Bab 03 — Python, Data, dan Notebook Reproducible

Status: Draft lengkap v0.3 untuk review editorial Bagian buku: Level B — Fondasi Data dan Eksperimen Target pembaca: Pemula yang ingin bisa menjalankan praktikum AI dengan Python, file `.py`, Jupyter Notebook, Google Colab, dan Kaggle Notebook.

Cara membaca bab ini

Bab 1–2 memberi peta konsep. Bab 3 memberi meja kerja. Mulai bab ini, pembaca tidak hanya membaca istilah, tetapi menjalankan eksperimen kecil yang bisa diulang.

Bab ini memakai 12 subbab belajar. Setiap subbab punya ilustrasi, catatan kata kunci, dan tes cepat. Fokusnya bukan menjadi programmer Python expert dalam satu bab, tetapi cukup kuat untuk menjalankan lab AI berikutnya.

> Catatan biru — tujuan bab:
> Setelah bab ini, kamu bisa menjalankan kode lewat terminal/VS Code, membuka notebook lokal, memakai Colab/Kaggle, membaca dataset kecil, membuat analisis sederhana, dan menyimpan eksperimen agar bisa diulang.

Pendalaman awal — Python sebagai alat berpikir komputasional

Python bukan hanya bahasa pemrograman populer; dalam AI, Python menjadi buku catatan, kalkulator, laboratorium data, dan jembatan ke library besar seperti NumPy, pandas, scikit-learn, PyTorch, dan TensorFlow. Namun sebelum library besar, pembaca perlu memahami inti komputasi: variabel, tipe data, struktur data, loop, fungsi, file, dan reproducibility.

Komputasi data sederhana dapat dibaca sebagai pipeline:

data mentah → parsing → cleaning → transformasi → analisis → baseline → catatan hasil

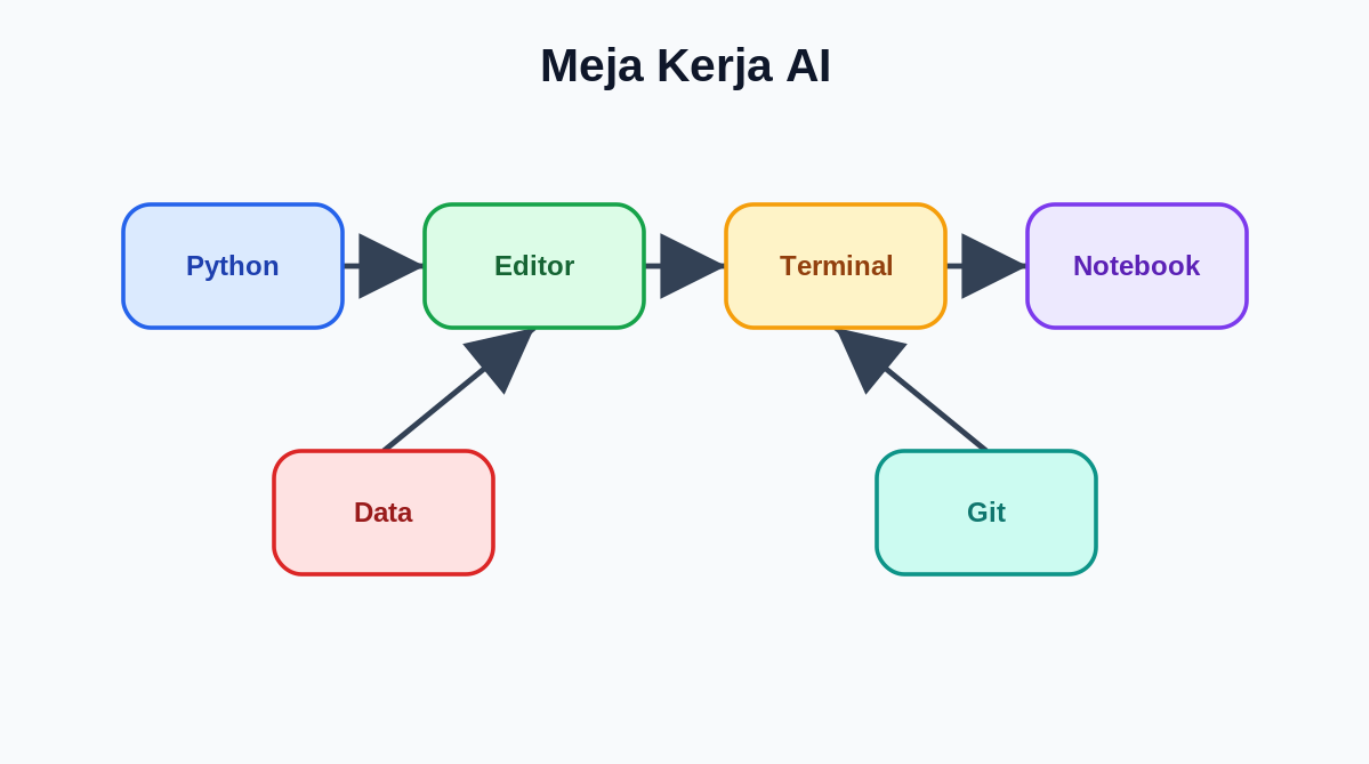
Setiap panah adalah operasi yang bisa diuji. Jika salah satu langkah tidak jelas, hasil AI ikut tidak jelas. Karena itu Bab 3 diperdalam bukan untuk membuat pembaca menjadi software engineer penuh, tetapi agar pembaca bisa membaca, mengetik, menjalankan, mengubah, dan memeriksa eksperimen AI.

