode.

Catatan penggunaan: tuliskan alasan pemakaian perintah, bukan hanya sintaksnya. Apabila perintah bergantung pada paket tambahan, catat versi paket dan sediakan sel instalasi. Bila perintah dapat gagal, tangkap pengecualian yang spesifik atau periksa nilai balik. Untuk proses yang menghasilkan berkas, verifikasi keberadaan, ukuran, encoding, dan kemampuan pembacaan ulang.

H.044 np.cross()

Fungsi utama: Menghitung cross product. Perintah ini sebaiknya ditempatkan pada tahap notebook yang jelas dan disertai pemeriksaan input. Contoh minimal: `c = np.cross(v1,v2)`. Setelah menjalankan contoh, periksa tipe, nilai, status, atau berkas yang dihasilkan. Jangan menafsirkan keluaran di luar konteks data dan metode.

Catatan penggunaan: tuliskan alasan pemakaian perintah, bukan hanya sintaksnya. Apabila perintah bergantung pada paket tambahan, catat versi paket dan sediakan sel instalasi. Bila perintah dapat gagal, tangkap pengecualian yang spesifik atau periksa nilai balik. Untuk proses yang menghasilkan berkas, verifikasi keberadaan, ukuran, encoding, dan kemampuan pembacaan ulang.

H.045 np.linalg.norm()

Fungsi utama: Menghitung norma vektor/matriks. Perintah ini sebaiknya ditempatkan pada tahap notebook yang jelas dan disertai pemeriksaan input. Contoh minimal: `r = np.linalg.norm(v)`. Setelah menjalankan contoh, periksa tipe, nilai, status, atau berkas yang dihasilkan. Jangan menafsirkan keluaran di luar konteks data dan metode.

Catatan penggunaan: tuliskan alasan pemakaian perintah, bukan hanya sintaksnya. Apabila perintah bergantung pada paket tambahan, catat versi paket dan sediakan sel instalasi. Bila perintah dapat gagal, tangkap pengecualian yang spesifik atau periksa nilai balik. Untuk proses yang menghasilkan berkas, verifikasi keberadaan, ukuran, encoding, dan kemampuan pembacaan ulang.

H.046 np.linalg.eigh()

Fungsi utama: Eigen untuk matriks simetris/Hermitian. Perintah ini sebaiknya ditempatkan pada tahap notebook yang jelas dan disertai pemeriksaan input. Contoh minimal: `w,v = np.linalg.eigh(A)`. Setelah menjalankan contoh, periksa tipe, nilai, status,

atau berkas yang dihasilkan. Jangan menafsirkan keluaran di luar konteks data dan metode.

Catatan penggunaan: tuliskan alasan pemakaian perintah, bukan hanya sintaksnya. Apabila perintah bergantung pada paket tambahan, catat versi paket dan sediakan sel instalasi. Bila perintah dapat gagal, tangkap pengecualian yang spesifik atau periksa nilai balik. Untuk proses yang menghasilkan berkas, verifikasi keberadaan, ukuran, encoding, dan kemampuan pembacaan ulang.

H.047 np.linalg.solve()

Fungsi utama: Menyelesaikan $Ax=b$. Perintah ini sebaiknya ditempatkan pada tahap notebook yang jelas dan disertai pemeriksaan input. Contoh minimal: `x = np.linalg.solve(A,b)`. Setelah menjalankan contoh, periksa tipe, nilai, status, atau berkas yang dihasilkan. Jangan menafsirkan keluaran di luar konteks data dan metode.

Catatan penggunaan: tuliskan alasan pemakaian perintah, bukan hanya sintaksnya. Apabila perintah bergantung pada paket tambahan, catat versi paket dan sediakan sel instalasi. Bila perintah dapat gagal, tangkap pengecualian yang spesifik atau periksa nilai balik. Untuk proses yang menghasilkan berkas, verifikasi keberadaan, ukuran, encoding, dan kemampuan pembacaan ulang.

H.048 np.mean()

Fungsi utama: Menghitung rata-rata. Perintah ini sebaiknya ditempatkan pada tahap notebook yang jelas dan disertai pemeriksaan input. Contoh minimal: `m = np.mean(a)`. Setelah menjalankan contoh, periksa tipe, nilai, status, atau berkas yang dihasilkan. Jangan menafsirkan keluaran di luar konteks data dan metode.

Catatan penggunaan: tuliskan alasan pemakaian perintah, bukan hanya sintaksnya. Apabila perintah bergantung pada paket tambahan, catat versi paket dan sediakan sel instalasi. Bila perintah dapat gagal, tangkap pengecualian yang spesifik atau periksa nilai balik. Untuk proses yang menghasilkan berkas, verifikasi keberadaan, ukuran, encoding, dan kemampuan pembacaan ulang.

H.049 np.median()

Fungsi utama: Menghitung median. Perintah ini sebaiknya ditempatkan pada tahap notebook yang jelas dan disertai pemeriksaan input. Contoh minimal: `m = np.median(a)`. Setelah menjalankan contoh, periksa tipe, nilai, status, atau berkas yang dihasilkan. Jangan menafsirkan keluaran di luar konteks data dan metode.

Catatan penggunaan: tuliskan alasan pemakaian perintah, bukan hanya sintaksnya. Apabila perintah bergantung pada paket tambahan, catat versi paket dan sediakan sel instalasi. Bila perintah dapat gagal, tangkap pengecualian yang spesifik atau periksa nilai

balik. Untuk proses yang menghasilkan berkas, verifikasi keberadaan, ukuran, encoding, dan kemampuan pembacaan ulang.

H.050 np.std()

Fungsi utama: Menghitung simpangan baku. Perintah ini sebaiknya ditempatkan pada tahap notebook yang jelas dan disertai pemeriksaan input. Contoh minimal: `s = np.std(a, ddof=1)`. Setelah menjalankan contoh, periksa tipe, nilai, status, atau berkas yang dihasilkan. Jangan menafsirkan keluaran di luar konteks data dan metode.

Catatan penggunaan: tuliskan alasan pemakaian perintah, bukan hanya sintaksnya. Apabila perintah bergantung pada paket tambahan, catat versi paket dan sediakan sel instalasi. Bila perintah dapat gagal, tangkap pengecualian yang spesifik atau periksa nilai balik. Untuk proses yang menghasilkan berkas, verifikasi keberadaan, ukuran, encoding, dan kemampuan pembacaan ulang.

H.051 np.argmin()

Fungsi utama: Indeks nilai minimum. Perintah ini sebaiknya ditempatkan pada tahap notebook yang jelas dan disertai pemeriksaan input. Contoh minimal: `i = np.argmin(a)`. Setelah menjalankan contoh, periksa tipe, nilai, status, atau berkas yang dihasilkan. Jangan menafsirkan keluaran di luar konteks data dan metode.

Catatan penggunaan: tuliskan alasan pemakaian perintah, bukan hanya sintaksnya. Apabila perintah bergantung pada paket tambahan, catat versi paket dan sediakan sel instalasi. Bila perintah dapat gagal, tangkap pengecualian yang spesifik atau periksa nilai balik. Untuk proses yang menghasilkan berkas, verifikasi keberadaan, ukuran, encoding, dan kemampuan pembacaan ulang.

H.052 np.argmax()

Fungsi utama: Indeks nilai maksimum. Perintah ini sebaiknya ditempatkan pada tahap notebook yang jelas dan disertai pemeriksaan input. Contoh minimal: `i = np.argmax(a)`. Setelah menjalankan contoh, periksa tipe, nilai, status, atau berkas yang dihasilkan. Jangan menafsirkan keluaran di luar konteks data dan metode.

Catatan penggunaan: tuliskan alasan pemakaian perintah, bukan hanya sintaksnya. Apabila perintah bergantung pada paket tambahan, catat versi paket dan sediakan sel instalasi. Bila perintah dapat gagal, tangkap pengecualian yang spesifik atau periksa nilai balik. Untuk proses yang menghasilkan berkas, verifikasi keberadaan, ukuran, encoding, dan kemampuan pembacaan ulang.

H.053 np.where()

Fungsi utama: Memilih indeks berdasarkan kondisi. Perintah ini sebaiknya ditempatkan pada tahap notebook yang jelas dan disertai pemeriksaan input. Contoh minimal: `idx =`

`np.where(a>0)[0]`. Setelah menjalankan contoh, periksa tipe, nilai, status, atau berkas yang dihasilkan. Jangan menafsirkan keluaran di luar konteks data dan metode.

Catatan penggunaan: tuliskan alasan pemakaian perintah, bukan hanya sintaksnya. Apabila perintah bergantung pada paket tambahan, catat versi paket dan sediakan sel instalasi. Bila perintah dapat gagal, tangkap pengecualian yang spesifik atau periksa nilai balik. Untuk proses yang menghasilkan berkas, verifikasi keberadaan, ukuran, encoding, dan kemampuan pembacaan ulang.

H.054 np.clip()

Fungsi utama: Membatasi nilai pada rentang. Perintah ini sebaiknya ditempatkan pada tahap notebook yang jelas dan disertai pemeriksaan input. Contoh minimal: `x = np.clip(x,-1,1)`. Setelah menjalankan contoh, periksa tipe, nilai, status, atau berkas yang dihasilkan. Jangan menafsirkan keluaran di luar konteks data dan metode.

Catatan penggunaan: tuliskan alasan pemakaian perintah, bukan hanya sintaksnya. Apabila perintah bergantung pada paket tambahan, catat versi paket dan sediakan sel instalasi. Bila perintah dapat gagal, tangkap pengecualian yang spesifik atau periksa nilai balik. Untuk proses yang menghasilkan berkas, verifikasi keberadaan, ukuran, encoding, dan kemampuan pembacaan ulang.

H.055 np.gradient()

Fungsi utama: Mengestimasi turunan numerik. Perintah ini sebaiknya ditempatkan pada tahap notebook yang jelas dan disertai pemeriksaan input. Contoh minimal: `g = np.gradient(y,x)`. Setelah menjalankan contoh, periksa tipe, nilai, status, atau berkas yang dihasilkan. Jangan menafsirkan keluaran di luar konteks data dan metode.

Catatan penggunaan: tuliskan alasan pemakaian perintah, bukan hanya sintaksnya. Apabila perintah bergantung pada paket tambahan, catat versi paket dan sediakan sel instalasi. Bila perintah dapat gagal, tangkap pengecualian yang spesifik atau periksa nilai balik. Untuk proses yang menghasilkan berkas, verifikasi keberadaan, ukuran, encoding, dan kemampuan pembacaan ulang.

H.056 np.trapezoid()

Fungsi utama: Menghitung integral trapezoid. Perintah ini sebaiknya ditempatkan pada tahap notebook yang jelas dan disertai pemeriksaan input. Contoh minimal: `I = np.trapezoid(y,x)`. Setelah menjalankan contoh, periksa tipe, nilai, status, atau berkas yang dihasilkan. Jangan menafsirkan keluaran di luar konteks data dan metode.

Catatan penggunaan: tuliskan alasan pemakaian perintah, bukan hanya sintaksnya. Apabila perintah bergantung pada paket tambahan, catat versi paket dan sediakan sel instalasi. Bila perintah dapat gagal, tangkap pengecualian yang spesifik atau periksa nilai

balik. Untuk proses yang menghasilkan berkas, verifikasi keberadaan, ukuran, encoding, dan kemampuan pembacaan ulang.

H.057 np.interp()

Fungsi utama: Interpolasi linear satu dimensi. Perintah ini sebaiknya ditempatkan pada tahap notebook yang jelas dan disertai pemeriksaan input. Contoh minimal: `y2 = np.interp(x2,x,y)`. Setelah menjalankan contoh, periksa tipe, nilai, status, atau berkas yang dihasilkan. Jangan menafsirkan keluaran di luar konteks data dan metode.

Catatan penggunaan: tuliskan alasan pemakaian perintah, bukan hanya sintaksnya. Apabila perintah bergantung pada paket tambahan, catat versi paket dan sediakan sel instalasi. Bila perintah dapat gagal, tangkap pengecualian yang spesifik atau periksa nilai balik. Untuk proses yang menghasilkan berkas, verifikasi keberadaan, ukuran, encoding, dan kemampuan pembacaan ulang.

H.058 np.save()

Fungsi utama: Menyimpan array biner NPY. Perintah ini sebaiknya ditempatkan pada tahap notebook yang jelas dan disertai pemeriksaan input. Contoh minimal: `np.save("a.npy",a)`. Setelah menjalankan contoh, periksa tipe, nilai, status, atau berkas yang dihasilkan. Jangan menafsirkan keluaran di luar konteks data dan metode.

Catatan penggunaan: tuliskan alasan pemakaian perintah, bukan hanya sintaksnya. Apabila perintah bergantung pada paket tambahan, catat versi paket dan sediakan sel instalasi. Bila perintah dapat gagal, tangkap pengecualian yang spesifik atau periksa nilai balik. Untuk proses yang menghasilkan berkas, verifikasi keberadaan, ukuran, encoding, dan kemampuan pembacaan ulang.

H.059 np.load()

Fungsi utama: Memuat NPY. Perintah ini sebaiknya ditempatkan pada tahap notebook yang jelas dan disertai pemeriksaan input. Contoh minimal: `a = np.load("a.npy")`. Setelah menjalankan contoh, periksa tipe, nilai, status, atau berkas yang dihasilkan. Jangan menafsirkan keluaran di luar konteks data dan metode.

Catatan penggunaan: tuliskan alasan pemakaian perintah, bukan hanya sintaksnya. Apabila perintah bergantung pada paket tambahan, catat versi paket dan sediakan sel instalasi. Bila perintah dapat gagal, tangkap pengecualian yang spesifik atau periksa nilai balik. Untuk proses yang menghasilkan berkas, verifikasi keberadaan, ukuran, encoding, dan kemampuan pembacaan ulang.

H.060 np.savetxt()

Fungsi utama: Menyimpan array teks. Perintah ini sebaiknya ditempatkan pada tahap notebook yang jelas dan disertai pemeriksaan input. Contoh minimal:

`np.savetxt("a.csv",a,delimiter=",")`. Setelah menjalankan contoh, periksa tipe, nilai, status, atau berkas yang dihasilkan. Jangan menafsirkan keluaran di luar konteks data dan metode.

Catatan penggunaan: tuliskan alasan pemakaian perintah, bukan hanya sintaksnya. Apabila perintah bergantung pada paket tambahan, catat versi paket dan sediakan sel instalasi. Bila perintah dapat gagal, tangkap pengecualian yang spesifik atau periksa nilai balik. Untuk proses yang menghasilkan berkas, verifikasi keberadaan, ukuran, encoding, dan kemampuan pembacaan ulang.

H.061 np.loadtxt()

Fungsi utama: Memuat array teks. Perintah ini sebaiknya ditempatkan pada tahap notebook yang jelas dan disertai pemeriksaan input. Contoh minimal: `a=np.loadtxt("a.csv",delimiter=",")`. Setelah menjalankan contoh, periksa tipe, nilai, status, atau berkas yang dihasilkan. Jangan menafsirkan keluaran di luar konteks data dan metode.

Catatan penggunaan: tuliskan alasan pemakaian perintah, bukan hanya sintaksnya. Apabila perintah bergantung pada paket tambahan, catat versi paket dan sediakan sel instalasi. Bila perintah dapat gagal, tangkap pengecualian yang spesifik atau periksa nilai balik. Untuk proses yang menghasilkan berkas, verifikasi keberadaan, ukuran, encoding, dan kemampuan pembacaan ulang.

H.Pandas

H.062 pd.DataFrame()

Fungsi utama: Membuat tabel. Perintah ini sebaiknya ditempatkan pada tahap notebook yang jelas dan disertai pemeriksaan input. Contoh minimal: `df=pd.DataFrame(data)`. Setelah menjalankan contoh, periksa tipe, nilai, status, atau berkas yang dihasilkan. Jangan menafsirkan keluaran di luar konteks data dan metode.

Catatan penggunaan: tuliskan alasan pemakaian perintah, bukan hanya sintaksnya. Apabila perintah bergantung pada paket tambahan, catat versi paket dan sediakan sel instalasi. Bila perintah dapat gagal, tangkap pengecualian yang spesifik atau periksa nilai balik. Untuk proses yang menghasilkan berkas, verifikasi keberadaan, ukuran, encoding, dan kemampuan pembacaan ulang.

H.063 pd.Series()

Fungsi utama: Membuat vektor berlabel. Perintah ini sebaiknya ditempatkan pada tahap notebook yang jelas dan disertai pemeriksaan input. Contoh minimal: `s=pd.Series(data)`. Setelah menjalankan contoh, periksa tipe, nilai, status, atau berkas yang dihasilkan. Jangan menafsirkan keluaran di luar konteks data dan metode.

Catatan penggunaan: tuliskan alasan pemakaian perintah, bukan hanya sintaksnya. Apabila perintah bergantung pada paket tambahan, catat versi paket dan sediakan sel instalasi. Bila perintah dapat gagal, tangkap pengecualian yang spesifik atau periksa nilai balik. Untuk proses yang menghasilkan berkas, verifikasi keberadaan, ukuran, encoding, dan kemampuan pembacaan ulang.

H.064 pd.read_csv()

Fungsi utama: Membaca CSV. Perintah ini sebaiknya ditempatkan pada tahap notebook yang jelas dan disertai pemeriksaan input. Contoh minimal: `df=pd.read_csv("data.csv")`. Setelah menjalankan contoh, periksa tipe, nilai, status, atau berkas yang dihasilkan. Jangan menafsirkan keluaran di luar konteks data dan metode.

Catatan penggunaan: tuliskan alasan pemakaian perintah, bukan hanya sintaksnya. Apabila perintah bergantung pada paket tambahan, catat versi paket dan sediakan sel instalasi. Bila perintah dapat gagal, tangkap pengecualian yang spesifik atau periksa nilai balik. Untuk proses yang menghasilkan berkas, verifikasi keberadaan, ukuran, encoding, dan kemampuan pembacaan ulang.

H.065 DataFrame.to_csv()

Fungsi utama: Menyimpan CSV. Perintah ini sebaiknya ditempatkan pada tahap notebook yang jelas dan disertai pemeriksaan input. Contoh minimal: `df.to_csv("hasil.csv",index=False)`. Setelah menjalankan contoh, periksa tipe, nilai, status, atau berkas yang dihasilkan. Jangan menafsirkan keluaran di luar konteks data dan metode.

Catatan penggunaan: tuliskan alasan pemakaian perintah, bukan hanya sintaksnya. Apabila perintah bergantung pada paket tambahan, catat versi paket dan sediakan sel instalasi. Bila perintah dapat gagal, tangkap pengecualian yang spesifik atau periksa nilai balik. Untuk proses yang menghasilkan berkas, verifikasi keberadaan, ukuran, encoding, dan kemampuan pembacaan ulang.

H.066 DataFrame.to_excel()

Fungsi utama: Menyimpan Excel. Perintah ini sebaiknya ditempatkan pada tahap notebook yang jelas dan disertai pemeriksaan input. Contoh minimal: `df.to_excel("hasil.xlsx",index=False)`. Setelah menjalankan contoh, periksa tipe, nilai, status, atau berkas yang dihasilkan. Jangan menafsirkan keluaran di luar konteks data dan metode.

Catatan penggunaan: tuliskan alasan pemakaian perintah, bukan hanya sintaksnya. Apabila perintah bergantung pada paket tambahan, catat versi paket dan sediakan sel instalasi. Bila perintah dapat gagal, tangkap pengecualian yang spesifik atau periksa nilai

balik. Untuk proses yang menghasilkan berkas, verifikasi keberadaan, ukuran, encoding, dan kemampuan pembacaan ulang.

H.067 DataFrame.head()

Fungsi utama: Menampilkan baris awal. Perintah ini sebaiknya ditempatkan pada tahap notebook yang jelas dan disertai pemeriksaan input. Contoh minimal: `display(df.head())`. Setelah menjalankan contoh, periksa tipe, nilai, status, atau berkas yang dihasilkan. Jangan menafsirkan keluaran di luar konteks data dan metode.

Catatan penggunaan: tuliskan alasan pemakaian perintah, bukan hanya sintaksnya. Apabila perintah bergantung pada paket tambahan, catat versi paket dan sediakan sel instalasi. Bila perintah dapat gagal, tangkap pengecualian yang spesifik atau periksa nilai balik. Untuk proses yang menghasilkan berkas, verifikasi keberadaan, ukuran, encoding, dan kemampuan pembacaan ulang.

H.068 DataFrame.info()

Fungsi utama: Ringkasan kolom dan dtype. Perintah ini sebaiknya ditempatkan pada tahap notebook yang jelas dan disertai pemeriksaan input. Contoh minimal: `df.info()`. Setelah menjalankan contoh, periksa tipe, nilai, status, atau berkas yang dihasilkan. Jangan menafsirkan keluaran di luar konteks data dan metode.

Catatan penggunaan: tuliskan alasan pemakaian perintah, bukan hanya sintaksnya. Apabila perintah bergantung pada paket tambahan, catat versi paket dan sediakan sel instalasi. Bila perintah dapat gagal, tangkap pengecualian yang spesifik atau periksa nilai balik. Untuk proses yang menghasilkan berkas, verifikasi keberadaan, ukuran, encoding, dan kemampuan pembacaan ulang.

H.069 DataFrame.describe()

Fungsi utama: Statistik deskriptif. Perintah ini sebaiknya ditempatkan pada tahap notebook yang jelas dan disertai pemeriksaan input. Contoh minimal: `print(df.describe())`. Setelah menjalankan contoh, periksa tipe, nilai, status, atau berkas yang dihasilkan. Jangan menafsirkan keluaran di luar konteks data dan metode.

Catatan penggunaan: tuliskan alasan pemakaian perintah, bukan hanya sintaksnya. Apabila perintah bergantung pada paket tambahan, catat versi paket dan sediakan sel instalasi. Bila perintah dapat gagal, tangkap pengecualian yang spesifik atau periksa nilai balik. Untuk proses yang menghasilkan berkas, verifikasi keberadaan, ukuran, encoding, dan kemampuan pembacaan ulang.

H.070 DataFrame.isna()

Fungsi utama: Mendeteksi nilai hilang. Perintah ini sebaiknya ditempatkan pada tahap notebook yang jelas dan disertai pemeriksaan input. Contoh minimal:

`print(df.isna().sum())`). Setelah menjalankan contoh, periksa tipe, nilai, status, atau berkas yang dihasilkan. Jangan menafsirkan keluaran di luar konteks data dan metode.

Catatan penggunaan: tuliskan alasan pemakaian perintah, bukan hanya sintaksnya. Apabila perintah bergantung pada paket tambahan, catat versi paket dan sediakan sel instalasi. Bila perintah dapat gagal, tangkap pengecualian yang spesifik atau periksa nilai balik. Untuk proses yang menghasilkan berkas, verifikasi keberadaan, ukuran, encoding, dan kemampuan pembacaan ulang.

H.071 DataFrame.dropna()

Fungsi utama: Menghapus baris/kolom NaN. Perintah ini sebaiknya ditempatkan pada tahap notebook yang jelas dan disertai pemeriksaan input. Contoh minimal: `df2=df.dropna()`. Setelah menjalankan contoh, periksa tipe, nilai, status, atau berkas yang dihasilkan. Jangan menafsirkan keluaran di luar konteks data dan metode.

Catatan penggunaan: tuliskan alasan pemakaian perintah, bukan hanya sintaksnya. Apabila perintah bergantung pada paket tambahan, catat versi paket dan sediakan sel instalasi. Bila perintah dapat gagal, tangkap pengecualian yang spesifik atau periksa nilai balik. Untuk proses yang menghasilkan berkas, verifikasi keberadaan, ukuran, encoding, dan kemampuan pembacaan ulang.

H.072 DataFrame.fillna()

Fungsi utama: Mengisi nilai hilang. Perintah ini sebaiknya ditempatkan pada tahap notebook yang jelas dan disertai pemeriksaan input. Contoh minimal: `df["x"]=df["x"].fillna(0)`. Setelah menjalankan contoh, periksa tipe, nilai, status, atau berkas yang dihasilkan. Jangan menafsirkan keluaran di luar konteks data dan metode.

Catatan penggunaan: tuliskan alasan pemakaian perintah, bukan hanya sintaksnya. Apabila perintah bergantung pada paket tambahan, catat versi paket dan sediakan sel instalasi. Bila perintah dapat gagal, tangkap pengecualian yang spesifik atau periksa nilai balik. Untuk proses yang menghasilkan berkas, verifikasi keberadaan, ukuran, encoding, dan kemampuan pembacaan ulang.

H.073 DataFrame.duplicated()

Fungsi utama: Mendeteksi duplikat. Perintah ini sebaiknya ditempatkan pada tahap notebook yang jelas dan disertai pemeriksaan input. Contoh minimal: `mask=df.duplicated()`. Setelah menjalankan contoh, periksa tipe, nilai, status, atau berkas yang dihasilkan. Jangan menafsirkan keluaran di luar konteks data dan metode.

Catatan penggunaan: tuliskan alasan pemakaian perintah, bukan hanya sintaksnya. Apabila perintah bergantung pada paket tambahan, catat versi paket dan sediakan sel instalasi. Bila perintah dapat gagal, tangkap pengecualian yang spesifik atau periksa nilai

balik. Untuk proses yang menghasilkan berkas, verifikasi keberadaan, ukuran, encoding, dan kemampuan pembacaan ulang.

H.074 DataFrame.drop_duplicates()

Fungsi utama: Menghapus duplikat. Perintah ini sebaiknya ditempatkan pada tahap notebook yang jelas dan disertai pemeriksaan input. Contoh minimal: `df=df.drop_duplicates()`. Setelah menjalankan contoh, periksa tipe, nilai, status, atau berkas yang dihasilkan. Jangan menafsirkan keluaran di luar konteks data dan metode.

Catatan penggunaan: tuliskan alasan pemakaian perintah, bukan hanya sintaksnya. Apabila perintah bergantung pada paket tambahan, catat versi paket dan sediakan sel instalasi. Bila perintah dapat gagal, tangkap pengecualian yang spesifik atau periksa nilai balik. Untuk proses yang menghasilkan berkas, verifikasi keberadaan, ukuran, encoding, dan kemampuan pembacaan ulang.

H.075 DataFrame.loc[]

Fungsi utama: Seleksi berlabel. Perintah ini sebaiknya ditempatkan pada tahap notebook yang jelas dan disertai pemeriksaan input. Contoh minimal: `sub=df.loc[df["ok"], ["x","y"]]`. Setelah menjalankan contoh, periksa tipe, nilai, status, atau berkas yang dihasilkan. Jangan menafsirkan keluaran di luar konteks data dan metode.

Catatan penggunaan: tuliskan alasan pemakaian perintah, bukan hanya sintaksnya. Apabila perintah bergantung pada paket tambahan, catat versi paket dan sediakan sel instalasi. Bila perintah dapat gagal, tangkap pengecualian yang spesifik atau periksa nilai balik. Untuk proses yang menghasilkan berkas, verifikasi keberadaan, ukuran, encoding, dan kemampuan pembacaan ulang.

H.076 DataFrame.iloc[]

Fungsi utama: Seleksi berdasarkan posisi. Perintah ini sebaiknya ditempatkan pada tahap notebook yang jelas dan disertai pemeriksaan input. Contoh minimal: `baris=df.iloc[:5]`. Setelah menjalankan contoh, periksa tipe, nilai, status, atau berkas yang dihasilkan. Jangan menafsirkan keluaran di luar konteks data dan metode.

Catatan penggunaan: tuliskan alasan pemakaian perintah, bukan hanya sintaksnya. Apabila perintah bergantung pada paket tambahan, catat versi paket dan sediakan sel instalasi. Bila perintah dapat gagal, tangkap pengecualian yang spesifik atau periksa nilai balik. Untuk proses yang menghasilkan berkas, verifikasi keberadaan, ukuran, encoding, dan kemampuan pembacaan ulang.

H.077 DataFrame.query()

Fungsi utama: Filter dengan ekspresi. Perintah ini sebaiknya ditempatkan pada tahap notebook yang jelas dan disertai pemeriksaan input. Contoh minimal:

`sub=df.query("energi < -1")`. Setelah menjalankan contoh, periksa tipe, nilai, status, atau berkas yang dihasilkan. Jangan menafsirkan keluaran di luar konteks data dan metode.

Catatan penggunaan: tuliskan alasan pemakaian perintah, bukan hanya sintaksnya. Apabila perintah bergantung pada paket tambahan, catat versi paket dan sediakan sel instalasi. Bila perintah dapat gagal, tangkap pengecualian yang spesifik atau periksa nilai balik. Untuk proses yang menghasilkan berkas, verifikasi keberadaan, ukuran, encoding, dan kemampuan pembacaan ulang.

H.078 DataFrame.sort_values()

Fungsi utama: Mengurutkan baris. Perintah ini sebaiknya ditempatkan pada tahap notebook yang jelas dan disertai pemeriksaan input. Contoh minimal: `df=df.sort_values("energi")`. Setelah menjalankan contoh, periksa tipe, nilai, status, atau berkas yang dihasilkan. Jangan menafsirkan keluaran di luar konteks data dan metode.

Catatan penggunaan: tuliskan alasan pemakaian perintah, bukan hanya sintaksnya. Apabila perintah bergantung pada paket tambahan, catat versi paket dan sediakan sel instalasi. Bila perintah dapat gagal, tangkap pengecualian yang spesifik atau periksa nilai balik. Untuk proses yang menghasilkan berkas, verifikasi keberadaan, ukuran, encoding, dan kemampuan pembacaan ulang.

H.079 DataFrame.groupby()

Fungsi utama: Mengelompokkan data. Perintah ini sebaiknya ditempatkan pada tahap notebook yang jelas dan disertai pemeriksaan input. Contoh minimal: `g=df.groupby("metode")["energi"].mean()`. Setelah menjalankan contoh, periksa tipe, nilai, status, atau berkas yang dihasilkan. Jangan menafsirkan keluaran di luar konteks data dan metode.

Catatan penggunaan: tuliskan alasan pemakaian perintah, bukan hanya sintaksnya. Apabila perintah bergantung pada paket tambahan, catat versi paket dan sediakan sel instalasi. Bila perintah dapat gagal, tangkap pengecualian yang spesifik atau periksa nilai balik. Untuk proses yang menghasilkan berkas, verifikasi keberadaan, ukuran, encoding, dan kemampuan pembacaan ulang.

H.080 DataFrame.agg()

Fungsi utama: Agregasi beberapa fungsi. Perintah ini sebaiknya ditempatkan pada tahap notebook yang jelas dan disertai pemeriksaan input. Contoh minimal: `r=df.agg({"x":["mean","std"]})`. Setelah menjalankan contoh, periksa tipe, nilai, status, atau berkas yang dihasilkan. Jangan menafsirkan keluaran di luar konteks data dan metode.

Catatan penggunaan: tuliskan alasan pemakaian perintah, bukan hanya sintaksnya. Apabila perintah bergantung pada paket tambahan, catat versi paket dan sediakan sel

instalasi. Bila perintah dapat gagal, tangkap pengecualian yang spesifik atau periksa nilai balik. Untuk proses yang menghasilkan berkas, verifikasi keberadaan, ukuran, encoding, dan kemampuan pembacaan ulang.

H.081 DataFrame.merge()

Fungsi utama: Menggabungkan tabel. Perintah ini sebaiknya ditempatkan pada tahap notebook yang jelas dan disertai pemeriksaan input. Contoh minimal: `df=a.merge(b,on="id",validate="one_to_one")`. Setelah menjalankan contoh, periksa tipe, nilai, status, atau berkas yang dihasilkan. Jangan menafsirkan keluaran di luar konteks data dan metode.

Catatan penggunaan: tuliskan alasan pemakaian perintah, bukan hanya sintaksnya. Apabila perintah bergantung pada paket tambahan, catat versi paket dan sediakan sel instalasi. Bila perintah dapat gagal, tangkap pengecualian yang spesifik atau periksa nilai balik. Untuk proses yang menghasilkan berkas, verifikasi keberadaan, ukuran, encoding, dan kemampuan pembacaan ulang.

H.082 pd.concat()

Fungsi utama: Menggabungkan objek sepanjang sumbu. Perintah ini sebaiknya ditempatkan pada tahap notebook yang jelas dan disertai pemeriksaan input. Contoh minimal: `df=pd.concat([a,b],ignore_index=True)`. Setelah menjalankan contoh, periksa tipe, nilai, status, atau berkas yang dihasilkan. Jangan menafsirkan keluaran di luar konteks data dan metode.

Catatan penggunaan: tuliskan alasan pemakaian perintah, bukan hanya sintaksnya. Apabila perintah bergantung pada paket tambahan, catat versi paket dan sediakan sel instalasi. Bila perintah dapat gagal, tangkap pengecualian yang spesifik atau periksa nilai balik. Untuk proses yang menghasilkan berkas, verifikasi keberadaan, ukuran, encoding, dan kemampuan pembacaan ulang.

H.083 DataFrame.pivot_table()

Fungsi utama: Membuat tabel pivot. Perintah ini sebaiknya ditempatkan pada tahap notebook yang jelas dan disertai pemeriksaan input. Contoh minimal: `p=df.pivot_table(index="m",columns="b",values="E")`. Setelah menjalankan contoh, periksa tipe, nilai, status, atau berkas yang dihasilkan. Jangan menafsirkan keluaran di luar konteks data dan metode.

Catatan penggunaan: tuliskan alasan pemakaian perintah, bukan hanya sintaksnya. Apabila perintah bergantung pada paket tambahan, catat versi paket dan sediakan sel instalasi. Bila perintah dapat gagal, tangkap pengecualian yang spesifik atau periksa nilai balik. Untuk proses yang menghasilkan berkas, verifikasi keberadaan, ukuran, encoding, dan kemampuan pembacaan ulang.

H.Matplotlib

H.084 plt.figure()

Fungsi utama: Membuat figure baru. Perintah ini sebaiknya ditempatkan pada tahap notebook yang jelas dan disertai pemeriksaan input. Contoh minimal: `plt.figure(figsize=(7,4))`. Setelah menjalankan contoh, periksa tipe, nilai, status, atau berkas yang dihasilkan. Jangan menafsirkan keluaran di luar konteks data dan metode.

Catatan penggunaan: tuliskan alasan pemakaian perintah, bukan hanya sintaksnya. Apabila perintah bergantung pada paket tambahan, catat versi paket dan sediakan sel instalasi. Bila perintah dapat gagal, tangkap pengecualian yang spesifik atau periksa nilai balik. Untuk proses yang menghasilkan berkas, verifikasi keberadaan, ukuran, encoding, dan kemampuan pembacaan ulang.

H.085 plt.plot()

Fungsi utama: Membuat grafik garis. Perintah ini sebaiknya ditempatkan pada tahap notebook yang jelas dan disertai pemeriksaan input. Contoh minimal: `plt.plot(x,y,marker="o")`. Setelah menjalankan contoh, periksa tipe, nilai, status, atau berkas yang dihasilkan. Jangan menafsirkan keluaran di luar konteks data dan metode.

Catatan penggunaan: tuliskan alasan pemakaian perintah, bukan hanya sintaksnya. Apabila perintah bergantung pada paket tambahan, catat versi paket dan sediakan sel instalasi. Bila perintah dapat gagal, tangkap pengecualian yang spesifik atau periksa nilai balik. Untuk proses yang menghasilkan berkas, verifikasi keberadaan, ukuran, encoding, dan kemampuan pembacaan ulang.

H.086 plt.scatter()

Fungsi utama: Membuat diagram pencar. Perintah ini sebaiknya ditempatkan pada tahap notebook yang jelas dan disertai pemeriksaan input. Contoh minimal: `plt.scatter(x,y)`. Setelah menjalankan contoh, periksa tipe, nilai, status, atau berkas yang dihasilkan. Jangan menafsirkan keluaran di luar konteks data dan metode.

Catatan penggunaan: tuliskan alasan pemakaian perintah, bukan hanya sintaksnya. Apabila perintah bergantung pada paket tambahan, catat versi paket dan sediakan sel instalasi. Bila perintah dapat gagal, tangkap pengecualian yang spesifik atau periksa nilai balik. Untuk proses yang menghasilkan berkas, verifikasi keberadaan, ukuran, encoding, dan kemampuan pembacaan ulang.

H.087 plt.hist()

Fungsi utama: Membuat histogram. Perintah ini sebaiknya ditempatkan pada tahap notebook yang jelas dan disertai pemeriksaan input. Contoh minimal:

`plt.hist(data,bins=20)`. Setelah menjalankan contoh, periksa tipe, nilai, status, atau berkas yang dihasilkan. Jangan menafsirkan keluaran di luar konteks data dan metode.

Catatan penggunaan: tuliskan alasan pemakaian perintah, bukan hanya sintaksnya. Apabila perintah bergantung pada paket tambahan, catat versi paket dan sediakan sel instalasi. Bila perintah dapat gagal, tangkap pengecualian yang spesifik atau periksa nilai balik. Untuk proses yang menghasilkan berkas, verifikasi keberadaan, ukuran, encoding, dan kemampuan pembacaan ulang.

H.088 `plt.imshow()`

Fungsi utama: Menampilkan matriks sebagai gambar. Perintah ini sebaiknya ditempatkan pada tahap notebook yang jelas dan disertai pemeriksaan input. Contoh minimal: `plt.imshow(M)`. Setelah menjalankan contoh, periksa tipe, nilai, status, atau berkas yang dihasilkan. Jangan menafsirkan keluaran di luar konteks data dan metode.

Catatan penggunaan: tuliskan alasan pemakaian perintah, bukan hanya sintaksnya. Apabila perintah bergantung pada paket tambahan, catat versi paket dan sediakan sel instalasi. Bila perintah dapat gagal, tangkap pengecualian yang spesifik atau periksa nilai balik. Untuk proses yang menghasilkan berkas, verifikasi keberadaan, ukuran, encoding, dan kemampuan pembacaan ulang.

H.089 `plt.xlabel()`

Fungsi utama: Memberi label sumbu x. Perintah ini sebaiknya ditempatkan pada tahap notebook yang jelas dan disertai pemeriksaan input. Contoh minimal: `plt.xlabel("Jarak (Å)")`. Setelah menjalankan contoh, periksa tipe, nilai, status, atau berkas yang dihasilkan. Jangan menafsirkan keluaran di luar konteks data dan metode.

Catatan penggunaan: tuliskan alasan pemakaian perintah, bukan hanya sintaksnya. Apabila perintah bergantung pada paket tambahan, catat versi paket dan sediakan sel instalasi. Bila perintah dapat gagal, tangkap pengecualian yang spesifik atau periksa nilai balik. Untuk proses yang menghasilkan berkas, verifikasi keberadaan, ukuran, encoding, dan kemampuan pembacaan ulang.

H.090 `plt.ylabel()`

Fungsi utama: Memberi label sumbu y. Perintah ini sebaiknya ditempatkan pada tahap notebook yang jelas dan disertai pemeriksaan input. Contoh minimal: `plt.ylabel("Energi (Hartree)")`. Setelah menjalankan contoh, periksa tipe, nilai, status, atau berkas yang dihasilkan. Jangan menafsirkan keluaran di luar konteks data dan metode.

Catatan penggunaan: tuliskan alasan pemakaian perintah, bukan hanya sintaksnya. Apabila perintah bergantung pada paket tambahan, catat versi paket dan sediakan sel instalasi. Bila perintah dapat gagal, tangkap pengecualian yang spesifik atau periksa nilai

balik. Untuk proses yang menghasilkan berkas, verifikasi keberadaan, ukuran, encoding, dan kemampuan pembacaan ulang.

H.091 plt.title()

Fungsi utama: Memberi judul grafik. Perintah ini sebaiknya ditempatkan pada tahap notebook yang jelas dan disertai pemeriksaan input. Contoh minimal: `plt.title("Kurva Energi")`. Setelah menjalankan contoh, periksa tipe, nilai, status, atau berkas yang dihasilkan. Jangan menafsirkan keluaran di luar konteks data dan metode.

Catatan penggunaan: tuliskan alasan pemakaian perintah, bukan hanya sintaksnya. Apabila perintah bergantung pada paket tambahan, catat versi paket dan sediakan sel instalasi. Bila perintah dapat gagal, tangkap pengecualian yang spesifik atau periksa nilai balik. Untuk proses yang menghasilkan berkas, verifikasi keberadaan, ukuran, encoding, dan kemampuan pembacaan ulang.

H.092 plt.legend()

Fungsi utama: Menampilkan legenda. Perintah ini sebaiknya ditempatkan pada tahap notebook yang jelas dan disertai pemeriksaan input. Contoh minimal: `plt.legend()`. Setelah menjalankan contoh, periksa tipe, nilai, status, atau berkas yang dihasilkan. Jangan menafsirkan keluaran di luar konteks data dan metode.

Catatan penggunaan: tuliskan alasan pemakaian perintah, bukan hanya sintaksnya. Apabila perintah bergantung pada paket tambahan, catat versi paket dan sediakan sel instalasi. Bila perintah dapat gagal, tangkap pengecualian yang spesifik atau periksa nilai balik. Untuk proses yang menghasilkan berkas, verifikasi keberadaan, ukuran, encoding, dan kemampuan pembacaan ulang.

H.093 plt.grid()

Fungsi utama: Menampilkan grid. Perintah ini sebaiknya ditempatkan pada tahap notebook yang jelas dan disertai pemeriksaan input. Contoh minimal: `plt.grid(True,alpha=0.3)`. Setelah menjalankan contoh, periksa tipe, nilai, status, atau berkas yang dihasilkan. Jangan menafsirkan keluaran di luar konteks data dan metode.

Catatan penggunaan: tuliskan alasan pemakaian perintah, bukan hanya sintaksnya. Apabila perintah bergantung pada paket tambahan, catat versi paket dan sediakan sel instalasi. Bila perintah dapat gagal, tangkap pengecualian yang spesifik atau periksa nilai balik. Untuk proses yang menghasilkan berkas, verifikasi keberadaan, ukuran, encoding, dan kemampuan pembacaan ulang.

H.094 plt.tight_layout()

Fungsi utama: Menyesuaikan tata letak. Perintah ini sebaiknya ditempatkan pada tahap notebook yang jelas dan disertai pemeriksaan input. Contoh minimal: `plt.tight_layout()`.

Setelah menjalankan contoh, periksa tipe, nilai, status, atau berkas yang dihasilkan. Jangan menafsirkan keluaran di luar konteks data dan metode.

Catatan penggunaan: tuliskan alasan pemakaian perintah, bukan hanya sintaksnya. Apabila perintah bergantung pada paket tambahan, catat versi paket dan sediakan sel instalasi. Bila perintah dapat gagal, tangkap pengecualian yang spesifik atau periksa nilai balik. Untuk proses yang menghasilkan berkas, verifikasi keberadaan, ukuran, encoding, dan kemampuan pembacaan ulang.

H.095 plt.savefig()

Fungsi utama: Menyimpan gambar. Perintah ini sebaiknya ditempatkan pada tahap notebook yang jelas dan disertai pemeriksaan input. Contoh minimal: `plt.savefig("grafik.png",dpi=200)`. Setelah menjalankan contoh, periksa tipe, nilai, status, atau berkas yang dihasilkan. Jangan menafsirkan keluaran di luar konteks data dan metode.

Catatan penggunaan: tuliskan alasan pemakaian perintah, bukan hanya sintaksnya. Apabila perintah bergantung pada paket tambahan, catat versi paket dan sediakan sel instalasi. Bila perintah dapat gagal, tangkap pengecualian yang spesifik atau periksa nilai balik. Untuk proses yang menghasilkan berkas, verifikasi keberadaan, ukuran, encoding, dan kemampuan pembacaan ulang.

H.096 plt.show()

Fungsi utama: Menampilkan figure. Perintah ini sebaiknya ditempatkan pada tahap notebook yang jelas dan disertai pemeriksaan input. Contoh minimal: `plt.show()`. Setelah menjalankan contoh, periksa tipe, nilai, status, atau berkas yang dihasilkan. Jangan menafsirkan keluaran di luar konteks data dan metode.

Catatan penggunaan: tuliskan alasan pemakaian perintah, bukan hanya sintaksnya. Apabila perintah bergantung pada paket tambahan, catat versi paket dan sediakan sel instalasi. Bila perintah dapat gagal, tangkap pengecualian yang spesifik atau periksa nilai balik. Untuk proses yang menghasilkan berkas, verifikasi keberadaan, ukuran, encoding, dan kemampuan pembacaan ulang.

H.Colab

H.097 %pip install

Fungsi utama: Memasang paket pada runtime. Perintah ini sebaiknya ditempatkan pada tahap notebook yang jelas dan disertai pemeriksaan input. Contoh minimal: `%pip -q install numpy pandas`. Setelah menjalankan contoh, periksa tipe, nilai, status, atau berkas yang dihasilkan. Jangan menafsirkan keluaran di luar konteks data dan metode.

Catatan penggunaan: tuliskan alasan pemakaian perintah, bukan hanya sintaksnya. Apabila perintah bergantung pada paket tambahan, catat versi paket dan sediakan sel instalasi. Bila perintah dapat gagal, tangkap pengecualian yang spesifik atau periksa nilai balik. Untuk proses yang menghasilkan berkas, verifikasi keberadaan, ukuran, encoding, dan kemampuan pembacaan ulang.

H.098 !pwd

Fungsi utama: Menampilkan direktori kerja melalui shell. Perintah ini sebaiknya ditempatkan pada tahap notebook yang jelas dan disertai pemeriksaan input. Contoh minimal: `!pwd`. Setelah menjalankan contoh, periksa tipe, nilai, status, atau berkas yang dihasilkan. Jangan menafsirkan keluaran di luar konteks data dan metode.

Catatan penggunaan: tuliskan alasan pemakaian perintah, bukan hanya sintaksnya. Apabila perintah bergantung pada paket tambahan, catat versi paket dan sediakan sel instalasi. Bila perintah dapat gagal, tangkap pengecualian yang spesifik atau periksa nilai balik. Untuk proses yang menghasilkan berkas, verifikasi keberadaan, ukuran, encoding, dan kemampuan pembacaan ulang.

H.099 !ls -lah

Fungsi utama: Mendaftar berkas. Perintah ini sebaiknya ditempatkan pada tahap notebook yang jelas dan disertai pemeriksaan input. Contoh minimal: `!ls -lah`. Setelah menjalankan contoh, periksa tipe, nilai, status, atau berkas yang dihasilkan. Jangan menafsirkan keluaran di luar konteks data dan metode.

Catatan penggunaan: tuliskan alasan pemakaian perintah, bukan hanya sintaksnya. Apabila perintah bergantung pada paket tambahan, catat versi paket dan sediakan sel instalasi. Bila perintah dapat gagal, tangkap pengecualian yang spesifik atau periksa nilai balik. Untuk proses yang menghasilkan berkas, verifikasi keberadaan, ukuran, encoding, dan kemampuan pembacaan ulang.

H.100 files.upload()

Fungsi utama: Mengunggah berkas. Perintah ini sebaiknya ditempatkan pada tahap notebook yang jelas dan disertai pemeriksaan input. Contoh minimal: `uploaded=files.upload()`. Setelah menjalankan contoh, periksa tipe, nilai, status, atau berkas yang dihasilkan. Jangan menafsirkan keluaran di luar konteks data dan metode.

Catatan penggunaan: tuliskan alasan pemakaian perintah, bukan hanya sintaksnya. Apabila perintah bergantung pada paket tambahan, catat versi paket dan sediakan sel instalasi. Bila perintah dapat gagal, tangkap pengecualian yang spesifik atau periksa nilai balik. Untuk proses yang menghasilkan berkas, verifikasi keberadaan, ukuran, encoding, dan kemampuan pembacaan ulang.

H.101 files.download()

Fungsi utama: Mengunduh berkas. Perintah ini sebaiknya ditempatkan pada tahap notebook yang jelas dan disertai pemeriksaan input. Contoh minimal: `files.download("hasil.zip")`. Setelah menjalankan contoh, periksa tipe, nilai, status, atau berkas yang dihasilkan. Jangan menafsirkan keluaran di luar konteks data dan metode.

Catatan penggunaan: tuliskan alasan pemakaian perintah, bukan hanya sintaksnya. Apabila perintah bergantung pada paket tambahan, catat versi paket dan sediakan sel instalasi. Bila perintah dapat gagal, tangkap pengecualian yang spesifik atau periksa nilai balik. Untuk proses yang menghasilkan berkas, verifikasi keberadaan, ukuran, encoding, dan kemampuan pembacaan ulang.

H.102 drive.mount()

Fungsi utama: Menghubungkan Google Drive. Perintah ini sebaiknya ditempatkan pada tahap notebook yang jelas dan disertai pemeriksaan input. Contoh minimal: `drive.mount("/content/drive")`. Setelah menjalankan contoh, periksa tipe, nilai, status, atau berkas yang dihasilkan. Jangan menafsirkan keluaran di luar konteks data dan metode.

Catatan penggunaan: tuliskan alasan pemakaian perintah, bukan hanya sintaksnya. Apabila perintah bergantung pada paket tambahan, catat versi paket dan sediakan sel instalasi. Bila perintah dapat gagal, tangkap pengecualian yang spesifik atau periksa nilai balik. Untuk proses yang menghasilkan berkas, verifikasi keberadaan, ukuran, encoding, dan kemampuan pembacaan ulang.

H.103 nvidia-smi

Fungsi utama: Memeriksa GPU NVIDIA. Perintah ini sebaiknya ditempatkan pada tahap notebook yang jelas dan disertai pemeriksaan input. Contoh minimal: `!nvidia-smi`. Setelah menjalankan contoh, periksa tipe, nilai, status, atau berkas yang dihasilkan. Jangan menafsirkan keluaran di luar konteks data dan metode.

Catatan penggunaan: tuliskan alasan pemakaian perintah, bukan hanya sintaksnya. Apabila perintah bergantung pada paket tambahan, catat versi paket dan sediakan sel instalasi. Bila perintah dapat gagal, tangkap pengecualian yang spesifik atau periksa nilai balik. Untuk proses yang menghasilkan berkas, verifikasi keberadaan, ukuran, encoding, dan kemampuan pembacaan ulang.

H.104 subprocess.run()

Fungsi utama: Menjalankan proses eksternal dengan status. Perintah ini sebaiknya ditempatkan pada tahap notebook yang jelas dan disertai pemeriksaan input. Contoh minimal: `r=subprocess.run(cmd,capture_output=True,text=True)`. Setelah menjalankan

contoh, periksa tipe, nilai, status, atau berkas yang dihasilkan. Jangan menafsirkan keluaran di luar konteks data dan metode.

Catatan penggunaan: tuliskan alasan pemakaian perintah, bukan hanya sintaksnya. Apabila perintah bergantung pada paket tambahan, catat versi paket dan sediakan sel instalasi. Bila perintah dapat gagal, tangkap pengecualian yang spesifik atau periksa nilai balik. Untuk proses yang menghasilkan berkas, verifikasi keberadaan, ukuran, encoding, dan kemampuan pembacaan ulang.

H.Archive

H.105 ZipFile()

Fungsi utama: Membuat atau membaca ZIP. Perintah ini sebaiknya ditempatkan pada tahap notebook yang jelas dan disertai pemeriksaan input. Contoh minimal: `with ZipFile("a.zip","w") as z: pass`. Setelah menjalankan contoh, periksa tipe, nilai, status, atau berkas yang dihasilkan. Jangan menafsirkan keluaran di luar konteks data dan metode.

Catatan penggunaan: tuliskan alasan pemakaian perintah, bukan hanya sintaksnya. Apabila perintah bergantung pada paket tambahan, catat versi paket dan sediakan sel instalasi. Bila perintah dapat gagal, tangkap pengecualian yang spesifik atau periksa nilai balik. Untuk proses yang menghasilkan berkas, verifikasi keberadaan, ukuran, encoding, dan kemampuan pembacaan ulang.