Kompleksitas ringan

Loop satu kali atas n data biasanya $O(n)$. Loop di dalam loop bisa $O(n^2)$. Untuk dataset kecil ini tidak masalah, tetapi kebiasaan berpikir kompleksitas akan berguna saat data membesar.

Subbab 1 — Meja kerja AI: bukan cuma model

Banyak pemula ingin langsung membuat model. Itu wajar. Tetapi di proyek AI nyata, model hanya satu bagian kecil. Kita butuh lingkungan kerja: Python, editor, terminal, notebook, folder rapi, data kecil, catatan eksperimen, dan cara menjalankan ulang hasil.



Meja kerja AI

Bayangkan membuka warung. Kamu tidak langsung memasak tanpa meja, kompor, bahan, catatan stok, dan alat timbang. AI juga begitu. Python adalah bahasa kerja, dataset adalah bahan, notebook adalah buku catatan eksperimen, file `.py` adalah resep yang bisa dijalankan ulang, dan Git adalah riwayat perubahan.

> Catatan hijau — kata kunci: Reproducible
> Reproducible berarti orang lain, atau kamu sendiri minggu depan, bisa menjalankan ulang langkah yang sama dan mendapatkan hasil yang masuk akal.

Komponen meja kerja:

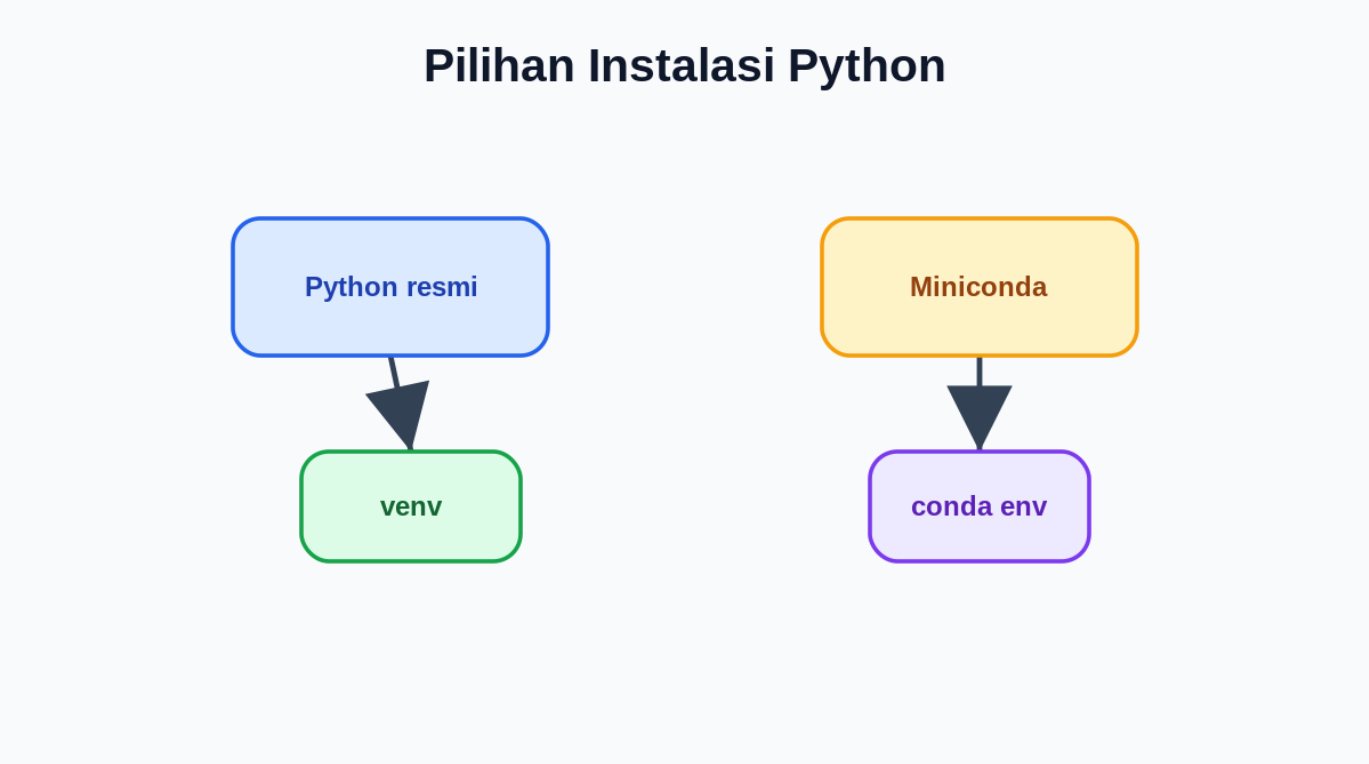
Komponen	Fungsi	Contoh alat
Python	Menjalankan logika dan analisis	Python 3.10+
Editor	Menulis file `.py` dan markdown	VS Code, PyCharm, Vim
Terminal	Menjalankan perintah	WSL, PowerShell, Terminal Linux
Notebook	Eksperimen interaktif	Jupyter, Colab, Kaggle
Git	Riwayat perubahan	git commit, GitHub
Folder proyek	Kerapian aset	`data/`, `code/`, `figures/`

Tes cepat subbab 1

1. Apa bedanya file `.py` dan notebook secara fungsi?
2. Mengapa AI membutuhkan folder proyek yang rapi?
3. Apa arti reproducible dengan bahasa sendiri?

Subbab 2 — Instalasi Python yang aman untuk pemula

Untuk buku ini, gunakan Python 3.10 atau lebih baru. Di WSL/Linux, biasanya Python sudah tersedia. Di Windows, Python bisa dipasang dari python.org atau lewat Anaconda/Miniconda. Untuk pembaca pemula, dua jalur paling praktis:



Pilihan instalasi Python

Jalur A — Python resmi + virtual environment

```
python3 --version
python3 -m venv .venv
source .venv/bin/activate
python -m pip install --upgrade pip
```

Di Windows PowerShell:

```
py -3 --version
py -3 -m venv .venv
.\.venv\Scripts\Activate.ps1
python -m pip install --upgrade pip
```

Jalur B — Miniconda/Anaconda

```
conda create -n zth-ai python=3.11
conda activate zth-ai
```

> Catatan kuning — kata kunci: Environment
> Environment adalah ruang kerja Python terpisah. Tujuannya agar paket proyek ini tidak bentrok dengan proyek lain.

Untuk bab awal, kode dibuat memakai standard library agar bisa jalan tanpa instalasi paket berat. Nanti saat masuk NumPy, pandas, scikit-learn, atau deep learning, environment menjadi lebih penting.