H.106 shutil.copy()

Fungsi utama: Menyalin berkas. Perintah ini sebaiknya ditempatkan pada tahap notebook yang jelas dan disertai pemeriksaan input. Contoh minimal: `shutil.copy(src,dst)`. Setelah menjalankan contoh, periksa tipe, nilai, status, atau berkas yang dihasilkan. Jangan menafsirkan keluaran di luar konteks data dan metode.

Catatan penggunaan: tuliskan alasan pemakaian perintah, bukan hanya sintaksnya. Apabila perintah bergantung pada paket tambahan, catat versi paket dan sediakan sel instalasi. Bila perintah dapat gagal, tangkap pengecualian yang spesifik atau periksa nilai balik. Untuk proses yang menghasilkan berkas, verifikasi keberadaan, ukuran, encoding, dan kemampuan pembacaan ulang.

H.107 shutil.disk_usage()

Fungsi utama: Memeriksa ruang disk. Perintah ini sebaiknya ditempatkan pada tahap notebook yang jelas dan disertai pemeriksaan input. Contoh minimal: `total,used,free=shutil.disk_usage("/content")`. Setelah menjalankan contoh, periksa tipe, nilai, status, atau berkas yang dihasilkan. Jangan menafsirkan keluaran di luar konteks data dan metode.

Catatan penggunaan: tuliskan alasan pemakaian perintah, bukan hanya sintaksnya. Apabila perintah bergantung pada paket tambahan, catat versi paket dan sediakan sel instalasi. Bila perintah dapat gagal, tangkap pengecualian yang spesifik atau periksa nilai balik. Untuk proses yang menghasilkan berkas, verifikasi keberadaan, ukuran, encoding, dan kemampuan pembacaan ulang.

H.Hash

H.108 hashlib.sha256()

Fungsi utama: Membuat hash SHA-256. Perintah ini sebaiknya ditempatkan pada tahap notebook yang jelas dan disertai pemeriksaan input. Contoh minimal: `h=hashlib.sha256(data).hexdigest()`. Setelah menjalankan contoh, periksa tipe, nilai, status, atau berkas yang dihasilkan. Jangan menafsirkan keluaran di luar konteks data dan metode.

Catatan penggunaan: tuliskan alasan pemakaian perintah, bukan hanya sintaksnya. Apabila perintah bergantung pada paket tambahan, catat versi paket dan sediakan sel instalasi. Bila perintah dapat gagal, tangkap pengecualian yang spesifik atau periksa nilai balik. Untuk proses yang menghasilkan berkas, verifikasi keberadaan, ukuran, encoding, dan kemampuan pembacaan ulang.

H.Timing

H.109 perf_counter()

Fungsi utama: Pengukur waktu resolusi tinggi. Perintah ini sebaiknya ditempatkan pada tahap notebook yang jelas dan disertai pemeriksaan input. Contoh minimal: `t0=perf_counter()`. Setelah menjalankan contoh, periksa tipe, nilai, status, atau berkas yang dihasilkan. Jangan menafsirkan keluaran di luar konteks data dan metode.

Catatan penggunaan: tuliskan alasan pemakaian perintah, bukan hanya sintaksnya. Apabila perintah bergantung pada paket tambahan, catat versi paket dan sediakan sel instalasi. Bila perintah dapat gagal, tangkap pengecualian yang spesifik atau periksa nilai balik. Untuk proses yang menghasilkan berkas, verifikasi keberadaan, ukuran, encoding, dan kemampuan pembacaan ulang.

H.Metadata

H.110 importlib.metadata.version()

Fungsi utama: Membaca versi distribusi. Perintah ini sebaiknya ditempatkan pada tahap notebook yang jelas dan disertai pemeriksaan input. Contoh minimal: `v=version("numpy")`. Setelah menjalankan contoh, periksa tipe, nilai, status, atau berkas yang dihasilkan. Jangan menafsirkan keluaran di luar konteks data dan metode.

Catatan penggunaan: tuliskan alasan pemakaian perintah, bukan hanya sintaksnya. Apabila perintah bergantung pada paket tambahan, catat versi paket dan sediakan sel instalasi. Bila perintah dapat gagal, tangkap pengecualian yang spesifik atau periksa nilai balik. Untuk proses yang menghasilkan berkas, verifikasi keberadaan, ukuran, encoding, dan kemampuan pembacaan ulang.

H.RDKit

H.111 Chem.MolFromSmiles()

Fungsi utama: Membuat molekul dari SMILES. Perintah ini sebaiknya ditempatkan pada tahap notebook yang jelas dan disertai pemeriksaan input. Contoh minimal: `mol=Chem.MolFromSmiles("CCO")`. Setelah menjalankan contoh, periksa tipe, nilai, status, atau berkas yang dihasilkan. Jangan menafsirkan keluaran di luar konteks data dan metode.

Catatan penggunaan: tuliskan alasan pemakaian perintah, bukan hanya sintaksnya. Apabila perintah bergantung pada paket tambahan, catat versi paket dan sediakan sel instalasi. Bila perintah dapat gagal, tangkap pengecualian yang spesifik atau periksa nilai balik. Untuk proses yang menghasilkan berkas, verifikasi keberadaan, ukuran, encoding, dan kemampuan pembacaan ulang.

H.112 Chem.AddHs()

Fungsi utama: Menambah hidrogen eksplisit. Perintah ini sebaiknya ditempatkan pada tahap notebook yang jelas dan disertai pemeriksaan input. Contoh minimal: `mol=Chem.AddHs(mol)`. Setelah menjalankan contoh, periksa tipe, nilai, status, atau berkas yang dihasilkan. Jangan menafsirkan keluaran di luar konteks data dan metode.

Catatan penggunaan: tuliskan alasan pemakaian perintah, bukan hanya sintaksnya. Apabila perintah bergantung pada paket tambahan, catat versi paket dan sediakan sel instalasi. Bila perintah dapat gagal, tangkap pengecualian yang spesifik atau periksa nilai balik. Untuk proses yang menghasilkan berkas, verifikasi keberadaan, ukuran, encoding, dan kemampuan pembacaan ulang.

H.113 AllChem.ETKDGv3()

Fungsi utama: Parameter embedding 3D. Perintah ini sebaiknya ditempatkan pada tahap notebook yang jelas dan disertai pemeriksaan input. Contoh minimal: `params=AllChem.ETKDGv3()`. Setelah menjalankan contoh, periksa tipe, nilai, status, atau berkas yang dihasilkan. Jangan menafsirkan keluaran di luar konteks data dan metode.

Catatan penggunaan: tuliskan alasan pemakaian perintah, bukan hanya sintaksnya. Apabila perintah bergantung pada paket tambahan, catat versi paket dan sediakan sel

instalasi. Bila perintah dapat gagal, tangkap pengecualian yang spesifik atau periksa nilai balik. Untuk proses yang menghasilkan berkas, verifikasi keberadaan, ukuran, encoding, dan kemampuan pembacaan ulang.

H.114 AllChem.EmbedMolecule()

Fungsi utama: Membuat konformer 3D. Perintah ini sebaiknya ditempatkan pada tahap notebook yang jelas dan disertai pemeriksaan input. Contoh minimal: `status=AllChem.EmbedMolecule(mol,params)`. Setelah menjalankan contoh, periksa tipe, nilai, status, atau berkas yang dihasilkan. Jangan menafsirkan keluaran di luar konteks data dan metode.

Catatan penggunaan: tuliskan alasan pemakaian perintah, bukan hanya sintaksnya. Apabila perintah bergantung pada paket tambahan, catat versi paket dan sediakan sel instalasi. Bila perintah dapat gagal, tangkap pengecualian yang spesifik atau periksa nilai balik. Untuk proses yang menghasilkan berkas, verifikasi keberadaan, ukuran, encoding, dan kemampuan pembacaan ulang.

H.115 AllChem.EmbedMultipleConfs()

Fungsi utama: Membuat beberapa konformer. Perintah ini sebaiknya ditempatkan pada tahap notebook yang jelas dan disertai pemeriksaan input. Contoh minimal: `ids=AllChem.EmbedMultipleConfs(mol,numConfs=10)`. Setelah menjalankan contoh, periksa tipe, nilai, status, atau berkas yang dihasilkan. Jangan menafsirkan keluaran di luar konteks data dan metode.

Catatan penggunaan: tuliskan alasan pemakaian perintah, bukan hanya sintaksnya. Apabila perintah bergantung pada paket tambahan, catat versi paket dan sediakan sel instalasi. Bila perintah dapat gagal, tangkap pengecualian yang spesifik atau periksa nilai balik. Untuk proses yang menghasilkan berkas, verifikasi keberadaan, ukuran, encoding, dan kemampuan pembacaan ulang.

H.116 AllChem.MMFFHasAllMoleculeParams()

Fungsi utama: Memeriksa parameter MMFF. Perintah ini sebaiknya ditempatkan pada tahap notebook yang jelas dan disertai pemeriksaan input. Contoh minimal: `ok=AllChem.MMFFHasAllMoleculeParams(mol)`. Setelah menjalankan contoh, periksa tipe, nilai, status, atau berkas yang dihasilkan. Jangan menafsirkan keluaran di luar konteks data dan metode.

Catatan penggunaan: tuliskan alasan pemakaian perintah, bukan hanya sintaksnya. Apabila perintah bergantung pada paket tambahan, catat versi paket dan sediakan sel instalasi. Bila perintah dapat gagal, tangkap pengecualian yang spesifik atau periksa nilai balik. Untuk proses yang menghasilkan berkas, verifikasi keberadaan, ukuran, encoding, dan kemampuan pembacaan ulang.

H.117 AllChem.MMFFOptimizeMolecule()

Fungsi utama: Optimasi MMFF. Perintah ini sebaiknya ditempatkan pada tahap notebook yang jelas dan disertai pemeriksaan input. Contoh minimal: `status=AllChem.MMFFOptimizeMolecule(mol)`. Setelah menjalankan contoh, periksa tipe, nilai, status, atau berkas yang dihasilkan. Jangan menafsirkan keluaran di luar konteks data dan metode.

Catatan penggunaan: tuliskan alasan pemakaian perintah, bukan hanya sintaksnya. Apabila perintah bergantung pada paket tambahan, catat versi paket dan sediakan sel instalasi. Bila perintah dapat gagal, tangkap pengecualian yang spesifik atau periksa nilai balik. Untuk proses yang menghasilkan berkas, verifikasi keberadaan, ukuran, encoding, dan kemampuan pembacaan ulang.

H.118 AllChem.UFFOptimizeMolecule()

Fungsi utama: Optimasi UFF. Perintah ini sebaiknya ditempatkan pada tahap notebook yang jelas dan disertai pemeriksaan input. Contoh minimal: `status=AllChem.UFFOptimizeMolecule(mol)`. Setelah menjalankan contoh, periksa tipe, nilai, status, atau berkas yang dihasilkan. Jangan menafsirkan keluaran di luar konteks data dan metode.

Catatan penggunaan: tuliskan alasan pemakaian perintah, bukan hanya sintaksnya. Apabila perintah bergantung pada paket tambahan, catat versi paket dan sediakan sel instalasi. Bila perintah dapat gagal, tangkap pengecualian yang spesifik atau periksa nilai balik. Untuk proses yang menghasilkan berkas, verifikasi keberadaan, ukuran, encoding, dan kemampuan pembacaan ulang.

H.119 mol.GetNumAtoms()

Fungsi utama: Jumlah atom. Perintah ini sebaiknya ditempatkan pada tahap notebook yang jelas dan disertai pemeriksaan input. Contoh minimal: `n=mol.GetNumAtoms()`. Setelah menjalankan contoh, periksa tipe, nilai, status, atau berkas yang dihasilkan. Jangan menafsirkan keluaran di luar konteks data dan metode.

Catatan penggunaan: tuliskan alasan pemakaian perintah, bukan hanya sintaksnya. Apabila perintah bergantung pada paket tambahan, catat versi paket dan sediakan sel instalasi. Bila perintah dapat gagal, tangkap pengecualian yang spesifik atau periksa nilai balik. Untuk proses yang menghasilkan berkas, verifikasi keberadaan, ukuran, encoding, dan kemampuan pembacaan ulang.

H.120 mol.GetConformer()

Fungsi utama: Mengambil konformer. Perintah ini sebaiknya ditempatkan pada tahap notebook yang jelas dan disertai pemeriksaan input. Contoh minimal:

`conf=mol.GetConformer()`. Setelah menjalankan contoh, periksa tipe, nilai, status, atau berkas yang dihasilkan. Jangan menafsirkan keluaran di luar konteks data dan metode.

Catatan penggunaan: tuliskan alasan pemakaian perintah, bukan hanya sintaksnya. Apabila perintah bergantung pada paket tambahan, catat versi paket dan sediakan sel instalasi. Bila perintah dapat gagal, tangkap pengecualian yang spesifik atau periksa nilai balik. Untuk proses yang menghasilkan berkas, verifikasi keberadaan, ukuran, encoding, dan kemampuan pembacaan ulang.

H.121 atom.GetSymbol()

Fungsi utama: Simbol unsur atom. Perintah ini sebaiknya ditempatkan pada tahap notebook yang jelas dan disertai pemeriksaan input. Contoh minimal: `simbol=atom.GetSymbol()`. Setelah menjalankan contoh, periksa tipe, nilai, status, atau berkas yang dihasilkan. Jangan menafsirkan keluaran di luar konteks data dan metode.

Catatan penggunaan: tuliskan alasan pemakaian perintah, bukan hanya sintaksnya. Apabila perintah bergantung pada paket tambahan, catat versi paket dan sediakan sel instalasi. Bila perintah dapat gagal, tangkap pengecualian yang spesifik atau periksa nilai balik. Untuk proses yang menghasilkan berkas, verifikasi keberadaan, ukuran, encoding, dan kemampuan pembacaan ulang.

H.122 atom.GetIdx()

Fungsi utama: Indeks atom. Perintah ini sebaiknya ditempatkan pada tahap notebook yang jelas dan disertai pemeriksaan input. Contoh minimal: `i=atom.GetIdx()`. Setelah menjalankan contoh, periksa tipe, nilai, status, atau berkas yang dihasilkan. Jangan menafsirkan keluaran di luar konteks data dan metode.

Catatan penggunaan: tuliskan alasan pemakaian perintah, bukan hanya sintaksnya. Apabila perintah bergantung pada paket tambahan, catat versi paket dan sediakan sel instalasi. Bila perintah dapat gagal, tangkap pengecualian yang spesifik atau periksa nilai balik. Untuk proses yang menghasilkan berkas, verifikasi keberadaan, ukuran, encoding, dan kemampuan pembacaan ulang.

H.123 Chem.MolToMolBlock()

Fungsi utama: Mengubah molekul menjadi blok MOL. Perintah ini sebaiknya ditempatkan pada tahap notebook yang jelas dan disertai pemeriksaan input. Contoh minimal: `blok=Chem.MolToMolBlock(mol)`. Setelah menjalankan contoh, periksa tipe, nilai, status, atau berkas yang dihasilkan. Jangan menafsirkan keluaran di luar konteks data dan metode.

Catatan penggunaan: tuliskan alasan pemakaian perintah, bukan hanya sintaksnya. Apabila perintah bergantung pada paket tambahan, catat versi paket dan sediakan sel instalasi. Bila perintah dapat gagal, tangkap pengecualian yang spesifik atau periksa nilai

balik. Untuk proses yang menghasilkan berkas, verifikasi keberadaan, ukuran, encoding, dan kemampuan pembacaan ulang.

H.py3Dmol

H.124 *py3Dmol.view()*

Fungsi utama: Membuat kanvas 3D. Perintah ini sebaiknya ditempatkan pada tahap notebook yang jelas dan disertai pemeriksaan input. Contoh minimal: `view=py3Dmol.view(width=600,height=400)`. Setelah menjalankan contoh, periksa tipe, nilai, status, atau berkas yang dihasilkan. Jangan menafsirkan keluaran di luar konteks data dan metode.

Catatan penggunaan: tuliskan alasan pemakaian perintah, bukan hanya sintaksnya. Apabila perintah bergantung pada paket tambahan, catat versi paket dan sediakan sel instalasi. Bila perintah dapat gagal, tangkap pengecualian yang spesifik atau periksa nilai balik. Untuk proses yang menghasilkan berkas, verifikasi keberadaan, ukuran, encoding, dan kemampuan pembacaan ulang.

H.125 *view.addModel()*

Fungsi utama: Menambah model molekul. Perintah ini sebaiknya ditempatkan pada tahap notebook yang jelas dan disertai pemeriksaan input. Contoh minimal: `view.addModel(xyz,"xyz")`. Setelah menjalankan contoh, periksa tipe, nilai, status, atau berkas yang dihasilkan. Jangan menafsirkan keluaran di luar konteks data dan metode.

Catatan penggunaan: tuliskan alasan pemakaian perintah, bukan hanya sintaksnya. Apabila perintah bergantung pada paket tambahan, catat versi paket dan sediakan sel instalasi. Bila perintah dapat gagal, tangkap pengecualian yang spesifik atau periksa nilai balik. Untuk proses yang menghasilkan berkas, verifikasi keberadaan, ukuran, encoding, dan kemampuan pembacaan ulang.

H.126 *view.setStyle()*

Fungsi utama: Mengatur gaya visual. Perintah ini sebaiknya ditempatkan pada tahap notebook yang jelas dan disertai pemeriksaan input. Contoh minimal: `view.setStyle({"stick":{}})`. Setelah menjalankan contoh, periksa tipe, nilai, status, atau berkas yang dihasilkan. Jangan menafsirkan keluaran di luar konteks data dan metode.

Catatan penggunaan: tuliskan alasan pemakaian perintah, bukan hanya sintaksnya. Apabila perintah bergantung pada paket tambahan, catat versi paket dan sediakan sel instalasi. Bila perintah dapat gagal, tangkap pengecualian yang spesifik atau periksa nilai balik. Untuk proses yang menghasilkan berkas, verifikasi keberadaan, ukuran, encoding, dan kemampuan pembacaan ulang.

H.127 view.zoomTo()

Fungsi utama: Memusatkan model. Perintah ini sebaiknya ditempatkan pada tahap notebook yang jelas dan disertai pemeriksaan input. Contoh minimal: `view.zoomTo()`. Setelah menjalankan contoh, periksa tipe, nilai, status, atau berkas yang dihasilkan. Jangan menafsirkan keluaran di luar konteks data dan metode.

Catatan penggunaan: tuliskan alasan pemakaian perintah, bukan hanya sintaksnya. Apabila perintah bergantung pada paket tambahan, catat versi paket dan sediakan sel instalasi. Bila perintah dapat gagal, tangkap pengecualian yang spesifik atau periksa nilai balik. Untuk proses yang menghasilkan berkas, verifikasi keberadaan, ukuran, encoding, dan kemampuan pembacaan ulang.

H.128 view.addLabel()

Fungsi utama: Menambah label. Perintah ini sebaiknya ditempatkan pada tahap notebook yang jelas dan disertai pemeriksaan input. Contoh minimal: `view.addLabel("O", {"position": {"x":0, "y":0, "z":0}})`. Setelah menjalankan contoh, periksa tipe, nilai, status, atau berkas yang dihasilkan. Jangan menafsirkan keluaran di luar konteks data dan metode.

Catatan penggunaan: tuliskan alasan pemakaian perintah, bukan hanya sintaksnya. Apabila perintah bergantung pada paket tambahan, catat versi paket dan sediakan sel instalasi. Bila perintah dapat gagal, tangkap pengecualian yang spesifik atau periksa nilai balik. Untuk proses yang menghasilkan berkas, verifikasi keberadaan, ukuran, encoding, dan kemampuan pembacaan ulang.

H.129 view.show()

Fungsi utama: Menampilkan visualisasi. Perintah ini sebaiknya ditempatkan pada tahap notebook yang jelas dan disertai pemeriksaan input. Contoh minimal: `view.show()`. Setelah menjalankan contoh, periksa tipe, nilai, status, atau berkas yang dihasilkan. Jangan menafsirkan keluaran di luar konteks data dan metode.

Catatan penggunaan: tuliskan alasan pemakaian perintah, bukan hanya sintaksnya. Apabila perintah bergantung pada paket tambahan, catat versi paket dan sediakan sel instalasi. Bila perintah dapat gagal, tangkap pengecualian yang spesifik atau periksa nilai balik. Untuk proses yang menghasilkan berkas, verifikasi keberadaan, ukuran, encoding, dan kemampuan pembacaan ulang.

H.PySCF

H.130 gto.M()

Fungsi utama: Membangun objek molekul. Perintah ini sebaiknya ditempatkan pada tahap notebook yang jelas dan disertai pemeriksaan input. Contoh minimal: `mol=gto.M(atom=geom,basis="sto-3g",charge=0,spin=0)`. Setelah menjalankan contoh, periksa tipe, nilai, status, atau berkas yang dihasilkan. Jangan menafsirkan keluaran di luar konteks data dan metode.

Catatan penggunaan: tuliskan alasan pemakaian perintah, bukan hanya sintaksnya. Apabila perintah bergantung pada paket tambahan, catat versi paket dan sediakan sel instalasi. Bila perintah dapat gagal, tangkap pengecualian yang spesifik atau periksa nilai balik. Untuk proses yang menghasilkan berkas, verifikasi keberadaan, ukuran, encoding, dan kemampuan pembacaan ulang.

H.131 mol.natm

Fungsi utama: Jumlah atom pada objek PySCF. Perintah ini sebaiknya ditempatkan pada tahap notebook yang jelas dan disertai pemeriksaan input. Contoh minimal: `print(mol.natm)`. Setelah menjalankan contoh, periksa tipe, nilai, status, atau berkas yang dihasilkan. Jangan menafsirkan keluaran di luar konteks data dan metode.

Catatan penggunaan: tuliskan alasan pemakaian perintah, bukan hanya sintaksnya. Apabila perintah bergantung pada paket tambahan, catat versi paket dan sediakan sel instalasi. Bila perintah dapat gagal, tangkap pengecualian yang spesifik atau periksa nilai balik. Untuk proses yang menghasilkan berkas, verifikasi keberadaan, ukuran, encoding, dan kemampuan pembacaan ulang.

H.132 mol.nelectron

Fungsi utama: Jumlah elektron. Perintah ini sebaiknya ditempatkan pada tahap notebook yang jelas dan disertai pemeriksaan input. Contoh minimal: `print(mol.nelectron)`. Setelah menjalankan contoh, periksa tipe, nilai, status, atau berkas yang dihasilkan. Jangan menafsirkan keluaran di luar konteks data dan metode.

Catatan penggunaan: tuliskan alasan pemakaian perintah, bukan hanya sintaksnya. Apabila perintah bergantung pada paket tambahan, catat versi paket dan sediakan sel instalasi. Bila perintah dapat gagal, tangkap pengecualian yang spesifik atau periksa nilai balik. Untuk proses yang menghasilkan berkas, verifikasi keberadaan, ukuran, encoding, dan kemampuan pembacaan ulang.

H.133 mol.atom_coords()

Fungsi utama: Koordinat atom. Perintah ini sebaiknya ditempatkan pada tahap notebook yang jelas dan disertai pemeriksaan input. Contoh minimal: `xyz=mol.atom_coords(unit="Angstrom")`. Setelah menjalankan contoh, periksa tipe, nilai, status, atau berkas yang dihasilkan. Jangan menafsirkan keluaran di luar konteks data dan metode.

Catatan penggunaan: tuliskan alasan pemakaian perintah, bukan hanya sintaksnya. Apabila perintah bergantung pada paket tambahan, catat versi paket dan sediakan sel instalasi. Bila perintah dapat gagal, tangkap pengecualian yang spesifik atau periksa nilai balik. Untuk proses yang menghasilkan berkas, verifikasi keberadaan, ukuran, encoding, dan kemampuan pembacaan ulang.

H.134 mol.atom_symbol()

Fungsi utama: Simbol atom berdasarkan indeks. Perintah ini sebaiknya ditempatkan pada tahap notebook yang jelas dan disertai pemeriksaan input. Contoh minimal: `s=mol.atom_symbol(i)`. Setelah menjalankan contoh, periksa tipe, nilai, status, atau berkas yang dihasilkan. Jangan menafsirkan keluaran di luar konteks data dan metode.

Catatan penggunaan: tuliskan alasan pemakaian perintah, bukan hanya sintaksnya. Apabila perintah bergantung pada paket tambahan, catat versi paket dan sediakan sel instalasi. Bila perintah dapat gagal, tangkap pengecualian yang spesifik atau periksa nilai balik. Untuk proses yang menghasilkan berkas, verifikasi keberadaan, ukuran, encoding, dan kemampuan pembacaan ulang.

H.135 scf.RHF()

Fungsi utama: Restricted Hartree-Fock. Perintah ini sebaiknya ditempatkan pada tahap notebook yang jelas dan disertai pemeriksaan input. Contoh minimal: `mf=scf.RHF(mol)`. Setelah menjalankan contoh, periksa tipe, nilai, status, atau berkas yang dihasilkan. Jangan menafsirkan keluaran di luar konteks data dan metode.

Catatan penggunaan: tuliskan alasan pemakaian perintah, bukan hanya sintaksnya. Apabila perintah bergantung pada paket tambahan, catat versi paket dan sediakan sel instalasi. Bila perintah dapat gagal, tangkap pengecualian yang spesifik atau periksa nilai balik. Untuk proses yang menghasilkan berkas, verifikasi keberadaan, ukuran, encoding, dan kemampuan pembacaan ulang.

H.136 scf.UHF()

Fungsi utama: Unrestricted Hartree-Fock. Perintah ini sebaiknya ditempatkan pada tahap notebook yang jelas dan disertai pemeriksaan input. Contoh minimal: `mf=scf.UHF(mol)`.

Setelah menjalankan contoh, periksa tipe, nilai, status, atau berkas yang dihasilkan. Jangan menafsirkan keluaran di luar konteks data dan metode.

Catatan penggunaan: tuliskan alasan pemakaian perintah, bukan hanya sintaksnya. Apabila perintah bergantung pada paket tambahan, catat versi paket dan sediakan sel instalasi. Bila perintah dapat gagal, tangkap pengecualian yang spesifik atau periksa nilai balik. Untuk proses yang menghasilkan berkas, verifikasi keberadaan, ukuran, encoding, dan kemampuan pembacaan ulang.

H.137 dft.RKS()

Fungsi utama: Restricted Kohn-Sham DFT. Perintah ini sebaiknya ditempatkan pada tahap notebook yang jelas dan disertai pemeriksaan input. Contoh minimal: `mf=dft.RKS(mol)`. Setelah menjalankan contoh, periksa tipe, nilai, status, atau berkas yang dihasilkan. Jangan menafsirkan keluaran di luar konteks data dan metode.

Catatan penggunaan: tuliskan alasan pemakaian perintah, bukan hanya sintaksnya. Apabila perintah bergantung pada paket tambahan, catat versi paket dan sediakan sel instalasi. Bila perintah dapat gagal, tangkap pengecualian yang spesifik atau periksa nilai balik. Untuk proses yang menghasilkan berkas, verifikasi keberadaan, ukuran, encoding, dan kemampuan pembacaan ulang.

H.138 dft.UKS()

Fungsi utama: Unrestricted Kohn-Sham DFT. Perintah ini sebaiknya ditempatkan pada tahap notebook yang jelas dan disertai pemeriksaan input. Contoh minimal: `mf=dft.UKS(mol)`. Setelah menjalankan contoh, periksa tipe, nilai, status, atau berkas yang dihasilkan. Jangan menafsirkan keluaran di luar konteks data dan metode.

Catatan penggunaan: tuliskan alasan pemakaian perintah, bukan hanya sintaksnya. Apabila perintah bergantung pada paket tambahan, catat versi paket dan sediakan sel instalasi. Bila perintah dapat gagal, tangkap pengecualian yang spesifik atau periksa nilai balik. Untuk proses yang menghasilkan berkas, verifikasi keberadaan, ukuran, encoding, dan kemampuan pembacaan ulang.

H.139 mf.xc

Fungsi utama: Menetapkan fungsional DFT. Perintah ini sebaiknya ditempatkan pada tahap notebook yang jelas dan disertai pemeriksaan input. Contoh minimal: `mf.xc="b3lyp"`. Setelah menjalankan contoh, periksa tipe, nilai, status, atau berkas yang dihasilkan. Jangan menafsirkan keluaran di luar konteks data dan metode.

Catatan penggunaan: tuliskan alasan pemakaian perintah, bukan hanya sintaksnya. Apabila perintah bergantung pada paket tambahan, catat versi paket dan sediakan sel instalasi. Bila perintah dapat gagal, tangkap pengecualian yang spesifik atau periksa nilai

balik. Untuk proses yang menghasilkan berkas, verifikasi keberadaan, ukuran, encoding, dan kemampuan pembacaan ulang.

H.140 mf.grids.level

Fungsi utama: Tingkat grid numerik DFT. Perintah ini sebaiknya ditempatkan pada tahap notebook yang jelas dan disertai pemeriksaan input. Contoh minimal: `mf.grids.level=4`. Setelah menjalankan contoh, periksa tipe, nilai, status, atau berkas yang dihasilkan. Jangan menafsirkan keluaran di luar konteks data dan metode.

Catatan penggunaan: tuliskan alasan pemakaian perintah, bukan hanya sintaksnya. Apabila perintah bergantung pada paket tambahan, catat versi paket dan sediakan sel instalasi. Bila perintah dapat gagal, tangkap pengecualian yang spesifik atau periksa nilai balik. Untuk proses yang menghasilkan berkas, verifikasi keberadaan, ukuran, encoding, dan kemampuan pembacaan ulang.

H.141 mf.conv_tol

Fungsi utama: Toleransi konvergensi SCF. Perintah ini sebaiknya ditempatkan pada tahap notebook yang jelas dan disertai pemeriksaan input. Contoh minimal: `mf.conv_tol=1e-9`. Setelah menjalankan contoh, periksa tipe, nilai, status, atau berkas yang dihasilkan. Jangan menafsirkan keluaran di luar konteks data dan metode.

Catatan penggunaan: tuliskan alasan pemakaian perintah, bukan hanya sintaksnya. Apabila perintah bergantung pada paket tambahan, catat versi paket dan sediakan sel instalasi. Bila perintah dapat gagal, tangkap pengecualian yang spesifik atau periksa nilai balik. Untuk proses yang menghasilkan berkas, verifikasi keberadaan, ukuran, encoding, dan kemampuan pembacaan ulang.

H.142 mf.max_cycle

Fungsi utama: Maksimum iterasi SCF. Perintah ini sebaiknya ditempatkan pada tahap notebook yang jelas dan disertai pemeriksaan input. Contoh minimal: `mf.max_cycle=200`. Setelah menjalankan contoh, periksa tipe, nilai, status, atau berkas yang dihasilkan. Jangan menafsirkan keluaran di luar konteks data dan metode.

Catatan penggunaan: tuliskan alasan pemakaian perintah, bukan hanya sintaksnya. Apabila perintah bergantung pada paket tambahan, catat versi paket dan sediakan sel instalasi. Bila perintah dapat gagal, tangkap pengecualian yang spesifik atau periksa nilai balik. Untuk proses yang menghasilkan berkas, verifikasi keberadaan, ukuran, encoding, dan kemampuan pembacaan ulang.

H.143 mf.kernel()

Fungsi utama: Menjalankan perhitungan. Perintah ini sebaiknya ditempatkan pada tahap notebook yang jelas dan disertai pemeriksaan input. Contoh minimal: `E=mf.kernel()`.

Setelah menjalankan contoh, periksa tipe, nilai, status, atau berkas yang dihasilkan. Jangan menafsirkan keluaran di luar konteks data dan metode.

Catatan penggunaan: tuliskan alasan pemakaian perintah, bukan hanya sintaksnya. Apabila perintah bergantung pada paket tambahan, catat versi paket dan sediakan sel instalasi. Bila perintah dapat gagal, tangkap pengecualian yang spesifik atau periksa nilai balik. Untuk proses yang menghasilkan berkas, verifikasi keberadaan, ukuran, encoding, dan kemampuan pembacaan ulang.

H.144 mf.converged

Fungsi utama: Status konvergensi. Perintah ini sebaiknya ditempatkan pada tahap notebook yang jelas dan disertai pemeriksaan input. Contoh minimal: `if not mf.converged: raise RuntimeError()`. Setelah menjalankan contoh, periksa tipe, nilai, status, atau berkas yang dihasilkan. Jangan menafsirkan keluaran di luar konteks data dan metode.

Catatan penggunaan: tuliskan alasan pemakaian perintah, bukan hanya sintaksnya. Apabila perintah bergantung pada paket tambahan, catat versi paket dan sediakan sel instalasi. Bila perintah dapat gagal, tangkap pengecualian yang spesifik atau periksa nilai balik. Untuk proses yang menghasilkan berkas, verifikasi keberadaan, ukuran, encoding, dan kemampuan pembacaan ulang.

H.145 mf.newton()

Fungsi utama: Menggunakan Newton solver. Perintah ini sebaiknya ditempatkan pada tahap notebook yang jelas dan disertai pemeriksaan input. Contoh minimal: `mf2=mf.newton(); E=mf2.kernel()`. Setelah menjalankan contoh, periksa tipe, nilai, status, atau berkas yang dihasilkan. Jangan menafsirkan keluaran di luar konteks data dan metode.

Catatan penggunaan: tuliskan alasan pemakaian perintah, bukan hanya sintaksnya. Apabila perintah bergantung pada paket tambahan, catat versi paket dan sediakan sel instalasi. Bila perintah dapat gagal, tangkap pengecualian yang spesifik atau periksa nilai balik. Untuk proses yang menghasilkan berkas, verifikasi keberadaan, ukuran, encoding, dan kemampuan pembacaan ulang.

H.146 mf.make_rdm1()

Fungsi utama: Membuat matriks densitas satu partikel. Perintah ini sebaiknya ditempatkan pada tahap notebook yang jelas dan disertai pemeriksaan input. Contoh minimal: `dm=mf.make_rdm1()`. Setelah menjalankan contoh, periksa tipe, nilai, status, atau berkas yang dihasilkan. Jangan menafsirkan keluaran di luar konteks data dan metode.

Catatan penggunaan: tuliskan alasan pemakaian perintah, bukan hanya sintaksnya. Apabila perintah bergantung pada paket tambahan, catat versi paket dan sediakan sel instalasi. Bila perintah dapat gagal, tangkap pengecualian yang spesifik atau periksa nilai balik. Untuk proses yang menghasilkan berkas, verifikasi keberadaan, ukuran, encoding, dan kemampuan pembacaan ulang.

H.147 mf.mo_energy

Fungsi utama: Energi orbital molekul. Perintah ini sebaiknya ditempatkan pada tahap notebook yang jelas dan disertai pemeriksaan input. Contoh minimal: `eps=mf.mo_energy`. Setelah menjalankan contoh, periksa tipe, nilai, status, atau berkas yang dihasilkan. Jangan menafsirkan keluaran di luar konteks data dan metode.

Catatan penggunaan: tuliskan alasan pemakaian perintah, bukan hanya sintaksnya. Apabila perintah bergantung pada paket tambahan, catat versi paket dan sediakan sel instalasi. Bila perintah dapat gagal, tangkap pengecualian yang spesifik atau periksa nilai balik. Untuk proses yang menghasilkan berkas, verifikasi keberadaan, ukuran, encoding, dan kemampuan pembacaan ulang.

H.148 mf.mo_occ

Fungsi utama: Okupansi orbital. Perintah ini sebaiknya ditempatkan pada tahap notebook yang jelas dan disertai pemeriksaan input. Contoh minimal: `occ=mf.mo_occ`. Setelah menjalankan contoh, periksa tipe, nilai, status, atau berkas yang dihasilkan. Jangan menafsirkan keluaran di luar konteks data dan metode.

Catatan penggunaan: tuliskan alasan pemakaian perintah, bukan hanya sintaksnya. Apabila perintah bergantung pada paket tambahan, catat versi paket dan sediakan sel instalasi. Bila perintah dapat gagal, tangkap pengecualian yang spesifik atau periksa nilai balik. Untuk proses yang menghasilkan berkas, verifikasi keberadaan, ukuran, encoding, dan kemampuan pembacaan ulang.

H.149 mf.mo_coeff

Fungsi utama: Koefisien orbital. Perintah ini sebaiknya ditempatkan pada tahap notebook yang jelas dan disertai pemeriksaan input. Contoh minimal: `C=mf.mo_coeff`. Setelah menjalankan contoh, periksa tipe, nilai, status, atau berkas yang dihasilkan. Jangan menafsirkan keluaran di luar konteks data dan metode.

Catatan penggunaan: tuliskan alasan pemakaian perintah, bukan hanya sintaksnya. Apabila perintah bergantung pada paket tambahan, catat versi paket dan sediakan sel instalasi. Bila perintah dapat gagal, tangkap pengecualian yang spesifik atau periksa nilai balik. Untuk proses yang menghasilkan berkas, verifikasi keberadaan, ukuran, encoding, dan kemampuan pembacaan ulang.

H.150 mf.dip_moment()

Fungsi utama: Menghitung momen dipol. Perintah ini sebaiknya ditempatkan pada tahap notebook yang jelas dan disertai pemeriksaan input. Contoh minimal: `mu=mf.dip_moment(unit="Debye")`. Setelah menjalankan contoh, periksa tipe, nilai, status, atau berkas yang dihasilkan. Jangan menafsirkan keluaran di luar konteks data dan metode.

Catatan penggunaan: tuliskan alasan pemakaian perintah, bukan hanya sintaksnya. Apabila perintah bergantung pada paket tambahan, catat versi paket dan sediakan sel instalasi. Bila perintah dapat gagal, tangkap pengecualian yang spesifik atau periksa nilai balik. Untuk proses yang menghasilkan berkas, verifikasi keberadaan, ukuran, encoding, dan kemampuan pembacaan ulang.

H.151 optimize()

Fungsi utama: Optimasi geometri melalui geomeTRIC. Perintah ini sebaiknya ditempatkan pada tahap notebook yang jelas dan disertai pemeriksaan input. Contoh minimal: `mol_opt=optimize(mf,maxsteps=100)`. Setelah menjalankan contoh, periksa tipe, nilai, status, atau berkas yang dihasilkan. Jangan menafsirkan keluaran di luar konteks data dan metode.

Catatan penggunaan: tuliskan alasan pemakaian perintah, bukan hanya sintaksnya. Apabila perintah bergantung pada paket tambahan, catat versi paket dan sediakan sel instalasi. Bila perintah dapat gagal, tangkap pengecualian yang spesifik atau periksa nilai balik. Untuk proses yang menghasilkan berkas, verifikasi keberadaan, ukuran, encoding, dan kemampuan pembacaan ulang.

H.152 hessian.Hessian()

Fungsi utama: Membangun objek Hessian. Perintah ini sebaiknya ditempatkan pada tahap notebook yang jelas dan disertai pemeriksaan input. Contoh minimal: `hess=mf.Hessian().kernel()`. Setelah menjalankan contoh, periksa tipe, nilai, status, atau berkas yang dihasilkan. Jangan menafsirkan keluaran di luar konteks data dan metode.

Catatan penggunaan: tuliskan alasan pemakaian perintah, bukan hanya sintaksnya. Apabila perintah bergantung pada paket tambahan, catat versi paket dan sediakan sel instalasi. Bila perintah dapat gagal, tangkap pengecualian yang spesifik atau periksa nilai balik. Untuk proses yang menghasilkan berkas, verifikasi keberadaan, ukuran, encoding, dan kemampuan pembacaan ulang.

H.Scientific

H.153 math.sqrt()

Fungsi utama: Akar kuadrat skalar. Perintah ini sebaiknya ditempatkan pada tahap notebook yang jelas dan disertai pemeriksaan input. Contoh minimal: `r=math.sqrt(x)`. Setelah menjalankan contoh, periksa tipe, nilai, status, atau berkas yang dihasilkan. Jangan menafsirkan keluaran di luar konteks data dan metode.

Catatan penggunaan: tuliskan alasan pemakaian perintah, bukan hanya sintaksnya. Apabila perintah bergantung pada paket tambahan, catat versi paket dan sediakan sel instalasi. Bila perintah dapat gagal, tangkap pengecualian yang spesifik atau periksa nilai balik. Untuk proses yang menghasilkan berkas, verifikasi keberadaan, ukuran, encoding, dan kemampuan pembacaan ulang.

H.154 math.isfinite()

Fungsi utama: Memeriksa bilangan hingga skalar. Perintah ini sebaiknya ditempatkan pada tahap notebook yang jelas dan disertai pemeriksaan input. Contoh minimal: `assert math.isfinite(x)`. Setelah menjalankan contoh, periksa tipe, nilai, status, atau berkas yang dihasilkan. Jangan menafsirkan keluaran di luar konteks data dan metode.

Catatan penggunaan: tuliskan alasan pemakaian perintah, bukan hanya sintaksnya. Apabila perintah bergantung pada paket tambahan, catat versi paket dan sediakan sel instalasi. Bila perintah dapat gagal, tangkap pengecualian yang spesifik atau periksa nilai balik. Untuk proses yang menghasilkan berkas, verifikasi keberadaan, ukuran, encoding, dan kemampuan pembacaan ulang.

H.155 statistics.mean()

Fungsi utama: Rata-rata data Python. Perintah ini sebaiknya ditempatkan pada tahap notebook yang jelas dan disertai pemeriksaan input. Contoh minimal: `m=statistics.mean(data)`. Setelah menjalankan contoh, periksa tipe, nilai, status, atau berkas yang dihasilkan. Jangan menafsirkan keluaran di luar konteks data dan metode.

Catatan penggunaan: tuliskan alasan pemakaian perintah, bukan hanya sintaksnya. Apabila perintah bergantung pada paket tambahan, catat versi paket dan sediakan sel instalasi. Bila perintah dapat gagal, tangkap pengecualian yang spesifik atau periksa nilai balik. Untuk proses yang menghasilkan berkas, verifikasi keberadaan, ukuran, encoding, dan kemampuan pembacaan ulang.

H.156 statistics.median()

Fungsi utama: Median data Python. Perintah ini sebaiknya ditempatkan pada tahap notebook yang jelas dan disertai pemeriksaan input. Contoh minimal:

`m=statistics.median(data)`. Setelah menjalankan contoh, periksa tipe, nilai, status, atau berkas yang dihasilkan. Jangan menafsirkan keluaran di luar konteks data dan metode.

Catatan penggunaan: tuliskan alasan pemakaian perintah, bukan hanya sintaksnya. Apabila perintah bergantung pada paket tambahan, catat versi paket dan sediakan sel instalasi. Bila perintah dapat gagal, tangkap pengecualian yang spesifik atau periksa nilai balik. Untuk proses yang menghasilkan berkas, verifikasi keberadaan, ukuran, encoding, dan kemampuan pembacaan ulang.

H.157 `dataclasses.dataclass`

Fungsi utama: Membuat kelas konfigurasi ringkas. Perintah ini sebaiknya ditempatkan pada tahap notebook yang jelas dan disertai pemeriksaan input. Contoh minimal: `@dataclass`

`class Config: method: str`. Setelah menjalankan contoh, periksa tipe, nilai, status, atau berkas yang dihasilkan. Jangan menafsirkan keluaran di luar konteks data dan metode.

Catatan penggunaan: tuliskan alasan pemakaian perintah, bukan hanya sintaksnya. Apabila perintah bergantung pada paket tambahan, catat versi paket dan sediakan sel instalasi. Bila perintah dapat gagal, tangkap pengecualian yang spesifik atau periksa nilai balik. Untuk proses yang menghasilkan berkas, verifikasi keberadaan, ukuran, encoding, dan kemampuan pembacaan ulang.

H.158 `functools.lru_cache`

Fungsi utama: Menyimpan hasil fungsi berulang. Perintah ini sebaiknya ditempatkan pada tahap notebook yang jelas dan disertai pemeriksaan input. Contoh minimal: `@lru_cache`

`def f(x): return x*x`. Setelah menjalankan contoh, periksa tipe, nilai, status, atau berkas yang dihasilkan. Jangan menafsirkan keluaran di luar konteks data dan metode.

Catatan penggunaan: tuliskan alasan pemakaian perintah, bukan hanya sintaksnya. Apabila perintah bergantung pada paket tambahan, catat versi paket dan sediakan sel instalasi. Bila perintah dapat gagal, tangkap pengecualian yang spesifik atau periksa nilai balik. Untuk proses yang menghasilkan berkas, verifikasi keberadaan, ukuran, encoding, dan kemampuan pembacaan ulang.

H.159 `contextlib.redirect_stdout`

Fungsi utama: Mengalihkan output ke berkas. Perintah ini sebaiknya ditempatkan pada tahap notebook yang jelas dan disertai pemeriksaan input. Contoh minimal: `with redirect_stdout(f): proses()`. Setelah menjalankan contoh, periksa tipe, nilai, status, atau berkas yang dihasilkan. Jangan menafsirkan keluaran di luar konteks data dan metode.

Catatan penggunaan: tuliskan alasan pemakaian perintah, bukan hanya sintaksnya. Apabila perintah bergantung pada paket tambahan, catat versi paket dan sediakan sel

instalasi. Bila perintah dapat gagal, tangkap pengecualian yang spesifik atau periksa nilai balik. Untuk proses yang menghasilkan berkas, verifikasi keberadaan, ukuran, encoding, dan kemampuan pembacaan ulang.

H.Testing

H.160 unittest

Fungsi utama: Framework pengujian standar. Perintah ini sebaiknya ditempatkan pada tahap notebook yang jelas dan disertai pemeriksaan input. Contoh minimal: `class TestX(unittest.TestCase): pass`. Setelah menjalankan contoh, periksa tipe, nilai, status, atau berkas yang dihasilkan. Jangan menafsirkan keluaran di luar konteks data dan metode.

Catatan penggunaan: tuliskan alasan pemakaian perintah, bukan hanya sintaksnya. Apabila perintah bergantung pada paket tambahan, catat versi paket dan sediakan sel instalasi. Bila perintah dapat gagal, tangkap pengecualian yang spesifik atau periksa nilai balik. Untuk proses yang menghasilkan berkas, verifikasi keberadaan, ukuran, encoding, dan kemampuan pembacaan ulang.

H.161 pytest.raises

Fungsi utama: Menguji pengecualian yang diharapkan. Perintah ini sebaiknya ditempatkan pada tahap notebook yang jelas dan disertai pemeriksaan input. Contoh minimal: `with pytest.raises(ValueError): f(-1)`. Setelah menjalankan contoh, periksa tipe, nilai, status, atau berkas yang dihasilkan. Jangan menafsirkan keluaran di luar konteks data dan metode.

Catatan penggunaan: tuliskan alasan pemakaian perintah, bukan hanya sintaksnya. Apabila perintah bergantung pada paket tambahan, catat versi paket dan sediakan sel instalasi. Bila perintah dapat gagal, tangkap pengecualian yang spesifik atau periksa nilai balik. Untuk proses yang menghasilkan berkas, verifikasi keberadaan, ukuran, encoding, dan kemampuan pembacaan ulang.

H.162 np.testing.assert_allclose

Fungsi utama: Menguji array numerik. Perintah ini sebaiknya ditempatkan pada tahap notebook yang jelas dan disertai pemeriksaan input. Contoh minimal: `np.testing.assert_allclose(a,b,rtol=1e-7)`. Setelah menjalankan contoh, periksa tipe, nilai, status, atau berkas yang dihasilkan. Jangan menafsirkan keluaran di luar konteks data dan metode.

Catatan penggunaan: tuliskan alasan pemakaian perintah, bukan hanya sintaksnya. Apabila perintah bergantung pada paket tambahan, catat versi paket dan sediakan sel instalasi. Bila perintah dapat gagal, tangkap pengecualian yang spesifik atau periksa nilai

balik. Untuk proses yang menghasilkan berkas, verifikasi keberadaan, ukuran, encoding, dan kemampuan pembacaan ulang.

H.163 tempfile.TemporaryDirectory

Fungsi utama: Membuat direktori sementara. Perintah ini sebaiknya ditempatkan pada tahap notebook yang jelas dan disertai pemeriksaan input. Contoh minimal: `with TemporaryDirectory() as d: pass`. Setelah menjalankan contoh, periksa tipe, nilai, status, atau berkas yang dihasilkan. Jangan menafsirkan keluaran di luar konteks data dan metode.

Catatan penggunaan: tuliskan alasan pemakaian perintah, bukan hanya sintaksnya. Apabila perintah bergantung pada paket tambahan, catat versi paket dan sediakan sel instalasi. Bila perintah dapat gagal, tangkap pengecualian yang spesifik atau periksa nilai balik. Untuk proses yang menghasilkan berkas, verifikasi keberadaan, ukuran, encoding, dan kemampuan pembacaan ulang.

H.Formatting

H.164 f-string

Fungsi utama: Menyisipkan nilai dengan format. Perintah ini sebaiknya ditempatkan pada tahap notebook yang jelas dan disertai pemeriksaan input. Contoh minimal: `print(f"E={E:.8f} Hartree")`. Setelah menjalankan contoh, periksa tipe, nilai, status, atau berkas yang dihasilkan. Jangan menafsirkan keluaran di luar konteks data dan metode.

Catatan penggunaan: tuliskan alasan pemakaian perintah, bukan hanya sintaksnya. Apabila perintah bergantung pada paket tambahan, catat versi paket dan sediakan sel instalasi. Bila perintah dapat gagal, tangkap pengecualian yang spesifik atau periksa nilai balik. Untuk proses yang menghasilkan berkas, verifikasi keberadaan, ukuran, encoding, dan kemampuan pembacaan ulang.

H.165 textwrap.dedent

Fungsi utama: Menghapus indentasi umum teks multiline. Perintah ini sebaiknya ditempatkan pada tahap notebook yang jelas dan disertai pemeriksaan input. Contoh minimal: `kode=textwrap.dedent(kode)`. Setelah menjalankan contoh, periksa tipe, nilai, status, atau berkas yang dihasilkan. Jangan menafsirkan keluaran di luar konteks data dan metode.

Catatan penggunaan: tuliskan alasan pemakaian perintah, bukan hanya sintaksnya. Apabila perintah bergantung pada paket tambahan, catat versi paket dan sediakan sel instalasi. Bila perintah dapat gagal, tangkap pengecualian yang spesifik atau periksa nilai balik. Untuk proses yang menghasilkan berkas, verifikasi keberadaan, ukuran, encoding, dan kemampuan pembacaan ulang.

H.166 csv.DictWriter

Fungsi utama: Menulis CSV berbasis dictionary. Perintah ini sebaiknya ditempatkan pada tahap notebook yang jelas dan disertai pemeriksaan input. Contoh minimal: `w=csv.DictWriter(f,fieldnames=kolom)`. Setelah menjalankan contoh, periksa tipe, nilai, status, atau berkas yang dihasilkan. Jangan menafsirkan keluaran di luar konteks data dan metode.

Catatan penggunaan: tuliskan alasan pemakaian perintah, bukan hanya sintaksnya. Apabila perintah bergantung pada paket tambahan, catat versi paket dan sediakan sel instalasi. Bila perintah dapat gagal, tangkap pengecualian yang spesifik atau periksa nilai balik. Untuk proses yang menghasilkan berkas, verifikasi keberadaan, ukuran, encoding, dan kemampuan pembacaan ulang.

H.167 datetime.now()

Fungsi utama: Merekam waktu lokal. Perintah ini sebaiknya ditempatkan pada tahap notebook yang jelas dan disertai pemeriksaan input. Contoh minimal: `waktu=datetime.now().isoformat()`. Setelah menjalankan contoh, periksa tipe, nilai, status, atau berkas yang dihasilkan. Jangan menafsirkan keluaran di luar konteks data dan metode.