Tes cepat subbab 2

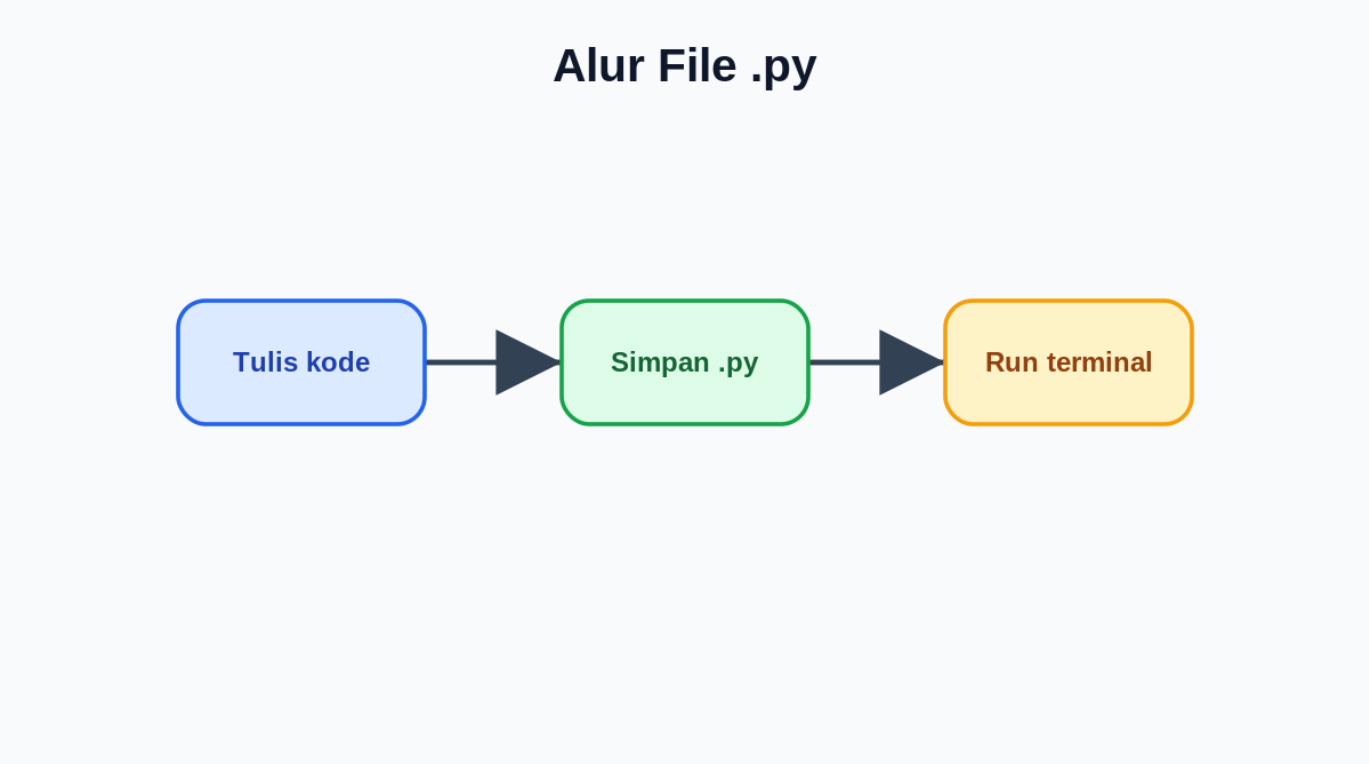
1. Mengapa sebaiknya memakai virtual environment?

2. Apa perintah untuk mengecek versi Python?

3. Apa risiko menginstal semua paket di Python global?

Subbab 3 — File `.py`: resep yang mudah dijalankan ulang

File `.py` cocok untuk kode yang ingin dijalankan berulang, dites, dan disimpan sebagai bagian proyek. Editor seperti VS Code nyaman untuk menulis `.py` karena punya syntax highlighting, terminal, linting, dan integrasi Git.



Alur menjalankan file py

Contoh menjalankan file Bab 3:

```
cd zero-to-hero-menaklukkan-ai/chapters/03-python-data-lab/code
python3 python_data_lab.py
```

Di Windows, bisa juga:

```
cd zero-to-hero-menaklukkan-ai\chapters\03-python-data-lab\code
py python_data_lab.py
```

Struktur sederhana file `.py`:

```
def main():
    print("Halo, eksperimen AI!")

if __name__ == "__main__":
    main()
```

Bagian `if __name__ == "__main__"` berarti: jalankan `main()` hanya ketika file dieksekusi langsung, bukan ketika diimpor oleh file lain.

Latihan ketik manual 3A — variabel, `print`, dan f-string

Ketik contoh berikut di file baru bernama `latihan_01_variabel.py`:

```
nama_warung = "Warung Maju"
es_teh = 42
kopi = 18
total = es_teh + kopi

print("Nama warung:", nama_warung)
print("Total minuman:", total)
print(f"Hari ini {nama_warung} menjual {total} minuman.")
```

Jalankan:

```
python3 latihan_01_variabel.py
```

Yang perlu diperhatikan:

- `nama_warung` berisi teks/string.
- `es_teh`, `kopi`, dan `total` berisi angka.
- `f"...{total}..."` disebut f-string, yaitu cara mudah memasukkan nilai variabel ke kalimat.

Latihan ketik manual 3B — input sederhana

Ketik file `latihan_02_input.py`:

```
nama = input("Nama kamu: ")
target = input("Mau belajar apa hari ini? ")
print(f"Halo {nama}, hari ini kita belajar {target}.")
```

Latihan ini belum AI, tetapi penting. Banyak aplikasi AI tetap dimulai dari input pengguna, memproses data, lalu menampilkan output.

> Catatan biru — kata kunci: Script
> Script `.py` adalah resep. Notebook adalah buku catatan interaktif. Keduanya penting dan saling melengkapi.

Model mental file `.py`

File `.py` adalah resep eksperimen. Jika resepnya sama, inputnya sama, dan environment-nya sama, hasil seharusnya sama. Ini dasar reproducibility.

```
script.py + data + seed + versi Python = eksperimen yang bisa diulang
```

Contoh struktur script sehat

```
def main():
    data = [10, 20, 30]
    rata = sum(data) / len(data)
    print(rata)

if __name__ == "__main__":
    main()
```