Catatan penggunaan: tuliskan alasan pemakaian perintah, bukan hanya sintaksnya. Apabila perintah bergantung pada paket tambahan, catat versi paket dan sediakan sel instalasi. Bila perintah dapat gagal, tangkap pengecualian yang spesifik atau periksa nilai balik. Untuk proses yang menghasilkan berkas, verifikasi keberadaan, ukuran, encoding, dan kemampuan pembacaan ulang.

H.168 time.time()

Fungsi utama: Timestamp Unix. Perintah ini sebaiknya ditempatkan pada tahap notebook yang jelas dan disertai pemeriksaan input. Contoh minimal: `t=time.time()`. Setelah menjalankan contoh, periksa tipe, nilai, status, atau berkas yang dihasilkan. Jangan menafsirkan keluaran di luar konteks data dan metode.

Catatan penggunaan: tuliskan alasan pemakaian perintah, bukan hanya sintaksnya. Apabila perintah bergantung pada paket tambahan, catat versi paket dan sediakan sel instalasi. Bila perintah dapat gagal, tangkap pengecualian yang spesifik atau periksa nilai balik. Untuk proses yang menghasilkan berkas, verifikasi keberadaan, ukuran, encoding, dan kemampuan pembacaan ulang.

H.System

H.169 os.getcwd()

Fungsi utama: Direktori kerja. Perintah ini sebaiknya ditempatkan pada tahap notebook yang jelas dan disertai pemeriksaan input. Contoh minimal: `print(os.getcwd())`. Setelah menjalankan contoh, periksa tipe, nilai, status, atau berkas yang dihasilkan. Jangan menafsirkan keluaran di luar konteks data dan metode.

Catatan penggunaan: tuliskan alasan pemakaian perintah, bukan hanya sintaksnya. Apabila perintah bergantung pada paket tambahan, catat versi paket dan sediakan sel instalasi. Bila perintah dapat gagal, tangkap pengecualian yang spesifik atau periksa nilai balik. Untuk proses yang menghasilkan berkas, verifikasi keberadaan, ukuran, encoding, dan kemampuan pembacaan ulang.

H.170 os.cpu_count()

Fungsi utama: Jumlah CPU logis. Perintah ini sebaiknya ditempatkan pada tahap notebook yang jelas dan disertai pemeriksaan input. Contoh minimal: `print(os.cpu_count())`. Setelah menjalankan contoh, periksa tipe, nilai, status, atau berkas yang dihasilkan. Jangan menafsirkan keluaran di luar konteks data dan metode.

Catatan penggunaan: tuliskan alasan pemakaian perintah, bukan hanya sintaksnya. Apabila perintah bergantung pada paket tambahan, catat versi paket dan sediakan sel instalasi. Bila perintah dapat gagal, tangkap pengecualian yang spesifik atau periksa nilai balik. Untuk proses yang menghasilkan berkas, verifikasi keberadaan, ukuran, encoding, dan kemampuan pembacaan ulang.

H.171 os.environ

Fungsi utama: Variabel lingkungan. Perintah ini sebaiknya ditempatkan pada tahap notebook yang jelas dan disertai pemeriksaan input. Contoh minimal: `nilai=os.environ.get("KEY")`. Setelah menjalankan contoh, periksa tipe, nilai, status, atau berkas yang dihasilkan. Jangan menafsirkan keluaran di luar konteks data dan metode.

Catatan penggunaan: tuliskan alasan pemakaian perintah, bukan hanya sintaksnya. Apabila perintah bergantung pada paket tambahan, catat versi paket dan sediakan sel instalasi. Bila perintah dapat gagal, tangkap pengecualian yang spesifik atau periksa nilai balik. Untuk proses yang menghasilkan berkas, verifikasi keberadaan, ukuran, encoding, dan kemampuan pembacaan ulang.

H.172 platform.platform()

Fungsi utama: Identitas platform. Perintah ini sebaiknya ditempatkan pada tahap notebook yang jelas dan disertai pemeriksaan input. Contoh minimal:

`print(platform.platform())`). Setelah menjalankan contoh, periksa tipe, nilai, status, atau berkas yang dihasilkan. Jangan menafsirkan keluaran di luar konteks data dan metode.

Catatan penggunaan: tuliskan alasan pemakaian perintah, bukan hanya sintaksnya. Apabila perintah bergantung pada paket tambahan, catat versi paket dan sediakan sel instalasi. Bila perintah dapat gagal, tangkap pengecualian yang spesifik atau periksa nilai balik. Untuk proses yang menghasilkan berkas, verifikasi keberadaan, ukuran, encoding, dan kemampuan pembacaan ulang.

H.173 sys.version

Fungsi utama: Versi Python lengkap. Perintah ini sebaiknya ditempatkan pada tahap notebook yang jelas dan disertai pemeriksaan input. Contoh minimal: `print(sys.version)`. Setelah menjalankan contoh, periksa tipe, nilai, status, atau berkas yang dihasilkan. Jangan menafsirkan keluaran di luar konteks data dan metode.

Catatan penggunaan: tuliskan alasan pemakaian perintah, bukan hanya sintaksnya. Apabila perintah bergantung pada paket tambahan, catat versi paket dan sediakan sel instalasi. Bila perintah dapat gagal, tangkap pengecualian yang spesifik atau periksa nilai balik. Untuk proses yang menghasilkan berkas, verifikasi keberadaan, ukuran, encoding, dan kemampuan pembacaan ulang.

H.174 sys.executable

Fungsi utama: Lokasi interpreter. Perintah ini sebaiknya ditempatkan pada tahap notebook yang jelas dan disertai pemeriksaan input. Contoh minimal: `print(sys.executable)`. Setelah menjalankan contoh, periksa tipe, nilai, status, atau berkas yang dihasilkan. Jangan menafsirkan keluaran di luar konteks data dan metode.

Catatan penggunaan: tuliskan alasan pemakaian perintah, bukan hanya sintaksnya. Apabila perintah bergantung pada paket tambahan, catat versi paket dan sediakan sel instalasi. Bila perintah dapat gagal, tangkap pengecualian yang spesifik atau periksa nilai balik. Untuk proses yang menghasilkan berkas, verifikasi keberadaan, ukuran, encoding, dan kemampuan pembacaan ulang.

H.175 gc.collect()

Fungsi utama: Meminta garbage collection. Perintah ini sebaiknya ditempatkan pada tahap notebook yang jelas dan disertai pemeriksaan input. Contoh minimal: `gc.collect()`. Setelah menjalankan contoh, periksa tipe, nilai, status, atau berkas yang dihasilkan. Jangan menafsirkan keluaran di luar konteks data dan metode.

Catatan penggunaan: tuliskan alasan pemakaian perintah, bukan hanya sintaksnya. Apabila perintah bergantung pada paket tambahan, catat versi paket dan sediakan sel instalasi. Bila perintah dapat gagal, tangkap pengecualian yang spesifik atau periksa nilai

balik. Untuk proses yang menghasilkan berkas, verifikasi keberadaan, ukuran, encoding, dan kemampuan pembacaan ulang.

H.176 warnings.warn()

Fungsi utama: Memberi peringatan. Perintah ini sebaiknya ditempatkan pada tahap notebook yang jelas dan disertai pemeriksaan input. Contoh minimal: `warnings.warn("basis minimal")`. Setelah menjalankan contoh, periksa tipe, nilai, status, atau berkas yang dihasilkan. Jangan menafsirkan keluaran di luar konteks data dan metode.

Catatan penggunaan: tuliskan alasan pemakaian perintah, bukan hanya sintaksnya. Apabila perintah bergantung pada paket tambahan, catat versi paket dan sediakan sel instalasi. Bila perintah dapat gagal, tangkap pengecualian yang spesifik atau periksa nilai balik. Untuk proses yang menghasilkan berkas, verifikasi keberadaan, ukuran, encoding, dan kemampuan pembacaan ulang.

H.177 logging.info()

Fungsi utama: Mencatat informasi. Perintah ini sebaiknya ditempatkan pada tahap notebook yang jelas dan disertai pemeriksaan input. Contoh minimal: `logging.info("selesai")`. Setelah menjalankan contoh, periksa tipe, nilai, status, atau berkas yang dihasilkan. Jangan menafsirkan keluaran di luar konteks data dan metode.

Catatan penggunaan: tuliskan alasan pemakaian perintah, bukan hanya sintaksnya. Apabila perintah bergantung pada paket tambahan, catat versi paket dan sediakan sel instalasi. Bila perintah dapat gagal, tangkap pengecualian yang spesifik atau periksa nilai balik. Untuk proses yang menghasilkan berkas, verifikasi keberadaan, ukuran, encoding, dan kemampuan pembacaan ulang.

H.178 logging.exception()

Fungsi utama: Mencatat traceback dalam except. Perintah ini sebaiknya ditempatkan pada tahap notebook yang jelas dan disertai pemeriksaan input. Contoh minimal: `logging.exception("gagal")`. Setelah menjalankan contoh, periksa tipe, nilai, status, atau berkas yang dihasilkan. Jangan menafsirkan keluaran di luar konteks data dan metode.

Catatan penggunaan: tuliskan alasan pemakaian perintah, bukan hanya sintaksnya. Apabila perintah bergantung pada paket tambahan, catat versi paket dan sediakan sel instalasi. Bila perintah dapat gagal, tangkap pengecualian yang spesifik atau periksa nilai balik. Untuk proses yang menghasilkan berkas, verifikasi keberadaan, ukuran, encoding, dan kemampuan pembacaan ulang.

H.179 subprocess.run()

Fungsi utama: Menjalankan perintah dengan kontrol. Perintah ini sebaiknya ditempatkan pada tahap notebook yang jelas dan disertai pemeriksaan input. Contoh minimal: `r=subprocess.run(cmd,check=False)`. Setelah menjalankan contoh, periksa tipe, nilai, status, atau berkas yang dihasilkan. Jangan menafsirkan keluaran di luar konteks data dan metode.

Catatan penggunaan: tuliskan alasan pemakaian perintah, bukan hanya sintaksnya. Apabila perintah bergantung pada paket tambahan, catat versi paket dan sediakan sel instalasi. Bila perintah dapat gagal, tangkap pengecualian yang spesifik atau periksa nilai balik. Untuk proses yang menghasilkan berkas, verifikasi keberadaan, ukuran, encoding, dan kemampuan pembacaan ulang.

H.180 shlex.split()

Fungsi utama: Memecah perintah secara aman. Perintah ini sebaiknya ditempatkan pada tahap notebook yang jelas dan disertai pemeriksaan input. Contoh minimal: `args=shlex.split(teks)`. Setelah menjalankan contoh, periksa tipe, nilai, status, atau berkas yang dihasilkan. Jangan menafsirkan keluaran di luar konteks data dan metode.

Catatan penggunaan: tuliskan alasan pemakaian perintah, bukan hanya sintaksnya. Apabila perintah bergantung pada paket tambahan, catat versi paket dan sediakan sel instalasi. Bila perintah dapat gagal, tangkap pengecualian yang spesifik atau periksa nilai balik. Untuk proses yang menghasilkan berkas, verifikasi keberadaan, ukuran, encoding, dan kemampuan pembacaan ulang.

H.Data integrity

H.181 Path.stat().st_size

Fungsi utama: Ukuran berkas. Perintah ini sebaiknya ditempatkan pada tahap notebook yang jelas dan disertai pemeriksaan input. Contoh minimal: `size=p.stat().st_size`. Setelah menjalankan contoh, periksa tipe, nilai, status, atau berkas yang dihasilkan. Jangan menafsirkan keluaran di luar konteks data dan metode.

Catatan penggunaan: tuliskan alasan pemakaian perintah, bukan hanya sintaksnya. Apabila perintah bergantung pada paket tambahan, catat versi paket dan sediakan sel instalasi. Bila perintah dapat gagal, tangkap pengecualian yang spesifik atau periksa nilai balik. Untuk proses yang menghasilkan berkas, verifikasi keberadaan, ukuran, encoding, dan kemampuan pembacaan ulang.

H.182 hashlib.md5()

Fungsi utama: Checksum cepat non-keamanan. Perintah ini sebaiknya ditempatkan pada tahap notebook yang jelas dan disertai pemeriksaan input. Contoh minimal: `md5=hashlib.md5(data).hexdigest()`. Setelah menjalankan contoh, periksa tipe, nilai, status, atau berkas yang dihasilkan. Jangan menafsirkan keluaran di luar konteks data dan metode.

Catatan penggunaan: tuliskan alasan pemakaian perintah, bukan hanya sintaksnya. Apabila perintah bergantung pada paket tambahan, catat versi paket dan sediakan sel instalasi. Bila perintah dapat gagal, tangkap pengecualian yang spesifik atau periksa nilai balik. Untuk proses yang menghasilkan berkas, verifikasi keberadaan, ukuran, encoding, dan kemampuan pembacaan ulang.

H.183 hashlib.sha256()

Fungsi utama: Checksum integritas kuat. Perintah ini sebaiknya ditempatkan pada tahap notebook yang jelas dan disertai pemeriksaan input. Contoh minimal: `sha=hashlib.sha256(data).hexdigest()`. Setelah menjalankan contoh, periksa tipe, nilai, status, atau berkas yang dihasilkan. Jangan menafsirkan keluaran di luar konteks data dan metode.

Catatan penggunaan: tuliskan alasan pemakaian perintah, bukan hanya sintaksnya. Apabila perintah bergantung pada paket tambahan, catat versi paket dan sediakan sel instalasi. Bila perintah dapat gagal, tangkap pengecualian yang spesifik atau periksa nilai balik. Untuk proses yang menghasilkan berkas, verifikasi keberadaan, ukuran, encoding, dan kemampuan pembacaan ulang.

H.184 jsonschema.validate

Fungsi utama: Memvalidasi struktur JSON. Perintah ini sebaiknya ditempatkan pada tahap notebook yang jelas dan disertai pemeriksaan input. Contoh minimal: `validate(instance=data,schema=schema)`. Setelah menjalankan contoh, periksa tipe, nilai, status, atau berkas yang dihasilkan. Jangan menafsirkan keluaran di luar konteks data dan metode.

Catatan penggunaan: tuliskan alasan pemakaian perintah, bukan hanya sintaksnya. Apabila perintah bergantung pada paket tambahan, catat versi paket dan sediakan sel instalasi. Bila perintah dapat gagal, tangkap pengecualian yang spesifik atau periksa nilai balik. Untuk proses yang menghasilkan berkas, verifikasi keberadaan, ukuran, encoding, dan kemampuan pembacaan ulang.

H.185 pandas.testing.assert_frame_equal

Fungsi utama: Menguji kesamaan DataFrame. Perintah ini sebaiknya ditempatkan pada tahap notebook yang jelas dan disertai pemeriksaan input. Contoh minimal: `assert_frame_equal(a,b)`. Setelah menjalankan contoh, periksa tipe, nilai, status, atau berkas yang dihasilkan. Jangan menafsirkan keluaran di luar konteks data dan metode.

Catatan penggunaan: tuliskan alasan pemakaian perintah, bukan hanya sintaksnya. Apabila perintah bergantung pada paket tambahan, catat versi paket dan sediakan sel instalasi. Bila perintah dapat gagal, tangkap pengecualian yang spesifik atau periksa nilai balik. Untuk proses yang menghasilkan berkas, verifikasi keberadaan, ukuran, encoding, dan kemampuan pembacaan ulang.

H.186 Path.rename()

Fungsi utama: Mengganti nama berkas. Perintah ini sebaiknya ditempatkan pada tahap notebook yang jelas dan disertai pemeriksaan input. Contoh minimal: `p.rename("baru.txt")`. Setelah menjalankan contoh, periksa tipe, nilai, status, atau berkas yang dihasilkan. Jangan menafsirkan keluaran di luar konteks data dan metode.

Catatan penggunaan: tuliskan alasan pemakaian perintah, bukan hanya sintaksnya. Apabila perintah bergantung pada paket tambahan, catat versi paket dan sediakan sel instalasi. Bila perintah dapat gagal, tangkap pengecualian yang spesifik atau periksa nilai balik. Untuk proses yang menghasilkan berkas, verifikasi keberadaan, ukuran, encoding, dan kemampuan pembacaan ulang.

H.187 Path.unlink()

Fungsi utama: Menghapus berkas. Perintah ini sebaiknya ditempatkan pada tahap notebook yang jelas dan disertai pemeriksaan input. Contoh minimal: `p.unlink(missing_ok=True)`. Setelah menjalankan contoh, periksa tipe, nilai, status, atau berkas yang dihasilkan. Jangan menafsirkan keluaran di luar konteks data dan metode.

Catatan penggunaan: tuliskan alasan pemakaian perintah, bukan hanya sintaksnya. Apabila perintah bergantung pada paket tambahan, catat versi paket dan sediakan sel instalasi. Bila perintah dapat gagal, tangkap pengecualian yang spesifik atau periksa nilai balik. Untuk proses yang menghasilkan berkas, verifikasi keberadaan, ukuran, encoding, dan kemampuan pembacaan ulang.

H.Packaging

H.188 requirements.txt

Fungsi utama: Mencatat versi paket. Perintah ini sebaiknya ditempatkan pada tahap notebook yang jelas dan disertai pemeriksaan input. Contoh minimal: `numpy==2.x`.

Setelah menjalankan contoh, periksa tipe, nilai, status, atau berkas yang dihasilkan. Jangan menafsirkan keluaran di luar konteks data dan metode.

Catatan penggunaan: tuliskan alasan pemakaian perintah, bukan hanya sintaksnya. Apabila perintah bergantung pada paket tambahan, catat versi paket dan sediakan sel instalasi. Bila perintah dapat gagal, tangkap pengecualian yang spesifik atau periksa nilai balik. Untuk proses yang menghasilkan berkas, verifikasi keberadaan, ukuran, encoding, dan kemampuan pembacaan ulang.

H.189 pip freeze

Fungsi utama: Mencetak semua paket. Perintah ini sebaiknya ditempatkan pada tahap notebook yang jelas dan disertai pemeriksaan input. Contoh minimal: `!pip freeze > freeze.txt`. Setelah menjalankan contoh, periksa tipe, nilai, status, atau berkas yang dihasilkan. Jangan menafsirkan keluaran di luar konteks data dan metode.

Catatan penggunaan: tuliskan alasan pemakaian perintah, bukan hanya sintaksnya. Apabila perintah bergantung pada paket tambahan, catat versi paket dan sediakan sel instalasi. Bila perintah dapat gagal, tangkap pengecualian yang spesifik atau periksa nilai balik. Untuk proses yang menghasilkan berkas, verifikasi keberadaan, ukuran, encoding, dan kemampuan pembacaan ulang.

H.190 pip show

Fungsi utama: Informasi paket. Perintah ini sebaiknya ditempatkan pada tahap notebook yang jelas dan disertai pemeriksaan input. Contoh minimal: `!pip show pyscf`. Setelah menjalankan contoh, periksa tipe, nilai, status, atau berkas yang dihasilkan. Jangan menafsirkan keluaran di luar konteks data dan metode.

Catatan penggunaan: tuliskan alasan pemakaian perintah, bukan hanya sintaksnya. Apabila perintah bergantung pada paket tambahan, catat versi paket dan sediakan sel instalasi. Bila perintah dapat gagal, tangkap pengecualian yang spesifik atau periksa nilai balik. Untuk proses yang menghasilkan berkas, verifikasi keberadaan, ukuran, encoding, dan kemampuan pembacaan ulang.

H.191 importlib.util.find_spec

Fungsi utama: Mendeteksi modul tanpa impor. Perintah ini sebaiknya ditempatkan pada tahap notebook yang jelas dan disertai pemeriksaan input. Contoh minimal: `ok=find_spec("numpy") is not None`. Setelah menjalankan contoh, periksa tipe, nilai, status, atau berkas yang dihasilkan. Jangan menafsirkan keluaran di luar konteks data dan metode.

Catatan penggunaan: tuliskan alasan pemakaian perintah, bukan hanya sintaksnya. Apabila perintah bergantung pada paket tambahan, catat versi paket dan sediakan sel instalasi. Bila perintah dapat gagal, tangkap pengecualian yang spesifik atau periksa nilai

balik. Untuk proses yang menghasilkan berkas, verifikasi keberadaan, ukuran, encoding, dan kemampuan pembacaan ulang.

H.Notebook

H.192 display()

Fungsi utama: Menampilkan objek kaya. Perintah ini sebaiknya ditempatkan pada tahap notebook yang jelas dan disertai pemeriksaan input. Contoh minimal: `display(df.head())`. Setelah menjalankan contoh, periksa tipe, nilai, status, atau berkas yang dihasilkan. Jangan menafsirkan keluaran di luar konteks data dan metode.

Catatan penggunaan: tuliskan alasan pemakaian perintah, bukan hanya sintaksnya. Apabila perintah bergantung pada paket tambahan, catat versi paket dan sediakan sel instalasi. Bila perintah dapat gagal, tangkap pengecualian yang spesifik atau periksa nilai balik. Untuk proses yang menghasilkan berkas, verifikasi keberadaan, ukuran, encoding, dan kemampuan pembacaan ulang.

H.193 Markdown()

Fungsi utama: Menampilkan Markdown dinamis. Perintah ini sebaiknya ditempatkan pada tahap notebook yang jelas dan disertai pemeriksaan input. Contoh minimal: `display(Markdown("## Hasil"))`. Setelah menjalankan contoh, periksa tipe, nilai, status, atau berkas yang dihasilkan. Jangan menafsirkan keluaran di luar konteks data dan metode.

Catatan penggunaan: tuliskan alasan pemakaian perintah, bukan hanya sintaksnya. Apabila perintah bergantung pada paket tambahan, catat versi paket dan sediakan sel instalasi. Bila perintah dapat gagal, tangkap pengecualian yang spesifik atau periksa nilai balik. Untuk proses yang menghasilkan berkas, verifikasi keberadaan, ukuran, encoding, dan kemampuan pembacaan ulang.

H.194 clear_output()

Fungsi utama: Membersihkan output sel. Perintah ini sebaiknya ditempatkan pada tahap notebook yang jelas dan disertai pemeriksaan input. Contoh minimal: `clear_output(wait=True)`. Setelah menjalankan contoh, periksa tipe, nilai, status, atau berkas yang dihasilkan. Jangan menafsirkan keluaran di luar konteks data dan metode.

Catatan penggunaan: tuliskan alasan pemakaian perintah, bukan hanya sintaksnya. Apabila perintah bergantung pada paket tambahan, catat versi paket dan sediakan sel instalasi. Bila perintah dapat gagal, tangkap pengecualian yang spesifik atau periksa nilai balik. Untuk proses yang menghasilkan berkas, verifikasi keberadaan, ukuran, encoding, dan kemampuan pembacaan ulang.

H.195 %%time

Fungsi utama: Mengukur waktu satu sel. Perintah ini sebaiknya ditempatkan pada tahap notebook yang jelas dan disertai pemeriksaan input. Contoh minimal: `%%time proses()`. Setelah menjalankan contoh, periksa tipe, nilai, status, atau berkas yang dihasilkan. Jangan menafsirkan keluaran di luar konteks data dan metode.

Catatan penggunaan: tuliskan alasan pemakaian perintah, bukan hanya sintaksnya. Apabila perintah bergantung pada paket tambahan, catat versi paket dan sediakan sel instalasi. Bila perintah dapat gagal, tangkap pengecualian yang spesifik atau periksa nilai balik. Untuk proses yang menghasilkan berkas, verifikasi keberadaan, ukuran, encoding, dan kemampuan pembacaan ulang.

H.196 %timeit

Fungsi utama: Benchmark ekspresi berulang. Perintah ini sebaiknya ditempatkan pada tahap notebook yang jelas dan disertai pemeriksaan input. Contoh minimal: `%timeit fungsi()`. Setelah menjalankan contoh, periksa tipe, nilai, status, atau berkas yang dihasilkan. Jangan menafsirkan keluaran di luar konteks data dan metode.

Catatan penggunaan: tuliskan alasan pemakaian perintah, bukan hanya sintaksnya. Apabila perintah bergantung pada paket tambahan, catat versi paket dan sediakan sel instalasi. Bila perintah dapat gagal, tangkap pengecualian yang spesifik atau periksa nilai balik. Untuk proses yang menghasilkan berkas, verifikasi keberadaan, ukuran, encoding, dan kemampuan pembacaan ulang.

H.Pola Notebook

H.197 checkpoint_01

Fungsi utama: Pola pencatatan checkpoint tahap proses. Perintah ini sebaiknya ditempatkan pada tahap notebook yang jelas dan disertai pemeriksaan input. Contoh minimal: `print("Checkpoint 01: selesai")`. Setelah menjalankan contoh, periksa tipe, nilai, status, atau berkas yang dihasilkan. Jangan menafsirkan keluaran di luar konteks data dan metode.

Catatan penggunaan: tuliskan alasan pemakaian perintah, bukan hanya sintaksnya. Apabila perintah bergantung pada paket tambahan, catat versi paket dan sediakan sel instalasi. Bila perintah dapat gagal, tangkap pengecualian yang spesifik atau periksa nilai balik. Untuk proses yang menghasilkan berkas, verifikasi keberadaan, ukuran, encoding, dan kemampuan pembacaan ulang.

H.198 checkpoint_02

Fungsi utama: Pola pencatatan checkpoint tahap proses. Perintah ini sebaiknya ditempatkan pada tahap notebook yang jelas dan disertai pemeriksaan input. Contoh minimal: `print("Checkpoint 02: selesai")`. Setelah menjalankan contoh, periksa tipe, nilai, status, atau berkas yang dihasilkan. Jangan menafsirkan keluaran di luar konteks data dan metode.

Catatan penggunaan: tuliskan alasan pemakaian perintah, bukan hanya sintaksnya. Apabila perintah bergantung pada paket tambahan, catat versi paket dan sediakan sel instalasi. Bila perintah dapat gagal, tangkap pengecualian yang spesifik atau periksa nilai balik. Untuk proses yang menghasilkan berkas, verifikasi keberadaan, ukuran, encoding, dan kemampuan pembacaan ulang.

H.199 checkpoint_03

Fungsi utama: Pola pencatatan checkpoint tahap proses. Perintah ini sebaiknya ditempatkan pada tahap notebook yang jelas dan disertai pemeriksaan input. Contoh minimal: `print("Checkpoint 03: selesai")`. Setelah menjalankan contoh, periksa tipe, nilai, status, atau berkas yang dihasilkan. Jangan menafsirkan keluaran di luar konteks data dan metode.

Catatan penggunaan: tuliskan alasan pemakaian perintah, bukan hanya sintaksnya. Apabila perintah bergantung pada paket tambahan, catat versi paket dan sediakan sel instalasi. Bila perintah dapat gagal, tangkap pengecualian yang spesifik atau periksa nilai balik. Untuk proses yang menghasilkan berkas, verifikasi keberadaan, ukuran, encoding, dan kemampuan pembacaan ulang.

H.200 checkpoint_04

Fungsi utama: Pola pencatatan checkpoint tahap proses. Perintah ini sebaiknya ditempatkan pada tahap notebook yang jelas dan disertai pemeriksaan input. Contoh minimal: `print("Checkpoint 04: selesai")`. Setelah menjalankan contoh, periksa tipe, nilai, status, atau berkas yang dihasilkan. Jangan menafsirkan keluaran di luar konteks data dan metode.

Catatan penggunaan: tuliskan alasan pemakaian perintah, bukan hanya sintaksnya. Apabila perintah bergantung pada paket tambahan, catat versi paket dan sediakan sel instalasi. Bila perintah dapat gagal, tangkap pengecualian yang spesifik atau periksa nilai balik. Untuk proses yang menghasilkan berkas, verifikasi keberadaan, ukuran, encoding, dan kemampuan pembacaan ulang.

LAMPIRAN I Tutorial Google Colab untuk Kimia Komputasi Dasar

Saya telah menyiapkan notebook Google Colab yang berisi seluruh latihan pada tutorial ini:

[Unduh Notebook Tutorial Kimia Komputasi PySCF](#)

File tersebut dapat langsung dibuka melalui **Google Colab** → **File** → **Upload notebook**.

1. Mengenal Google Colab

Google Colab adalah lingkungan Jupyter Notebook berbasis web untuk menulis dan menjalankan Python tanpa instalasi Python di komputer. Notebook tersimpan di Google Drive dan dapat dibagikan seperti dokumen Google lainnya. Kode dijalankan pada mesin virtual, sehingga pustaka tambahan perlu dipasang kembali ketika runtime baru dibuat. ([Google Research](#))

Untuk perhitungan molekul kecil seperti H_2 dan H_2O , gunakan runtime CPU standar. GPU belum diperlukan.

2. Membuat notebook baru

1. Buka Google Colab.
2. Masuk menggunakan akun Google.
3. Pilih **File** → **New notebook**.
4. Ubah nama notebook, misalnya:

`Kimia_Komputasi_Dasar.ipynb`

5. Jalankan sebuah sel dengan menekan tombol  di sebelah kiri sel atau menggunakan:

`Ctrl + Enter`

Notebook menggabungkan sel kode, teks, persamaan, tabel, dan gambar dalam satu dokumen. ([Google Colab](#))

Bagian A — Instalasi PySCF

Pada tutorial ini digunakan **PySCF**, yaitu pustaka Python untuk perhitungan struktur elektronik seperti Hartree–Fock, DFT, MP2, coupled cluster, dan beberapa metode kimia kuantum lainnya. Pola umum penggunaannya adalah membuat objek molekul, memilih metode, kemudian menjalankan metode tersebut dengan `kernel()`. ([pyscf.org](#))

3. Memasang PySCF

Buat sel kode pertama, kemudian masukkan:

```
%pip install --prefer-binary pyscf
```

Jalankan sel tersebut.

Opsi `--prefer-binary` meminta `pip` memakai paket biner yang sudah dikompilasi sehingga instalasi biasanya lebih sederhana daripada membangun PySCF dari kode sumber. Cara ini direkomendasikan dalam dokumentasi instalasi PySCF. (pyscf.org)

Untuk tutorial lengkap, termasuk optimasi geometri dan visualisasi molekul, gunakan:

```
%pip install --prefer-binary pyscf geometric py3Dmol
```

Setelah instalasi selesai, periksa versinya:

```
import pyscf  
  
print("Versi PySCF:", pyscf.__version__)
```

Apabila nomor versi tampil tanpa pesan kesalahan, instalasi berhasil.

Bagian B — Perhitungan Molekul H₂

Kita mulai dari molekul paling sederhana, yaitu hidrogen:

```
[  
  \mathrm{H_2}  
]
```

Perhitungan menggunakan:

Komponen	Pilihan
Molekul	H ₂
Muatan	0
Keadaan spin	singlet
Metode	Restricted Hartree–Fock
Basis set	STO-3G
Jarak H–H	0,74 Å

4. Menjalankan perhitungan H₂

Masukkan kode berikut ke sel baru:

```
from pyscf import gto, scf

# Membuat objek molekul H2
mol_h2 = gto.M(
    atom="""
    H  0.000000  0.000000  0.000000
    H  0.000000  0.000000  0.740000
    """,
    unit="Angstrom",
    basis="sto-3g",
    charge=0,
    spin=0,
    verbose=4
)

# Menentukan metode Restricted Hartree-Fock
hf_h2 = scf.RHF(mol_h2)

# Menjalankan perhitungan SCF
energi_h2 = hf_h2.kernel()

print("\nHasil perhitungan")
print("Konvergen      :", hf_h2.converged)
print(f"Energi total : {energi_h2:.10f} Hartree")
```

Pembuatan molekul dengan modul `gto` dan penggunaan metode SCF seperti RHF merupakan pola dasar perhitungan PySCF. (pyscf.org)

Contoh lain:

```
!pip -q install pyscf geometric
import numpy as np
from pyscf import gto, dft
from pyscf.geomopt.geometric_solver import optimize

# Struktur awal H2
mol_h2 = gto.M(
    atom="""
    H  0.000000  0.000000  0.000000
    H  0.000000  0.000000  0.740000
    """,
    unit="Angstrom",
    basis="def2-tzvp",
    charge=0,
    spin=0,
    verbose=4
)

# DFT B3LYP
ks_h2 = dft.RKS(mol_h2)
ks_h2.xc = "b3lyp"
ks_h2.grids.level = 4
ks_h2.conv_tol = 1.0e-10
ks_h2.max_cycle = 200
```

```
# Optimasi geometri
mol_h2_opt = optimize(
    ks_h2,
    maxsteps=100
)

# Energi akhir pada struktur optimum
ks_h2_final = dft.RKS(mol_h2_opt)
ks_h2_final.xc = "b3lyp"
ks_h2_final.grids.level = 4
ks_h2_final.conv_tol = 1.0e-10
ks_h2_final.max_cycle = 200

energi_h2 = ks_h2_final.kernel()

# Panjang ikatan optimum
koordinat = mol_h2_opt.atom_coords(
    unit="Angstrom"
)

panjang_hh = np.linalg.norm(
    koordinat[1] - koordinat[0]
)

print("\n" + "=" * 55)
print("HASIL OPTIMASI MOLEKUL H2")
print("=" * 55)
print("Metode           : DFT B3LYP")
print("Basis set         : def2-TZVP")
print("Konvergen         :", ks_h2_final.converged)
print(f"Panjang H-H       : {panjang_hh:.8f} Å")
print(f"Energi total      : {energi_h2:.12f} Hartree")
```

Contoh skrip lebih panjang:

```
# =====
# DFT B3LYP UNTUK OPTIMASI GEOMETRI DAN ENERGI TOTAL MOLEKUL H2
#
# Optimasi geometri : B3LYP/def2-TZVP
# Energi akhir      : B3LYP/def2-QZVP
# =====

# Jalankan instalasi ini sekali pada setiap runtime Google Colab
!pip -q install pyscf geometric

import numpy as np

from pyscf import gto, dft
from pyscf.geomopt.geometric_solver import optimize
# =====
# 1. MEMBUAT STRUKTUR AWAL MOLEKUL H2
# =====

mol_h2 = gto.M(
    atom="""
    H  0.000000  0.000000  0.000000
    H  0.000000  0.000000  0.740000
    """,
```

```
unit="Angstrom",

# Basis set untuk optimasi geometri
basis="def2-tzvp",

charge=0,

# spin = jumlah elektron alfa - jumlah elektron beta
# H2 netral keadaan dasar adalah singlet
spin=0,

symmetry=True,
verbose=4
)

# =====
# 2. MENENTUKAN METODE DFT B3LYP
# =====

dft_opt = dft.RKS(mol_h2)

# Fungsional pertukaran-korelasi
dft_opt.xc = "b3lyp"

# Grid integrasi numerik DFT
# level 4 cukup baik untuk optimasi molekul kecil
dft_opt.grids.level = 4

# Kriteria konvergensi SCF
dft_opt.conv_tol = 1.0e-10

# Jumlah maksimum iterasi SCF
dft_opt.max_cycle = 200

# =====
# 3. PERHITUNGAN ENERGI STRUKTUR AWAL
# =====

energi_awal = dft_opt.kernel()

if not dft_opt.converged:
    raise RuntimeError(
        "Perhitungan SCF struktur awal tidak konvergen."
    )

print("\n" + "=" * 65)
print("HASIL STRUKTUR AWAL")
print("=" * 65)
print("Metode          : B3LYP")
print("Basis set        : def2-TZVP")
print("Konvergen        :", dft_opt.converged)
print(f"Energi awal      : {energi_awal:.12f} Hartree")

# =====
# 4. OPTIMASI GEOMETRI
# =====

mol_h2_opt = optimize(
    dft_opt,
```

```
        maxsteps=1000
    )

# =====
# 5. MENAMPILKAN GEOMETRI HASIL OPTIMASI
# =====

koordinat_opt = mol_h2_opt.atom_coords(
    unit="Angstrom"
)

print("\n" + "=" * 65)
print("GEOMETRI HASIL OPTIMASI B3LYP/def2-TZVP")
print("=" * 65)

for i, xyz in enumerate(koordinat_opt):
    simbol = mol_h2_opt.atom_symbol(i)

    print(
        f"{simbol:2s}"
        f"{xyz[0]:16.10f}"
        f"{xyz[1]:16.10f}"
        f"{xyz[2]:16.10f}"
    )

# Menghitung panjang ikatan H-H hasil optimasi
panjang_hh = np.linalg.norm(
    koordinat_opt[1] - koordinat_opt[0]
)

print(f"\nPanjang ikatan H-H optimum = {panjang_hh:.8f} Å")

# =====
# 6. ENERGI PADA GEOMETRI OPTIMUM DENGAN def2-TZVP
# =====

dft_optimum_tzvp = dft.RKS(mol_h2_opt)
dft_optimum_tzvp.xc = "b3lyp"
dft_optimum_tzvp.grids.level = 4
dft_optimum_tzvp.conv_tol = 1.0e-10
dft_optimum_tzvp.max_cycle = 200

energi_optimum_tzvp = dft_optimum_tzvp.kernel()

if not dft_optimum_tzvp.converged:
    raise RuntimeError(
        "Perhitungan energi geometri optimum tidak konvergen."
    )

# =====
# 7. MEMBUAT MOLEKUL DENGAN BASIS def2-QZVP
#     PADA GEOMETRI HASIL OPTIMASI
# =====

mol_h2_qzvp = gto.M(
    atom=[
        (
            mol_h2_opt.atom_symbol(i),
```

```
        koordinat_opt[i]
    )
    for i in range(mol_h2_opt.natm)
],
unit="Angstrom",

# Basis set lebih besar untuk energi akhir
basis="def2-qzvp",

charge=0,
spin=0,
symmetry=True,
verbose=4
)

# =====
# 8. SINGLE-POINT ENERGY B3LYP/def2-QZVP
# =====

dft_final = dft.RKS(mol_h2_qzvp)
dft_final.xc = "b3lyp"

# Grid lebih rapat untuk energi akhir
dft_final.grids.level = 5

dft_final.conv_tol = 1.0e-11
dft_final.max_cycle = 200

energi_final = dft_final.kernel()

if not dft_final.converged:
    raise RuntimeError(
        "Perhitungan energi akhir B3LYP/def2-QZVP tidak konvergen."
    )

# =====
# 9. MEMISAHKAN ENERGI ELEKTRONIK DAN REPULSI INTI
# =====

energi_elektronik, energi_dua_elektron = (
    dft_final.energy_elec()
)

energi_repulsi_inti = dft_final.energy_nuc()

energi_total_periksa = (
    energi_elektronik + energi_repulsi_inti
)

# =====
# 10. MENAMPILKAN RINGKASAN HASIL
# =====

HARTREE_KE_EV = 27.211386245988
HARTREE_KE_KJMOL = 2625.499639

perubahan_energi = (
    energi_optimum_tzvvp - energi_awal
)
```

```
print("\n" + "=" * 65)
print("RINGKASAN HASIL PERHITUNGAN")
print("=" * 65)

print("Optimasi geometri : B3LYP/def2-TZVP")
print("Energi akhir      : B3LYP/def2-QZVP")

print(f"\nPanjang H-H optimum:")
print(f"    {panjang_hh:.10f} Å")

print("\nEnergi B3LYP/def2-TZVP:")
print(f"    Struktur awal    = {energi_awal:.12f} Hartree")
print(f"    Struktur optimum= {energi_optimum_tzvp:.12f} Hartree")

print("\nPerubahan energi selama optimasi:")
print(f"    {perubahan_energi:.12f} Hartree")
print(
    f"    {perubahan_energi * HARTREE_KE_KJMOL:.6f} kJ/mol"
)

print("\nEnergi akhir B3LYP/def2-QZVP:")
print(f"    {energi_final:.12f} Hartree")
print(f"    {energi_final * HARTREE_KE_EV:.6f} eV")

print("\nKomponen energi akhir:")
print(
    f"    Energi elektronik = "
    f"{energi_elektronik:.12f} Hartree"
)
print(
    f"    Repulsi inti      = "
    f"{energi_repulsi_inti:.12f} Hartree"
)
print(
    f"    Energi total      = "
    f"{energi_total_periksa:.12f} Hartree"
)

print("\nStatus SCF:")
print(
    "    B3LYP/def2-TZVP :",
    dft_optimum_tzvp.converged
)
print(
    "    B3LYP/def2-QZVP :",
    dft_final.converged
)
```

Contoh skrip menggunakan GPU untuk Paracetamol

```
# =====
# INSTALASI GPU4PySCF BERDASARKAN VERSI CUDA GOOGLE COLAB
# =====

import re
import subprocess
import sys
```



```
# -----
# 1. Memastikan GPU tersedia
# -----

try:
    informasi_gpu = subprocess.check_output(
        ["nvidia-smi"],
        stderr=subprocess.STDOUT,
        text=True
    )
except Exception as error:
    raise RuntimeError(
        "GPU tidak ditemukan. Aktifkan GPU melalui "
        "Runtime → Change runtime type → GPU."
    ) from error

print(informasi_gpu)

# -----
# 2. Mendeteksi versi utama CUDA
# -----

hasil_cuda = re.search(
    r"CUDA Version:\s*(\d+)\.",
    informasi_gpu
)

if hasil_cuda is None:
    try:
        informasi_nvcc = subprocess.check_output(
            ["nvcc", "--version"],
            stderr=subprocess.STDOUT,
            text=True
        )

        hasil_cuda = re.search(
            r"release\s*(\d+)\.",
            informasi_nvcc
        )

    except Exception:
        hasil_cuda = None

if hasil_cuda is None:
    raise RuntimeError(
        "Versi CUDA tidak berhasil dideteksi."
    )

cuda_major = int(hasil_cuda.group(1))

print("Versi utama CUDA:", cuda_major)

# -----
# 3. Menentukan paket GPU4PySCF
# -----

paket_gpu = {
    11: (
        "gpu4pyscf-cuda11x",
```

```
        "cutensor-cu11"
    ),
    12: (
        "gpu4pyscf-cuda12x",
        "cutensor-cu12"
    ),
    13: (
        "gpu4pyscf-cuda13x",
        "cutensor-cu13"
    ),
}

if cuda_major not in paket_gpu:
    raise RuntimeError(
        f"CUDA {cuda_major} belum ditangani oleh skrip ini."
    )

gpu4pyscf_package, cutensor_package = paket_gpu[cuda_major]

# -----
# 4. Memasang pustaka
# -----

paket = [
    "pyscf",
    "geometric",
    "rdkit",
    gpu4pyscf_package,
    cutensor_package,
]

print("\nMemasang paket:")
for nama in paket:
    print(" -", nama)

subprocess.check_call(
    [
        sys.executable,
        "-m",
        "pip",
        "install",
        "--upgrade",
        "--no-cache-dir",
        "--quiet",
        *paket,
    ]
)

print("\nInstalasi selesai.")
print(
    "Jika impor GPU4PySCF gagal, pilih "
    "Runtime → Restart session, lalu jalankan sel berikutnya."
)
```

5. Membaca hasil

Perhatikan dua bagian berikut:

converged SCF energy

dan:

```
Konvergen : True
```

Apabila hasilnya:

```
hf_h2.converged == True
```

berarti proses iterasi SCF telah mencapai konvergensi.

Energi yang ditampilkan adalah energi elektronik total dalam satuan **Hartree**.

Untuk mengubah energi Hartree menjadi elektronvolt:

```
HARTREE_KE_EV = 27.211386245988

energi_ev = energi_h2 * HARTREE_KE_EV

print("Energi dalam Hartree :", energi_h2)
print("Energi dalam eV      :", energi_ev)
```

Namun, energi absolut molekul biasanya lebih berguna untuk dibandingkan dengan energi lain yang dihitung menggunakan **metode dan basis set yang sama**.

Bagian C — Mempelajari Pengaruh Jarak Ikatan H₂

Sekarang kita hitung energi H₂ pada beberapa jarak ikatan.

6. Pemindaian jarak H–H

```
import numpy as np

jarak_hh = np.arange(0.40, 2.01, 0.10)
daftar_energi = []

for jarak in jarak_hh:
    mol = gto.M(
        atom=f"""
        H  0.0  0.0  0.0
        H  0.0  0.0  {jarak}
        """,
        unit="Angstrom",
        basis="sto-3g",
        charge=0,
        spin=0,
        verbose=0
    )

    mf = scf.RHF(mol)
    energi = mf.kernel()
```

```
daftar_energi.append(energi)

print(
    f"Jarak = {jarak:4.2f} Å | "
    f"Energi = {energi:12.8f} Hartree"
)
```

Atau

```
import numpy as np

from pyscf import gto, dft

# =====
# PEMINDAIAN KURVA ENERGI POTENSIAL MOLEKUL H2
#
# Metode      : DFT B3LYP
# Basis set:  def2-TZVP
# Rentang     : 0,40-2,00 Angstrom
# Interval    : 0,10 Angstrom
# =====

jarak_hh = np.arange(
    0.40,
    2.01,
    0.10
)

daftar_energi = []
daftar_konvergensi = []

# Matriks densitas sebelumnya dapat digunakan sebagai tebakan awal
# untuk mempercepat titik perhitungan berikutnya.
dm_awal = None

for jarak in jarak_hh:

    # -----
    # 1. Membuat molekul H2 pada jarak tertentu
    # -----

    mol = gto.M(
        atom=f"""
        H  0.000000  0.000000  0.000000
        H  0.000000  0.000000  {jarak:.8f}
        """,
        unit="Angstrom",

        # Basis set yang lebih baik daripada STO-3G
        basis="def2-tzvp",

        charge=0,

        # H2 keadaan dasar merupakan singlet tertutup
        spin=0,

        symmetry=True,
        verbose=0
```

```
)

# -----
# 2. Menentukan metode DFT B3LYP
# -----

mf = dft.RKS(mol)

mf.xc = "b3lyp"

# Grid integrasi numerik DFT
mf.grids.level = 4

# Kriteria konvergensi energi SCF
mf.conv_tol = 1.0e-10

# Jumlah maksimum iterasi
mf.max_cycle = 200

# -----
# 3. Menjalankan perhitungan energi
# -----

energi = mf.kernel(
    dm0=dm_awal
)

# -----
# 4. Jika SCF belum konvergen, gunakan Newton solver
# -----

if not mf.converged:

    print(
        f"SCF biasa belum konvergen pada "
        f"jarak {jarak:.2f} Å."
    )

    mf = mf.newton()
    mf.max_cycle = 200

    energi = mf.kernel()

# -----
# 5. Menyimpan hasil
# -----

if mf.converged:

    daftar_energi.append(energi)
    daftar_konvergensi.append(True)

    # Matriks densitas digunakan untuk titik berikutnya
    dm_awal = mf.make_rdm1()

    print(
        f"Jarak = {jarak:5.2f} Å | "
        f"Energi = {energi:16.10f} Hartree | "
```

```
        f"Konvergen"
    )

    else:

        daftar_energi.append(np.nan)
        daftar_konvergensi.append(False)

        print(
            f"Jarak = {jarak:5.2f} Å | "
            f"Energi = tidak tersedia | "
            f"Tidak konvergen"
        )

# =====
# MENENTUKAN TITIK DENGAN ENERGI TERENDAH
# =====

array_energi = np.array(
    daftar_energi,
    dtype=float
)

data_valid = np.isfinite(
    array_energi
)

if np.any(data_valid):

    indeks_valid = np.where(
        data_valid
    )[0]

    indeks_minimum_lokal = np.argmin(
        array_energi[data_valid]
    )

    indeks_minimum = indeks_valid[
        indeks_minimum_lokal
    ]

    jarak_minimum = jarak_hh[
        indeks_minimum
    ]

    energi_minimum = array_energi[
        indeks_minimum
    ]

    print("\n" + "=" * 68)
    print("HASIL PEMINDAIAN ENERGI H2")
    print("=" * 68)

    print("Metode           : DFT B3LYP")
    print("Basis set           : def2-TZVP")

    print(
        f"Jarak energi minimum: "
        f"{jarak_minimum:.4f} Å"
    )
)
```

```
print(
    f"Energi minimum      : "
    f"{energi_minimum:.12f} Hartree"
)

else:

    print(
        "\nTidak ada titik perhitungan "
        "yang berhasil konvergen."
    )
```

7. Membuat grafik energi potensial

```
import matplotlib.pyplot as plt

plt.figure(figsize=(8, 5))

plt.plot(
    jarak_hh,
    daftar_energi,
    marker="o"
)

plt.xlabel("Jarak H-H (Å)")
plt.ylabel("Energi total (Hartree)")
plt.title("Kurva Energi Potensial Molekul H2")
plt.grid()

plt.show()
```

Titik terendah pada grafik menunjukkan perkiraan jarak ikatan paling stabil pada tingkat teori HF/STO-3G.

Mencari energi minimum dari grafik:

```
indeks_minimum = np.argmin(daftar_energi)

jarak_optimum = jarak_hh[indeks_minimum]
energi_minimum = daftar_energi[indeks_minimum]

print(f"Jarak dengan energi minimum : {jarak_optimum:.2f} Å")
print(f"Energi minimum                : {energi_minimum:.8f} Hartree")
```

Bagian D — Perhitungan Molekul Air dengan DFT

Setelah memahami H₂, kita beralih ke molekul air:

```
[  
\mathrm{H_2O}  
]
```

Perhitungan menggunakan:

Komponen	Pilihan
Metode	DFT
Fungsional	B3LYP
Basis set	6-31G
Muatan	0
Spin	0

8. Membuat molekul H₂O

```
from pyscf import gto, dft  
  
geometri_air = """  
O   0.000000   0.000000   0.000000  
H   0.758602   0.000000   0.504284  
H  -0.758602   0.000000   0.504284  
"""  
  
mol_h2o = gto.M(  
    atom=geometri_air,  
    unit="Angstrom",  
    basis="6-31g",  
    charge=0,  
    spin=0,  
    verbose=4  
)
```

Arti parameter penting:

basis="6-31g"

menentukan basis set orbital atom.

charge=0

berarti molekul netral.

spin=0

digunakan untuk molekul singlet tertutup seperti H₂O.

9. Menjalankan DFT B3LYP

```
dft_h2o = dft.RKS(mol_h2o)
```



```
dft_h2o.xc = "b3lyp"
dft_h2o.conv_tol = 1e-9

energi_h2o = dft_h2o.kernel()

print("\nHasil DFT H2O")
print("Konvergen      :", dft_h2o.converged)
print(f"Energi total : {energi_h2o:.10f} Hartree")
```

RKS berarti **Restricted Kohn–Sham**, yang sesuai untuk sistem tertutup dengan semua elektron berpasangan.

Bagian E — Menghitung HOMO dan LUMO

HOMO adalah orbital molekul terisi dengan energi tertinggi, sedangkan LUMO adalah orbital kosong dengan energi terendah.

10. Mengambil energi orbital

```
import numpy as np

HARTREE_KE_EV = 27.211386245988

# Menentukan orbital terisi dan kosong
orbital_terisi = np.where(dft_h2o.mo_occ > 0)[0]
orbital_kosong = np.where(dft_h2o.mo_occ == 0)[0]

# Orbital terisi terakhir adalah HOMO
indeks_homo = orbital_terisi[-1]

# Orbital kosong pertama adalah LUMO
indeks_lumo = orbital_kosong[0]

energi_homo = dft_h2o.mo_energy[indeks_homo]
energi_lumo = dft_h2o.mo_energy[indeks_lumo]

celah_homo_lumo = energi_lumo - energi_homo

print("Indeks HOMO :", indeks_homo)
print("Indeks LUMO :", indeks_lumo)

print(
    f"Energi HOMO : "
    f"{energi_homo:.6f} Hartree = "
    f"{energi_homo * HARTREE_KE_EV:.4f} eV"
)

print(
    f"Energi LUMO : "
    f"{energi_lumo:.6f} Hartree = "
    f"{energi_lumo * HARTREE_KE_EV:.4f} eV"
)
```

```
print(
    f"Celah HOMO-LUMO : "
    f"{celah_homo_lumo:.6f} Hartree = "
    f"{celah_homo_lumo * HARTREE_KE_EV:.4f} eV"
)
```

Nilai tersebut adalah energi orbital Kohn–Sham pada tingkat B3LYP/6-31G. Celah HOMO–LUMO ini tidak otomatis sama dengan celah energi eksperimental karena bergantung pada metode, basis set, geometri, dan interpretasi energi orbital.