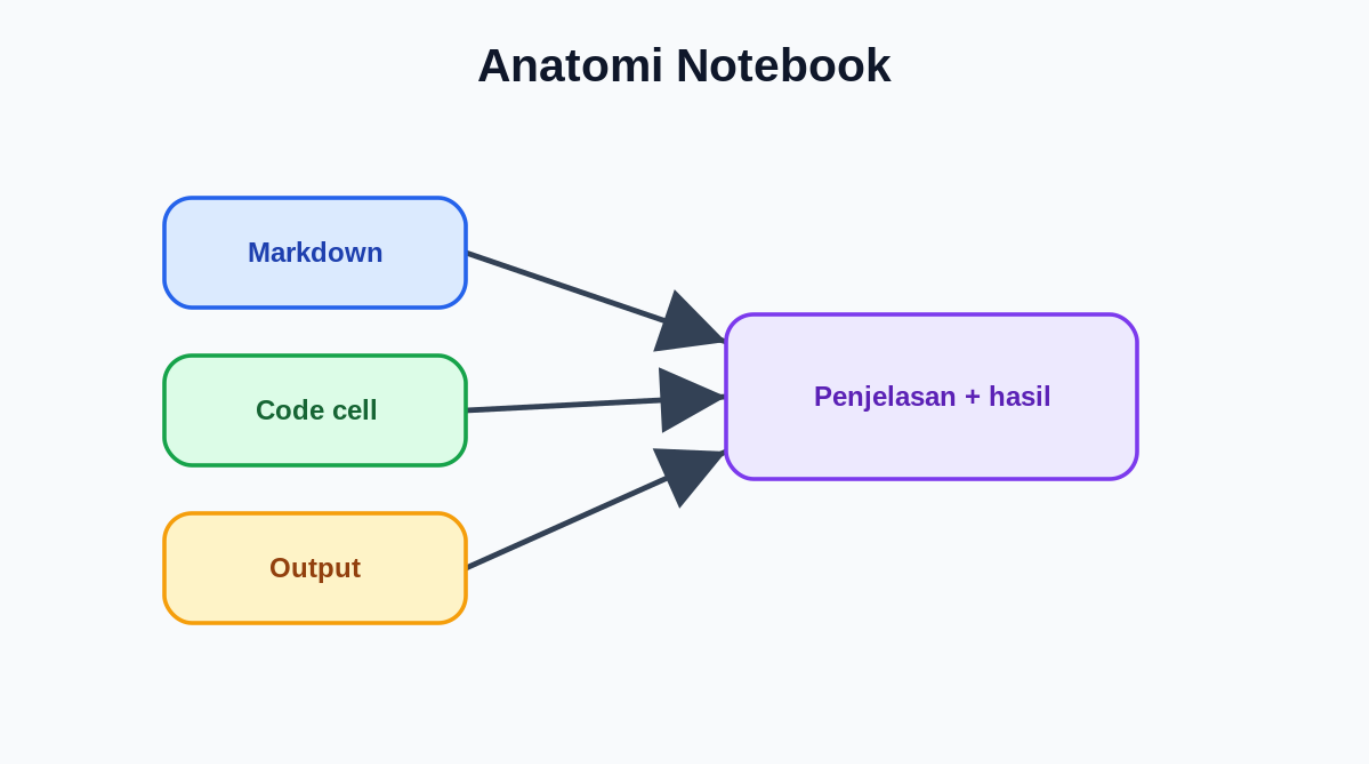
Blok `if __name__ == "__main__"` membuat file bisa dijalankan langsung, tetapi fungsi-fungsinya juga bisa diimport oleh notebook atau test.

Tes cepat subbab 3

1. Kapan lebih cocok memakai file `.py` daripada notebook?
2. Apa fungsi `main()` dalam script?
3. Jalankan `python_data_lab.py` dan tulis output yang muncul.

Subbab 4 — Jupyter Notebook: tempat berpikir sambil bereksperimen

Jupyter Notebook memungkinkan kita menulis teks, kode, output, tabel, dan grafik dalam satu dokumen `.ipynb`. Ini sangat cocok untuk eksplorasi data, pembelajaran matematika, dan praktikum AI.



Anatomi notebook

Instalasi lokal minimal:

```
python3 -m venv .venv
source .venv/bin/activate
python -m pip install notebook
jupyter notebook
```

Alternatif modern:

```
python -m pip install jupyterlab
jupyter lab
```

Kelebihan notebook:

- nyaman untuk mencoba kode sel demi sel,
- teks penjelasan dan output menyatu,
- bagus untuk laporan eksperimen,
- mudah dibuka di Jupyter, Colab, atau Kaggle.

Kekurangan notebook:

- mudah berantakan bila sel dijalankan tidak berurutan,
- hasil bisa tidak reproducible jika state tersembunyi,
- kurang ideal untuk kode produksi panjang,
- file JSON `.ipynb` sulit dibaca di diff Git.

Catatan merah — kata kunci: Run All
Sebelum menganggap notebook selesai, restart kernel lalu jalankan semua sel dari atas. Jika gagal, notebook belum reproducible.