Bagian F — Menghitung Momen Dipol

11. Momen dipol H₂O

```
dipol = dft_h2o.dip_moment(unit="Debye")

besar_dipol = np.linalg.norm(dipol)

print("Komponen momen dipol :", dipol, "Debye")
print(f"Besar momen dipol      : {besar_dipol:.4f} Debye")
```

Hasil komponen biasanya ditampilkan sebagai:

```
[X, Y, Z]
```

Besar vektor momen dipol dihitung dengan:

```
[
    \mu=\sqrt{\mu_x^2+\mu_y^2+\mu_z^2}
]
```

Karena orientasi molekul bergantung pada koordinat yang dimasukkan, arah komponen dipol dapat berubah ketika molekul diputar. Akan tetapi, besar total momen dipol seharusnya tetap sama untuk geometri dan tingkat teori yang sama.

Bagian G — Visualisasi Molekul

12. Menampilkan struktur H₂O

```
import py3Dmol

xyz_h2o = f"""3
Molekul air
{geometri_air.strip()}
"""
```

```
view = py3Dmol.view(
    width=600,
    height=400
)

view.addModel(xyz_h2o, "xyz")

view.setStyle({
    "stick": {},
    "sphere": {
        "scale": 0.35
    }
})

view.zoomTo()
view.show()
```

Molekul dapat diputar menggunakan mouse pada tampilan hasil.

Bagian H — Optimasi Geometri

PySCF menyediakan antarmuka optimasi geometri melalui program seperti geomeTRIC dan PyBerny. (pyscf.org)

13. Mengoptimasi struktur H₂O

Pastikan paket `geometric` telah dipasang:

```
%pip install geometric
```

Kemudian jalankan:

```
from pyscf.geomopt.geometric_solver import optimize

mol_h2o_opt = optimize(
    dft_h2o,
    maxsteps=50
)
```

Tampilkan koordinat hasil optimasi:

```
koordinat_opt = mol_h2o_opt.atom_coords(
    unit="Angstrom"
)

print("Koordinat hasil optimasi:")

for i, koordinat in enumerate(koordinat_opt):
    simbol = mol_h2o_opt.atom_symbol(i)

    print(
        f"{simbol:2s} "
        f"{koordinat[0]:12.7f} "
```

```
f"{koordinat[1]:12.7f} "  
f"{koordinat[2]:12.7f}"  
)
```

14. Menghitung energi geometri hasil optimasi

```
dft_h2o_opt = dft.RKS(mol_h2o_opt)  
  
dft_h2o_opt.xc = "b3lyp"  
  
energi_h2o_opt = dft_h2o_opt.kernel()  
  
print(  
    f"Energi sebelum optimasi : "  
    f"{energi_h2o:.10f} Hartree"  
)  
  
print(  
    f"Energi setelah optimasi : "  
    f"{energi_h2o_opt:.10f} Hartree"  
)  
  
print(  
    f"Perubahan energi : "  
    f"{energi_h2o_opt - energi_h2o:.10f} Hartree"  
)
```

Hasil optimasi yang baik biasanya menunjukkan energi akhir yang lebih rendah atau setidaknya struktur yang memenuhi kriteria gradien optimasi.

Bagian I — Menyimpan Hasil ke File

15. Menyimpan koordinat hasil optimasi sebagai XYZ

```
nama_file = "H2O_teroptimasi.xyz"  
  
with open(nama_file, "w") as file:  
    file.write(f"{mol_h2o_opt.natm}\n")  
    file.write("H2O hasil optimasi B3LYP/6-31G\n")  
  
    koordinat = mol_h2o_opt.atom_coords(  
        unit="Angstrom"  
    )  
  
    for i, xyz in enumerate(koordinat):  
        simbol = mol_h2o_opt.atom_symbol(i)  
  
        file.write(  
            f"{simbol:2s} "  
            f"{xyz[0]:14.8f} "  
            f"{xyz[1]:14.8f} "  
            f"{xyz[2]:14.8f}\n"  
        )
```

```
print("File berhasil dibuat:", nama_file)
```

Mengunduh file:

```
from google.colab import files  
files.download("H2O_teroptimasi.xyz")
```

Bagian J — Menyimpan File di Google Drive

Karena mesin virtual Colab bersifat sementara, file yang hanya disimpan dalam direktori runtime dapat hilang setelah runtime berakhir. Notebook sendiri tersimpan di Google Drive, tetapi pustaka dan file lokal runtime tidak otomatis menjadi bagian dari notebook. ([Google Research](#))

Hubungkan Google Drive:

```
from google.colab import drive  
drive.mount("/content/drive")
```

Buat direktori kerja:

```
from pathlib import Path  
  
folder_hasil = Path(  
    "/content/drive/MyDrive/Kimia_Komputasi"  
)  
  
folder_hasil.mkdir(  
    parents=True,  
    exist_ok=True  
)  
  
print("Folder hasil:", folder_hasil)
```

Menyalin file XYZ:

```
import shutil  
  
shutil.copy(  
    "H2O_teroptimasi.xyz",  
    folder_hasil / "H2O_teroptimasi.xyz"  
)  
  
print("File tersimpan di Google Drive.")
```

Struktur Dasar Skrip PySCF

Hampir semua perhitungan sederhana mengikuti alur berikut:

```
# 1. Impor modul
from pyscf import gto, scf

# 2. Membuat molekul
mol = gto.M(
    atom="...",
    basis="...",
    charge=0,
    spin=0
)

# 3. Memilih metode
mf = scf.RHF(mol)

# 4. Menjalankan perhitungan
energi = mf.kernel()

# 5. Memeriksa hasil
print(mf.converged)
print(energi)
```

Untuk DFT, bagian metode diganti menjadi:

```
from pyscf import dft

mf = dft.RKS(mol)
mf.xc = "b3lyp"

energi = mf.kernel()
```

Kesalahan yang Sering Terjadi

ModuleNotFoundError: No module named 'pyscf'

Jalankan kembali:

```
%pip install --prefer-binary pyscf
```

Instalasi perlu diulangi ketika menggunakan runtime Colab baru karena pustaka khusus pada mesin virtual tidak disimpan bersama notebook. ([Google Research](https://research.google.com/colab-research-notebooks/))

Hasil menunjukkan `converged = False`

Tambahkan jumlah iterasi:

```
mf.max_cycle = 200
mf.kernel()
```

Untuk SCF yang sulit:

```
mf = mf.newton()
```

```
mf.kernel()
```

Kesalahan jumlah elektron dan spin

Pastikan kombinasi berikut benar:

```
charge=0  
spin=0
```

Untuk molekul tertutup netral seperti H₂, H₂O, CH₄, NH₃, dan CO₂, `spin=0` biasanya menjadi titik awal yang sesuai.

Optimasi geometri tidak berjalan

Pastikan `geometric` terpasang:

```
%pip install geometric
```

Kemudian lakukan impor ulang:

```
from pyscf.geomopt.geometric_solver import optimize
```

Urutan Latihan yang Disarankan

1. Energi H₂ dengan HF/STO-3G.
2. Kurva energi H₂ terhadap jarak ikatan.
3. H₂O dengan HF/STO-3G.
4. H₂O dengan HF/6-31G.
5. H₂O dengan B3LYP/6-31G.
6. HOMO, LUMO, dan momen dipol.
7. Optimasi geometri H₂O.
8. Molekul NH₃ dan CH₄.
9. Perbandingan basis set.
10. Perhitungan frekuensi vibrasi dan spektrum IR.

Notebook siap dijalankan:

[Download Tutorial Kimia Komputasi Dasar PySCF](#)

GLOSARIUM 1

Istilah	Definisi ringkas
Akselerator	Perangkat keras khusus, seperti GPU atau TPU, yang dapat mempercepat jenis operasi tertentu.
Angstrom	Satuan panjang 10^{-10} meter yang umum untuk jarak atom.
Artefak	Berkas atau keluaran yang dihasilkan proses, seperti notebook, log, XYZ, grafik, dan laporan.
Basis set	Kumpulan fungsi matematika untuk merepresentasikan orbital atom dalam perhitungan kuantum.
B3LYP	Fungsional hibrida DFT yang menggabungkan pertukaran dan korelasi.
Cell ID	Pengenal sel pada format notebook modern.
Charge	Muatan total sistem dalam satuan muatan elementer.
Penggunaan istilah: Istilah teknis dipertahankan bila terjemahan dapat menimbulkan ambiguitas. Definisi dalam glosarium harus dibaca bersama konteks bab.	

Glosarium - Bagian 2

Istilah	Definisi ringkas
Checkpoint	Berkas antara untuk melanjutkan atau memeriksa proses.
Colab	Layanan notebook Jupyter terkelola berbasis web dari Google.

Conformer	Susunan tiga dimensi molekul yang berbeda akibat rotasi ikatan atau konfigurasi ruang.
Convergence	Keadaan ketika iterasi memenuhi kriteria perubahan yang ditetapkan.
CPU	Unit pemrosesan umum yang menjalankan sebagian besar kode Python.
CUDA	Platform dan ekosistem komputasi GPU NVIDIA.
cuTENSOR	Pustaka tensor NVIDIA yang digunakan oleh sebagian backend GPU.
Penggunaan istilah: Istilah teknis dipertahankan bila terjemahan dapat menimbulkan ambiguitas. Definisi dalam glosarium harus dibaca bersama konteks bab.	

Glosarium - Bagian 3

Istilah	Definisi ringkas
Density fitting	Pendekatan aproksimasi integral untuk mengurangi biaya komputasi.
DFT	Density Functional Theory, metode struktur elektronik berbasis kerapatan elektron.
Dipole moment	Besaran vektor yang menggambarkan pemisahan muatan.
ETKDGv3	Metode RDKit untuk menghasilkan konformer 3D dengan pengetahuan torsional.
Fallback	Jalur alternatif ketika jalur utama tidak tersedia, misalnya GPU ke CPU.
Force field	Model energi klasik untuk geometri dan konformasi molekul.

Frekuensi imajiner

Indikator arah kelengkungan negatif pada titik stasioner.

Penggunaan istilah: Istilah teknis dipertahankan bila terjemahan dapat menimbulkan ambiguitas. Definisi dalam glosarium harus dibaca bersama konteks bab.

Glosarium - Bagian 4

Istilah	Definisi ringkas
geomeTRIC	Program optimasi geometri yang dapat diakses melalui antarmuka PySCF.
GPU	Unit pemrosesan paralel yang dapat mempercepat operasi terdukung.
GPU4PySCF	Plugin akselerasi GPU untuk sebagian fungsi PySCF.
Hartree	Satuan energi atom, sekitar 27.2114 eV.
Hessian	Matriks turunan kedua energi terhadap koordinat.
HF	Hartree-Fock, metode medan rata-rata berbasis fungsi gelombang.
HOMO	Orbital molekul terisi dengan energi tertinggi.
Penggunaan istilah: Istilah teknis dipertahankan bila terjemahan dapat menimbulkan ambiguitas. Definisi dalam glosarium harus dibaca bersama konteks bab.	

Glosarium - Bagian 5

Istilah	Definisi ringkas
IPYNB	Ekstensi berkas notebook Jupyter berbasis JSON.
JSON	Format teks terstruktur pasangan kunci-nilai.

Jupyter	Ekosistem notebook interaktif untuk kode, teks, data, dan media.
Kernel	Proses komputasi yang menjalankan kode notebook.
Kohn-Sham	Formulasi DFT dengan orbital bantu untuk menghasilkan kerapatan.
LUMO	Orbital molekul kosong dengan energi terendah.
Markdown	Sintaks teks ringan untuk dokumentasi notebook.

Penggunaan istilah: Istilah teknis dipertahankan bila terjemahan dapat menimbulkan ambiguitas. Definisi dalam glosarium harus dibaca bersama konteks bab.

Glosarium - Bagian 6

Istilah	Definisi ringkas
MMFF94	Force field Merck yang digunakan untuk optimasi molekul organik.
Molden	Format berkas untuk orbital dan visualisasi kimia kuantum.
Multiplicity	$2S+1$; pada PySCF spin diisi sebagai jumlah elektron alfa minus beta.
nbformat	Pustaka Python dan skema resmi untuk membaca, membuat, serta memvalidasi notebook.
Notebook	Dokumen interaktif yang menggabungkan sel kode, narasi, output, dan metadata.
NumPy	Pustaka array numerik Python.
Orbital occupation	Jumlah elektron yang menempati orbital molekul.

Penggunaan istilah: Istilah teknis dipertahankan bila terjemahan dapat menimbulkan ambiguitas. Definisi dalam glosarium harus dibaca bersama konteks bab.

Glosarium - Bagian 7

Istilah	Definisi ringkas
Pandas	Pustaka data tabular Python.
Parameter	Nilai yang mengendalikan proses dan harus terdokumentasi.
PES	Potential Energy Surface, hubungan energi dengan koordinat inti.
Provenance	Asal-usul data, struktur, parameter, perangkat, dan proses.
py3Dmol	Antarmuka Python untuk visualisasi molekul 3Dmol.js.
PySCF	Kerangka Python sumber terbuka untuk simulasi kimia kuantum.
Random seed	Nilai awal generator acak untuk membantu reproduksibilitas.

Penggunaan istilah: Istilah teknis dipertahankan bila terjemahan dapat menimbulkan ambiguitas. Definisi dalam glosarium harus dibaca bersama konteks bab.

Glosarium - Bagian 8

Istilah	Definisi ringkas
RDF/RHF	Restricted Hartree-Fock untuk sistem berpasangan.
RDKit	Toolkit cheminformatika sumber terbuka.
Reproducibility	Kemampuan mengulang prosedur dan memperoleh hasil konsisten dalam kondisi yang dilaporkan.

RKS	Restricted Kohn-Sham untuk sistem closed-shell.
Runtime	Mesin virtual dan lingkungan eksekusi yang terhubung ke notebook.
SCF	Self-Consistent Field, iterasi untuk memperoleh solusi medan rata-rata konsisten.
SDF	Structure Data File yang dapat menyimpan satu atau banyak record molekul.

Penggunaan istilah: Istilah teknis dipertahankan bila terjemahan dapat menimbulkan ambiguitas. Definisi dalam glosarium harus dibaca bersama konteks bab.

Glosarium - Bagian 9

Istilah	Definisi ringkas
SMILES	Notasi linear untuk struktur dan konektivitas molekul.
Spin PySCF	$N_{\alpha} - N_{\beta}$, sama dengan multiplicity - 1.
State kernel	Kumpulan variabel dan impor yang masih tersimpan dalam proses kernel.
STO-3G	Basis set minimal yang sering dipakai untuk demonstrasi.
TPU	Tensor Processing Unit; tidak otomatis relevan untuk PySCF.
UHF	Unrestricted Hartree-Fock untuk sistem open-shell.
UKS	Unrestricted Kohn-Sham untuk sistem open-shell.

Penggunaan istilah: Istilah teknis dipertahankan bila terjemahan dapat menimbulkan ambiguitas. Definisi dalam glosarium harus dibaca bersama konteks bab.

Glosarium - Bagian 10

Istilah	Definisi ringkas
UFF	Universal Force Field untuk optimasi klasik berbagai unsur.
Validation	Pemeriksaan sistematis atas format, sintaks, kimia, numerik, dan keluaran.
XYZ	Format sederhana jumlah atom, komentar, simbol, dan koordinat 3D.
ZIP	Format arsip untuk menggabungkan berkas hasil.

Penggunaan istilah: Istilah teknis dipertahankan bila terjemahan dapat menimbulkan ambiguitas. Definisi dalam glosarium harus dibaca bersama konteks bab.

GLOSARIUM 2

A

Istilah	Definisi singkat	Penjelasan kontekstual dan padanan
Akselerator	Perangkat keras khusus untuk mempercepat operasi komputasi tertentu.	Dalam buku merujuk terutama pada GPU dan TPU. Padanan: <i>hardware accelerator</i> .
Alhamdulillah rabbil ‘ālamīn	Ungkapan Arab yang berarti “segala puji bagi Allah, Tuhan seluruh alam.”	Digunakan dalam kata pengantar sebagai ungkapan syukur atas tersusunnya buku.
Angstrom	Satuan panjang sebesar (10^{-10}) meter.	Digunakan untuk menyatakan koordinat, panjang ikatan, dan jarak antaratom. Simbol: Å.
Antarmuka	Bagian visual yang menghubungkan pengguna dengan aplikasi.	Mencakup menu, toolbar, area sel, panel berkas, dan indikator runtime Google Colab. Padanan: <i>interface</i> .

Array	Struktur data multidimensi yang berisi elemen bertipe seragam.	Digunakan dalam NumPy untuk koordinat atom, data energi, matriks, dan perhitungan numerik.
Artefak	Berkas atau keluaran yang dihasilkan selama proses komputasi.	Contohnya notebook, grafik, log, CSV, JSON, XYZ, Molden, dan arsip ZIP.

B

Istilah	Definisi singkat	Penjelasan kontekstual dan padanan
B3LYP	Fungsional hibrida dalam Density Functional Theory.	Digunakan untuk optimasi geometri dan analisis molekul, termasuk H ₂ O dan parasetamol.
Backend	Sistem atau perangkat komputasi yang menjalankan proses di balik program.	Buku membedakan backend CPU dan GPU serta menyediakan mekanisme <i>fallback</i> .
Basis set	Kumpulan fungsi matematika untuk merepresentasikan orbital atom.	Digunakan dalam perhitungan HF dan DFT, misalnya STO-3G dan def2-TZVP.
Berkas biner	Berkas yang tidak disimpan sebagai teks biasa.	Paket biner harus kompatibel dengan versi Python, CUDA, dan lingkungan runtime. Padanan: <i>binary file/package</i> .
Broadcasting	Mekanisme NumPy untuk mengoperasikan array dengan bentuk berbeda yang kompatibel.	Digunakan untuk menyederhanakan perhitungan numerik tanpa perulangan manual.

C

Istilah	Definisi singkat	Penjelasan kontekstual dan padanan
Cell ID	Pengenal unik sebuah sel dalam format notebook.	Membantu mempertahankan struktur dan identitas sel pada berkas IPYNB modern.
Charge	Muatan listrik total suatu sistem molekul.	Harus ditentukan dengan benar sebelum perhitungan PySCF. Padanan: muatan total.
Checkpoint	Penanda atau keluaran antara untuk memeriksa kemajuan proses.	Ditempatkan pada tahap penting agar kegagalan dapat dilacak dan proses dapat diverifikasi.

Checksum	Nilai ringkas yang digunakan untuk memeriksa integritas berkas.	Buku menggunakan SHA-256 atau metode serupa dalam manifest hasil.
Cloud computing	Penggunaan sumber daya komputasi melalui jaringan internet.	Google Colab menyediakan notebook dan runtime berbasis awan tanpa instalasi lokal penuh.
Colab	Layanan notebook Jupyter berbasis web yang dikelola Google.	Menyediakan sel kode, Markdown, CPU, GPU, penyimpanan sementara, dan integrasi Google Drive.
Conformer	Susunan tiga dimensi molekul akibat rotasi ikatan dan variasi ruang.	Dibangkitkan menggunakan RDKit, kemudian dapat dioptimasi dengan MMFF94 atau UFF.
Convergence	Keadaan ketika proses iteratif memenuhi ambang perubahan yang ditetapkan.	Status konvergensi harus diperiksa sebelum energi atau sifat molekul ditafsirkan. Padanan: konvergensi.
CPU	Unit pemrosesan umum komputer.	Menjadi backend standar dan jalur alternatif ketika GPU tidak tersedia.
CSV	Format teks tabular dengan nilai yang dipisahkan tanda tertentu.	Digunakan untuk menyimpan data numerik, parameter, energi, dan ringkasan hasil.
CUDA	Platform komputasi paralel untuk GPU NVIDIA.	Kompatibilitas CUDA menentukan keberhasilan pemasangan CuPy dan GPU4PySCF.
cuTENSOR	Pustaka NVIDIA untuk operasi tensor pada GPU.	Dapat menjadi salah satu dependensi backend GPU dalam komputasi PySCF.

Istilah A–C tersebut sesuai dengan terminologi perangkat keras, artefak, basis set, Colab, konformer, CPU, CUDA, dan cuTENSOR yang digunakan dalam glosarium asli buku.

D

Istilah	Definisi singkat	Penjelasan kontekstual dan padanan
DataFrame	Struktur data tabular dua dimensi dalam Pandas.	Digunakan untuk membaca, membersihkan, mengelompokkan, menganalisis, dan mengeksplor data.
def2-TZVP	Basis set valensi rangkap tiga dengan fungsi polarisasi.	Digunakan dalam contoh optimasi geometri B3LYP untuk H ₂ O dan parasetamol.
Density fitting	Aproksimasi integral elektron untuk mengurangi biaya komputasi.	Dapat mempercepat perhitungan DFT atau HF pada PySCF.

DFT	Metode struktur elektronik berbasis kerapatan elektron.	Kepanjangan: <i>Density Functional Theory</i> . B3LYP merupakan salah satu fungsional DFT.
Diagnosis kesalahan	Proses menentukan penyebab kegagalan berdasarkan bukti.	Dilakukan dengan membaca pesan error, versi, tipe data, bentuk array, jalur berkas, dan parameter.
Dipole moment	Besaran vektor yang menggambarkan pemisahan muatan.	Dihitung sebagai salah satu sifat molekul setelah perhitungan elektronik. Padanan: momen dipol.
Dokumentasi versi	Pencatatan versi Python, pustaka, runtime, dan perangkat.	Diperlukan agar notebook dapat diuji ulang dan hasilnya dapat ditelusuri.

E

Istilah	Definisi singkat	Penjelasan kontekstual dan padanan
Eigenvalue	Nilai skalar yang berkaitan dengan transformasi suatu matriks.	Muncul dalam latihan aljabar linear dan analisis matriks simetris. Padanan: nilai eigen.
Elektron alfa	Elektron dengan salah satu orientasi spin.	Selisih jumlah elektron alfa dan beta digunakan untuk menentukan parameter spin pada PySCF.
Elektron beta	Elektron dengan orientasi spin berlawanan terhadap elektron alfa.	Bersama elektron alfa menentukan keadaan spin dan multiplicity sistem.
Encoding	Aturan representasi karakter dalam berkas.	UTF-8 digunakan agar berkas teks dapat ditulis dan dibaca secara konsisten.
Energi elektronik	Energi yang diperoleh dari penyelesaian struktur elektronik molekul.	Umumnya dinyatakan dalam Hartree dan hanya ditafsirkan setelah SCF konvergen.
Environment	Kumpulan sistem, versi Python, pustaka, dan variabel runtime.	Direkam melalui <code>pip freeze</code> , <code>requirements.txt</code> , atau <code>environment.json</code> . Padanan: lingkungan komputasi.
ETKDGv3	Metode RDKit untuk menghasilkan konformer tiga dimensi.	Memanfaatkan geometri jarak dan pengetahuan torsional molekul.
Exception	Kondisi kesalahan yang menghentikan alur normal program.	Ditangani dengan <code>try-except</code> yang spesifik dan pesan kesalahan informatif.

Ekspor	Proses menyimpan keluaran ke format berkas tertentu.	Hasil dapat diekspor sebagai TXT, CSV, JSON, XYZ, Molden, PNG, PDF, atau ZIP.

F

Istilah	Definisi singkat	Penjelasan kontekstual dan padanan
Fallback	Jalur alternatif ketika jalur utama tidak tersedia.	Contoh utama adalah menggunakan CPU apabila backend GPU gagal atau tidak kompatibel.
Force field	Model energi klasik untuk menghitung geometri dan konformasi molekul.	Buku menggunakan MMFF94 dan UFF untuk optimasi awal sebelum metode kuantum.
Format berkas	Aturan penyusunan dan penyimpanan data dalam sebuah berkas.	Buku membahas TXT, CSV, JSON, SDF, MOL, XYZ, Molden, IPYNB, PNG, PDF, dan ZIP.
Frekuensi imajiner	Frekuensi hasil analisis Hessian yang menunjukkan kelengkungan negatif.	Digunakan untuk menilai apakah geometri merupakan minimum atau titik stasioner lain.
Fungsi modular	Fungsi yang dirancang untuk menjalankan satu tugas yang jelas dan dapat digunakan ulang.	Membantu notebook lebih terstruktur, mudah diuji, dan mudah dipelihara.

Definisi DFT, *density fitting*, momen dipol, ETKDGv3, *fallback*, *force field*, dan frekuensi imajiner berasal langsung dari kerangka istilah komputasi molekul buku.

G

Istilah	Definisi singkat	Penjelasan kontekstual dan padanan
Generator acak	Algoritma yang menghasilkan urutan bilangan semuacak.	Digunakan dalam pembangkitan konformer atau data simulasi bersama <i>random seed</i> .
Geometri molekul	Susunan tiga dimensi atom-atom dalam molekul.	Dinyatakan dengan koordinat Cartesian serta dianalisis melalui jarak dan sudut ikatan.
geomeTRIC	Program optimasi geometri berbasis koordinat internal.	Diakses melalui antarmuka optimasi PySCF untuk memperoleh struktur berenergi lebih rendah.
Google Drive	Layanan penyimpanan awan dari Google.	Digunakan untuk menyimpan notebook dan hasil agar tidak hilang saat runtime berakhir.

GPU	Unit pemrosesan paralel yang cocok untuk operasi numerik tertentu.	Dapat mempercepat bagian perhitungan PySCF yang didukung.
GPU4PySCF	Ekstensi PySCF yang memanfaatkan GPU.	Digunakan untuk akselerasi DFT dengan pemeriksaan kompatibilitas CUDA dan dependensi.
Grafik	Representasi visual data numerik.	Dibuat dengan Matplotlib dan harus memiliki judul, label sumbu, satuan, serta legenda bila diperlukan.

H

Istilah	Definisi singkat	Penjelasan kontekstual dan padanan
Hartree	Satuan energi atom yang setara sekitar 27,2114 eV.	Digunakan untuk melaporkan energi elektronik PySCF.
Hartree–Fock	Metode struktur elektronik berbasis pendekatan medan rata-rata.	Disingkat HF dan menjadi dasar perbandingan dengan metode DFT.
Hash	Nilai ringkas yang dihasilkan dari isi suatu berkas.	Digunakan untuk mendeteksi perubahan atau memeriksa integritas artefak.
Hessian	Matriks turunan kedua energi terhadap koordinat atom.	Digunakan dalam analisis frekuensi dan karakterisasi titik stasioner.
HF	Singkatan dari <i>Hartree–Fock</i> .	Dalam buku digunakan untuk perhitungan struktur elektronik dasar, misalnya energi H ₂ .
HOMO	Orbital molekul terisi dengan energi tertinggi.	Dianalisis bersama LUMO untuk menentukan celah energi orbital. <i>Highest Occupied Molecular Orbital</i> .
HTML	Bahasa markah untuk menyusun halaman web.	Digunakan dalam salah satu resep untuk menghasilkan laporan sederhana dari notebook.

I

Istilah	Definisi singkat	Penjelasan kontekstual dan padanan
Import	Perintah untuk memuat modul atau objek Python.	Keberhasilan impor digunakan untuk memeriksa pemasangan pustaka dan kompatibilitas lingkungan.

ImportError	Kesalahan ketika modul ada tetapi komponen yang diminta gagal dimuat.	Dapat muncul setelah perubahan versi; sering memerlukan pemeriksaan dependensi dan restart.
Indeks atom	Nomor posisi atom dalam objek molekul.	Digunakan untuk mengambil simbol atom, koordinat, atau memberi label pada visualisasi.
Input	Data dan parameter yang dimasukkan ke dalam proses.	Harus memuat tipe, bentuk, satuan, muatan, spin, koordinat, dan jalur berkas secara eksplisit.
Integritas data	Keadaan ketika data tetap utuh, konsisten, dan tidak berubah tanpa diketahui.	Diperiksa melalui ukuran berkas, pembacaan ulang, manifest, dan checksum.
Interpretasi ilmiah	Penjelasan makna hasil berdasarkan metode dan batas penerapannya.	Harus dibedakan dari keluaran mentah program dan tidak boleh melampaui data yang dihitung.
IPYNB	Ekstensi berkas notebook Jupyter berbasis JSON.	Menyimpan sel kode, Markdown, output, metadata, dan identitas sel.

J

Istilah	Definisi singkat	Penjelasan kontekstual dan padanan
Jalur berkas	Alamat lokasi sebuah berkas dalam sistem penyimpanan.	Dikelola dengan <code>pathlib</code> agar kode lebih aman dan mudah dipindahkan. Padanan: <i>file path</i> .
JSON	Format teks terstruktur berbasis pasangan kunci dan nilai.	Digunakan untuk konfigurasi, metadata, manifest, dan ringkasan hasil.
Jupyter	Ekosistem notebook interaktif untuk kode, teks, data, dan media.	Google Colab menggunakan format dan konsep notebook Jupyter.
Justifikasi hasil	Alasan berbasis bukti untuk menerima atau menolak suatu keluaran.	Mengacu pada status konvergensi, pemeriksaan numerik, metode, dan batas kewajaran.

K

Istilah	Definisi singkat	Penjelasan kontekstual dan padanan
Kasus uji	Input terencana untuk memeriksa perilaku program.	Buku meminta sedikitnya kasus benar dan kasus salah dengan pesan yang informatif. Padanan: <i>test case</i> .

Kernel	Proses komputasi yang menjalankan kode notebook.	Menyimpan variabel, impor, dan objek selama sesi masih aktif.
Kerapatan elektron	Distribusi probabilitas elektron dalam ruang.	Menjadi besaran utama dalam Density Functional Theory.
Kimia komputasi	Penggunaan metode matematis dan komputasi untuk mempelajari sistem kimia.	Buku menerapkannya pada H ₂ , H ₂ O, konformer, optimasi geometri, dan parasetamol.
Kohn–Sham	Formulasi DFT yang menggunakan orbital bantu untuk menghasilkan kerapatan elektron.	Menjadi dasar metode RKS dan UKS dalam PySCF.
Konfigurasi	Kumpulan nilai pengaturan yang mengendalikan program.	Parameter sebaiknya diletakkan pada satu bagian agar mudah diubah dan direkam.
Konformer	Lihat Conformer .	Bentuk ruang molekul yang berbeda tanpa mengubah konektivitas atom.
Konvergensi	Lihat Convergence .	Menentukan apakah iterasi SCF atau optimasi telah mencapai kriteria keberhasilan.
Koordinat Cartesian	Posisi atom yang dinyatakan sebagai nilai (x), (y), dan (z).	Digunakan dalam format XYZ dan sebagai input geometri PySCF.

Istilah G–K mendukung alur buku dari optimasi molekul, penggunaan GPU, analisis orbital, format notebook, hingga formulasi Kohn–Sham.

L

Istilah	Definisi singkat	Penjelasan kontekstual dan padanan
Label sumbu	Teks yang menjelaskan variabel dan satuan pada sumbu grafik.	Wajib dicantumkan agar grafik dapat ditafsirkan dengan benar.
Library	Kumpulan modul dan fungsi yang dapat digunakan oleh program.	Dalam buku meliputi NumPy, Pandas, Matplotlib, RDKit, py3Dmol, PySCF, dan geomeTRIC. Padanan: pustaka.
Linear algebra	Cabang matematika tentang vektor, matriks, dan transformasi linear.	Digunakan dalam latihan NumPy, penyelesaian persamaan linear, dan nilai eigen.
Log	Catatan teks mengenai tahapan dan status eksekusi program.	Digunakan untuk pelacakan kesalahan, waktu proses, dan dokumentasi pengujian.
LUMO	Orbital molekul kosong dengan energi terendah.	Dibandingkan dengan HOMO untuk memperoleh celah energi. <i>Lowest Unoccupied Molecular Orbital</i> .

M

Istilah	Definisi singkat	Penjelasan kontekstual dan padanan
Manifest	Daftar terstruktur yang menjelaskan berkas hasil dan atributnya.	Dapat memuat nama, ukuran, checksum, dan waktu pembuatan artefak.
Markdown	Sintaks teks ringan untuk membuat dokumentasi terformat.	Digunakan dalam sel narasi untuk judul, teori, prosedur, tabel, dan interpretasi.
Matplotlib	Pustaka Python untuk visualisasi data.	Digunakan untuk grafik garis, pencar, histogram, heatmap, dan ekspor gambar.
Metadata	Data yang menjelaskan data atau dokumen lain.	Notebook menyimpan metadata kernel, bahasa, sel, dan konfigurasi.
Metode numerik	Teknik pendekatan untuk menyelesaikan masalah matematika secara komputasional.	Mencakup interpolasi, turunan, integral, dan aljabar linear.
MMFF94	<i>Merck Molecular Force Field 1994.</i>	Digunakan untuk optimasi klasik molekul organik dalam RDKit.
MOL	Format berkas untuk menyimpan struktur dan ikatan molekul.	Menjadi salah satu format representasi molekul yang dibahas.
Molden	Format berkas untuk orbital dan visualisasi hasil kimia kuantum.	Digunakan untuk mengeksport data yang dapat dibuka dalam program visualisasi tertentu.
Momen dipol	Lihat Dipole moment .	Menunjukkan besarnya dan arah pemisahan muatan dalam molekul.
Multiplicity	Nilai $(2S+1)$ yang menyatakan keadaan spin sistem.	Harus dibedakan dari parameter $spin$ PySCF, yaitu $(N_{\alpha}-N_{\beta})$.
ModuleNotFoundError	Kesalahan karena modul Python tidak ditemukan.	Diatasi dengan memeriksa nama paket, menjalankan <code>%pip</code> , dan memastikan runtime yang aktif.

N

Istilah	Definisi singkat	Penjelasan kontekstual dan padanan
NaN	Nilai numerik yang menyatakan hasil tidak terdefinisi atau hilang.	Harus dideteksi sebelum data dianalisis atau hasil komputasi diterima. <i>Not a Number</i> .
nbformat	Pustaka dan skema resmi untuk membaca serta memvalidasi notebook Jupyter.	Digunakan untuk memastikan dokumen IPYNB memiliki struktur yang sah.
Neural accelerator	Perangkat yang dioptimalkan untuk operasi pembelajaran mesin.	TPU dibahas sebagai akselerator, tetapi tidak otomatis cocok untuk PySCF.
Newton solver	Metode iteratif berbasis turunan untuk memperbaiki penyelesaian.	Dapat digunakan sebagai strategi tambahan ketika SCF sulit konvergen.
Notebook	Dokumen interaktif yang menggabungkan kode, narasi, keluaran, dan metadata.	Menjadi media utama seluruh kegiatan pembelajaran dalam buku.
NumPy	Pustaka Python untuk array dan komputasi numerik.	Digunakan untuk vektor, matriks, koordinat, statistik, interpolasi, dan operasi numerik.
Numerik	Berkaitan dengan data atau perhitungan berbentuk bilangan.	Validasi numerik mencakup pemeriksaan NaN, inf, bentuk array, jumlah atom, dan batas kewajaran.

O

Istilah	Definisi singkat	Penjelasan kontekstual dan padanan
Open-shell	Sistem yang memiliki elektron tidak berpasangan.	Umumnya dihitung menggunakan UHF atau UKS.
Operator	Simbol atau kata yang menjalankan operasi terhadap suatu nilai.	Dibahas dalam dasar Python untuk operasi aritmetika, perbandingan, dan logika.
Optimasi geometri	Proses mencari susunan atom dengan energi lebih rendah atau titik stasioner.	Dilakukan dengan geomeTRIC dan metode energi PySCF.
Orbital molekul	Fungsi yang menggambarkan perilaku elektron dalam molekul.	Buku membahas energi orbital, okupasi, HOMO, dan LUMO.
Orbital occupation	Jumlah elektron yang menempati suatu orbital.	Digunakan untuk mengidentifikasi orbital terisi dan kosong.

Output	Hasil yang dikeluarkan oleh program atau sel notebook.	Dapat berupa angka, teks, tabel, grafik, struktur molekul, atau berkas.

P

Istilah	Definisi singkat	Penjelasan kontekstual dan padanan
Pandas	Pustaka Python untuk pengelolaan data tabular.	Digunakan untuk DataFrame, CSV, pembersihan data, pengelompokan, pivot, dan ekspor.
Parameter	Nilai yang mengendalikan perilaku atau proses komputasi.	Harus dinyatakan eksplisit, terpusat, terdokumentasi, dan tidak tersembunyi.
Parasetamol	Senyawa obat yang digunakan sebagai proyek akhir komputasi.	Proyek mencakup pembuatan struktur, optimasi B3LYP/def2-TZVP, analisis, dan ekspor.
Pathlib	Modul Python untuk mengelola jalur dan operasi berkas.	Digunakan untuk membuat folder, membaca, menulis, mencari, dan memeriksa berkas.
PES	Hubungan antara energi dan koordinat inti atom.	Kepanjangan: <i>Potential Energy Surface</i> atau permukaan energi potensial.
Pip	Pengelola paket Python.	Dalam notebook digunakan melalui <code>%pip install</code> agar paket dipasang pada kernel yang aktif.
Provenance	Informasi mengenai asal-usul data, struktur, parameter, perangkat, dan proses.	Penting untuk audit, pelacakan hasil, dan reproduksibilitas.
Pustaka	Kumpulan fungsi atau modul yang dapat digunakan oleh program.	Padanan bahasa Inggris: <i>library</i> .
py3Dmol	Antarmuka Python untuk pustaka visualisasi 3Dmol.js.	Digunakan untuk menampilkan, memberi gaya, melabeli, dan membandingkan struktur molekul.
PySCF	Kerangka Python sumber terbuka untuk simulasi kimia kuantum.	Digunakan untuk HF, DFT, energi, optimasi, orbital, dan momen dipol.
Python	Bahasa pemrograman tingkat tinggi yang digunakan dalam buku.	Menjadi dasar pengolahan data, visualisasi, automasi, dan komputasi molekul.

Istilah L–P merepresentasikan pustaka utama, dokumentasi notebook, struktur molekul, metode numerik, dan proyek parasetamol. Buku secara khusus menempatkan NumPy,

Pandas, Matplotlib, RDKit, py3Dmol, PySCF, geomeTRIC, dan GPU4PySCF sebagai perangkat pembelajaran bertahap.

Q

Istilah	Definisi singkat	Penjelasan kontekstual dan padanan
Quantum chemistry	Cabang kimia yang menggunakan mekanika kuantum untuk mempelajari sistem molekul.	Menjadi dasar teoretis perhitungan HF, DFT, orbital, energi, dan sifat molekul. Padanan: kimia kuantum.
Query	Instruksi untuk memilih atau menyaring data berdasarkan syarat tertentu.	Digunakan melalui <code>DataFrame.query()</code> pada pengelolaan data Pandas.

R

Istilah	Definisi singkat	Penjelasan kontekstual dan padanan
Random seed	Nilai awal untuk generator bilangan acak.	Dicatat agar pembangkitan data atau konformer lebih mudah diulang.
RDF/RHF	Metode Hartree–Fock terbatas untuk sistem elektron berpasangan.	Istilah yang digunakan buku terutama adalah RHF; RDF muncul sebagai rujukan silang pada indeks.
RDKit	Toolkit sumber terbuka untuk cheminformatika.	Digunakan untuk membaca SMILES, menambah hidrogen, membuat konformer, dan optimasi klasik.
Record molekul	Satu unit data struktur molekul dalam sebuah berkas.	Format SDF dapat menyimpan satu atau banyak record molekul.
Reproducibility	Kemampuan mengulang prosedur dan memperoleh hasil konsisten dalam kondisi yang dilaporkan.	Diuji melalui runtime bersih, <code>Restart session</code> , <code>Run all</code> , pencatatan versi, dan penyimpanan artefak.
Requirements.txt	Berkas daftar paket dan versi yang diperlukan program.	Digunakan untuk mendokumentasikan atau memulihkan lingkungan komputasi.
Restart session	Perintah untuk memulai ulang proses runtime atau kernel.	Digunakan untuk membersihkan state lama dan memuat dependensi baru.

RKS	Metode Kohn–Sham terbatas untuk sistem <i>closed-shell</i> .	Digunakan dalam perhitungan DFT molekul dengan elektron berpasangan.
Run all	Perintah menjalankan semua sel secara berurutan.	Menjadi pengujian utama bahwa notebook dapat berjalan dari keadaan bersih.
Runtime	Mesin virtual dan lingkungan eksekusi yang terhubung ke notebook.	Mencakup Python, RAM, disk, CPU atau GPU, serta paket yang terpasang.

S

Istilah	Definisi singkat	Penjelasan kontekstual dan padanan
SCF	Prosedur iteratif untuk memperoleh solusi medan rata-rata yang konsisten.	Kepanjangan: <i>Self-Consistent Field</i> . Statusnya harus konvergen sebelum hasil dipakai.
SDF	Format data struktur yang dapat menyimpan satu atau banyak molekul.	Kepanjangan: <i>Structure Data File</i> .
Sel kode	Bagian notebook yang berisi perintah yang dapat dieksekusi.	Menghasilkan output dan berinteraksi dengan state kernel.
Sel Markdown	Bagian notebook yang berisi teks terformat.	Digunakan untuk tujuan, teori, prosedur, penjelasan, dan interpretasi.
Shape	Ukuran setiap dimensi suatu array atau tabel.	Diperiksa untuk mencegah ketidakcocokan data dan kesalahan operasi numerik.
SHA-256	Algoritma hash yang menghasilkan nilai pemeriksaan 256-bit.	Digunakan untuk membuat checksum artefak dalam manifest hasil.
SMILES	Notasi linear untuk merepresentasikan struktur dan konektivitas molekul.	Digunakan RDKit sebagai input pembuatan molekul dan konformer.
Spin PySCF	Selisih jumlah elektron alfa dan beta.	Dalam PySCF berlaku <code>spin = multiplicity - 1</code> , bukan nilai <code>multiplicity</code> itu sendiri.
State kernel	Kumpulan variabel, objek, dan impor yang tersimpan dalam kernel.	State tersembunyi dapat membuat notebook gagal ketika dijalankan ulang dari awal.

STO-3G	Basis set minimal yang menggunakan tiga fungsi Gaussian untuk setiap orbital basis.	Cocok untuk demonstrasi, tetapi memiliki keterbatasan akurasi.
Struktur kontrol	Konstruksi program untuk mengatur alur eksekusi.	Meliputi percabangan, perulangan, dan penghentian proses berdasarkan kondisi.

T

Istilah	Definisi singkat	Penjelasan kontekstual dan padanan
Tensor	Array multidimensi yang digunakan dalam komputasi numerik.	Operasi tensor dapat dipercepat dengan pustaka GPU seperti cuTENSOR.
Test case	Lihat Kasus uji .	Digunakan untuk memeriksa masukan valid dan tidak valid.
Titik stasioner	Geometri ketika gradien energi mendekati nol.	Dapat berupa minimum, maksimum, atau titik pelana; frekuensi membantu membedakannya.
TPU	Perangkat akselerator yang dirancang terutama untuk operasi tensor pembelajaran mesin.	Tidak otomatis memberikan akselerasi untuk perhitungan PySCF. <i>Tensor Processing Unit</i> .
Troubleshooting	Proses sistematis untuk menemukan dan memperbaiki kesalahan.	Buku menekankan satu perubahan pada satu waktu, penyimpanan pesan error, dan pengujian ulang.
TXT	Format berkas teks sederhana.	Digunakan untuk log, README, ringkasan hasil, dan dokumentasi proses.
Type hint	Anotasi tipe pada parameter dan keluaran fungsi Python.	Membantu memperjelas penggunaan fungsi dan memudahkan pemeriksaan kode.

Reproduksibilitas, validasi input, status eksekusi, pemeriksaan NaN/inf, penyimpanan keluaran, dan pembatasan interpretasi menjadi kriteria utama notebook dalam buku. Buku juga menekankan bahwa hasil tidak boleh bergantung pada variabel tersembunyi atau state kernel lama.

U

Istilah	Definisi singkat	Penjelasan kontekstual dan padanan
UFF	Medan gaya universal untuk optimasi klasik berbagai unsur.	Kepanjangan: <i>Universal Force Field</i> . Digunakan jika parameter MMFF94 tidak tersedia.

UHF	Metode Hartree–Fock tak terbatas untuk sistem <i>open-shell</i> .	Orbital elektron alfa dan beta dapat memiliki bentuk berbeda.
UKS	Metode Kohn–Sham tak terbatas untuk sistem <i>open-shell</i> .	Digunakan untuk perhitungan DFT ketika terdapat elektron tidak berpasangan.
Unit test	Pengujian terhadap satu fungsi atau bagian kecil program.	Digunakan untuk memastikan fungsi konversi atau perhitungan menghasilkan nilai yang diharapkan.
Unggah	Proses memindahkan berkas dari komputer pengguna ke runtime.	Dilakukan melalui antarmuka Colab atau <code>files.upload()</code> . Padanan: <i>upload</i> .
Urutan eksekusi	Susunan pelaksanaan sel notebook.	Sel harus dijalankan dari atas ke bawah agar dependensi dan variabel tersedia secara benar.
UTF-8	Standar encoding karakter yang mendukung banyak sistem tulisan.	Digunakan untuk menyimpan dan membaca berkas teks secara konsisten.

V

Istilah	Definisi singkat	Penjelasan kontekstual dan padanan
Validasi	Pemeriksaan sistematis terhadap input, proses, format, numerik, kimia, dan keluaran.	Menjadi unsur inti setiap notebook dan proyek dalam buku. Padanan: <i>validation</i> .
Validasi kimia	Pemeriksaan kewajaran sistem berdasarkan prinsip kimia.	Mencakup unsur, muatan, jumlah elektron, spin, multiplicity, ikatan, dan geometri.
Validasi numerik	Pemeriksaan kualitas dan konsistensi data bilangan.	Mencakup NaN, inf, bentuk array, jumlah atom, satuan, dan ambang konvergensi.
Variabel	Nama yang merujuk pada nilai atau objek dalam program.	Harus diberi nama bermakna dan tidak bergantung pada state tersembunyi.
Vectorization	Penggunaan operasi array langsung sebagai pengganti perulangan elemen demi elemen.	Digunakan dalam NumPy untuk meningkatkan keterbacaan dan efisiensi. Padanan: vektorisasi.
Verifikasi eksternal	Pemeriksaan hasil menggunakan sumber atau perangkat lain di luar notebook.	Diperlukan untuk hasil yang belum dapat dipastikan hanya dari eksekusi program.

Versi pustaka	Nomor rilis paket perangkat lunak yang digunakan.	Harus dicatat karena perubahan versi dapat memengaruhi sintaks dan hasil komputasi.
Visualisasi molekul	Penyajian struktur molekul dalam bentuk grafis dua atau tiga dimensi.	Buku menggunakan py3Dmol untuk menampilkan struktur awal dan hasil optimasi.

W

Istilah	Definisi singkat	Penjelasan kontekstual dan padanan
Warning	Pesan peringatan yang tidak selalu menghentikan program.	Harus dibaca dan tidak boleh diabaikan apabila memengaruhi data atau interpretasi.
Workflow	Urutan kerja dari input sampai penyimpanan hasil.	Buku menerapkan alur input–proses–validasi–interpretasi–ekspor. Padanan: alur kerja.
Working directory	Folder aktif tempat program membaca dan menulis berkas secara default.	Diperiksa saat terjadi <code>FileNotFoundError</code> atau masalah jalur berkas.
Workspace	Ruang kerja yang mencakup notebook, berkas, variabel, dan sumber daya runtime.	Dalam Colab bersifat sementara kecuali hasil disimpan ke lokasi persisten.

X

Istilah	Definisi singkat	Penjelasan kontekstual dan padanan
X-axis	Sumbu horizontal pada sebuah grafik.	Harus diberi label dan satuan yang sesuai ketika menggunakan Matplotlib. Padanan: sumbu-X.
XYZ	Format teks sederhana untuk menyimpan simbol atom dan koordinat tiga dimensi.	Baris pertama berisi jumlah atom, diikuti komentar serta simbol dan koordinat (x,y,z).

Y

Istilah	Definisi singkat	Penjelasan kontekstual dan padanan
Y-axis	Sumbu vertikal pada sebuah grafik.	Digunakan untuk menampilkan variabel respons seperti energi, frekuensi, atau nilai pengamatan. Padanan: sumbu-Y.
Yield eksekusi	Hasil yang benar-benar diperoleh setelah kode dijalankan.	Dalam konteks buku, hasil harus dibedakan dari keluaran lama atau hasil yang belum diuji ulang.

Z

Istilah	Definisi singkat	Penjelasan kontekstual dan padanan
Zero-point energy	Energi kuantum minimum yang tetap ada pada keadaan vibrasi dasar.	Relevan dalam analisis frekuensi dan koreksi energi molekul. Padanan: energi titik nol.
ZIP	Format arsip yang menggabungkan dan mengompresi sejumlah berkas.	Digunakan untuk mengemas notebook, grafik, data, metadata, dan laporan sebelum runtime berakhir.

- KASMUI -
Dosen Kimia, Komputasi, IT, AI UNNES

<https://hisabmu.com/>
<https://kasmui.cloud/>

Ketua PCM
Gunungpati 2

Anggota Majelis Tabligh
PDM Kota Semarang
& PWM Jawa Tengah

Tim Pengembang
Software KHGT
MTT PP Muhammadiyah

Praktisi
Ilmu Falak


KASMUI



Tutorial **Menjalankan** **Notebook Python** di **Google Colab**

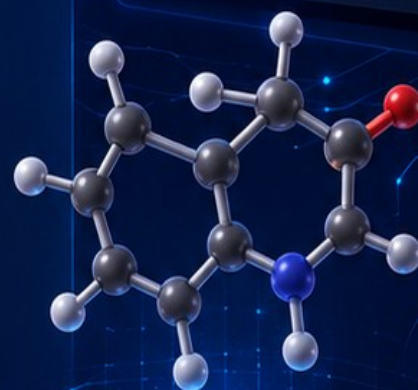


SINOPSIS

Buku ini dirancang sebagai panduan praktis untuk mempelajari cara menjalankan notebook Python di Google Colab, mulai dari tingkat dasar hingga penerapan yang lebih ilmiah. Pembaca diajak mengenal antarmuka Colab, pengelolaan sel kode dan Markdown, runtime, pengelolaan berkas, serta dasar-dasar Python, NumPy, Pandas, dan Matplotlib dengan langkah yang sistematis dan mudah diikuti.

Selain membahas penggunaan Google Colab untuk komputasi umum, buku ini juga memperkenalkan penerapan notebook pada visualisasi molekul tiga dimensi, pembuatan konformer, optimasi geometri, dan analisis sifat molekul menggunakan RDKit, py3Dmol, PySCF, geomeTRIC, hingga GPU4PySCF. Setiap pembahasan dilengkapi penekanan pada validasi, reproduksibilitas, troubleshooting, serta penyimpanan hasil agar pekerjaan ilmiah tetap rapi, transparan, dan dapat diulang.

Dengan cakupan lima bab dan berbagai lampiran praktis, buku ini cocok untuk mahasiswa, dosen, guru kimia, peneliti pemula, dan siapa pun yang ingin memanfaatkan Google Colab sebagai sarana belajar Python dan komputasi ilmiah secara efektif.



KASMUI