Tes cepat subbab 4

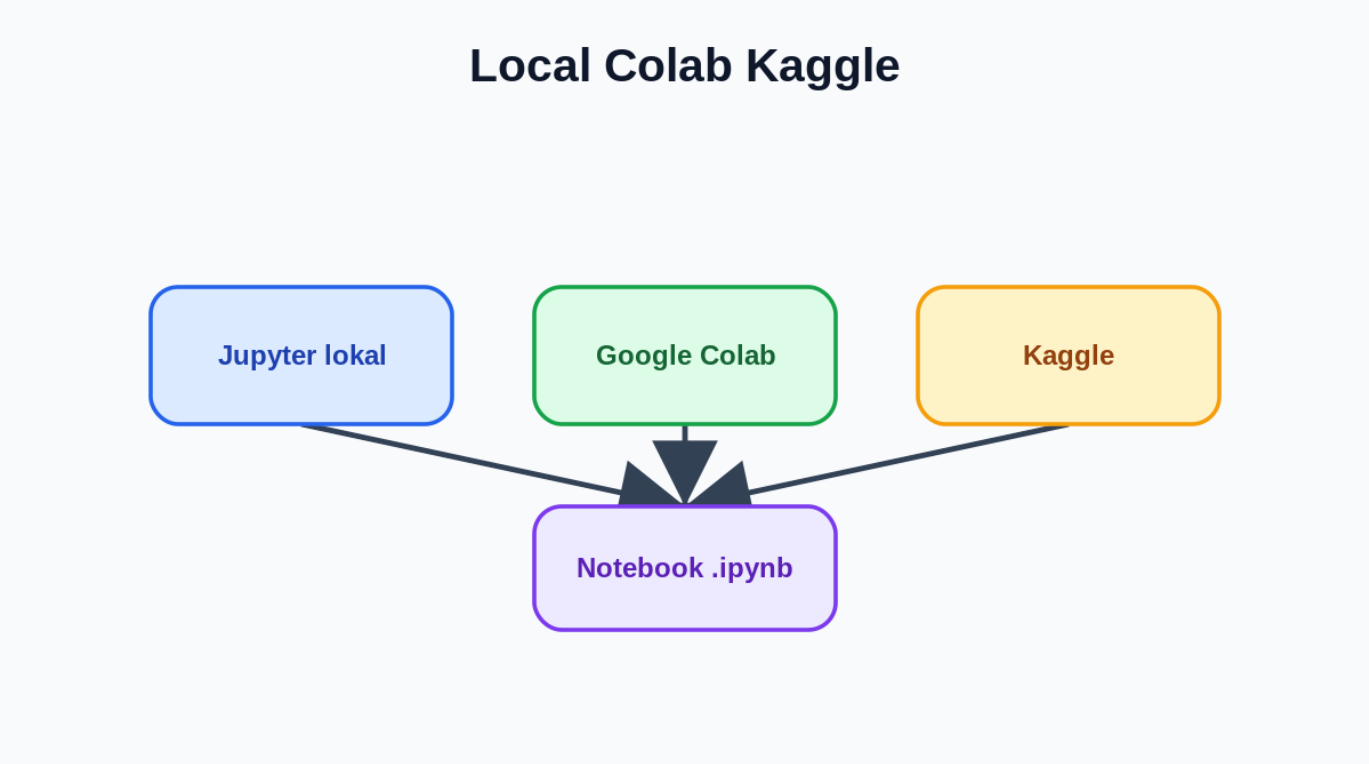
1. Sebutkan dua kelebihan notebook.

2. Sebutkan dua kekurangan notebook.

3. Mengapa “Restart & Run All” penting?

Subbab 5 — Google Colab dan Kaggle Notebook

Tidak semua pembaca ingin memasang Jupyter lokal. Google Colab dan Kaggle Notebook memungkinkan menjalankan notebook dari browser. Keduanya berguna untuk belajar karena lingkungan sudah siap dan bisa memakai GPU pada kondisi tertentu.



Colab Kaggle Local

Google Colab cocok untuk:

- belajar cepat tanpa setup lokal,
- berbagi notebook lewat link,
- menjalankan demo dengan GPU/TPU terbatas,
- membuka file `.ipynb` dari GitHub atau upload manual.

Kaggle Notebook cocok untuk:

- belajar dengan dataset Kaggle,
- eksperimen kompetisi,
- GPU gratis terbatas,
- menyimpan notebook bersama dataset.

Jupyter lokal cocok untuk:

- data pribadi yang tidak boleh diupload,
- kontrol penuh environment,
- bekerja offline,
- integrasi dengan folder proyek lokal.

Catatan ungu — kata kunci: Pilih Platform
Colab/Kaggle praktis untuk belajar, tetapi jangan upload data pribadi/sensitif sembarangan.

Cara memakai notebook Bab 3:

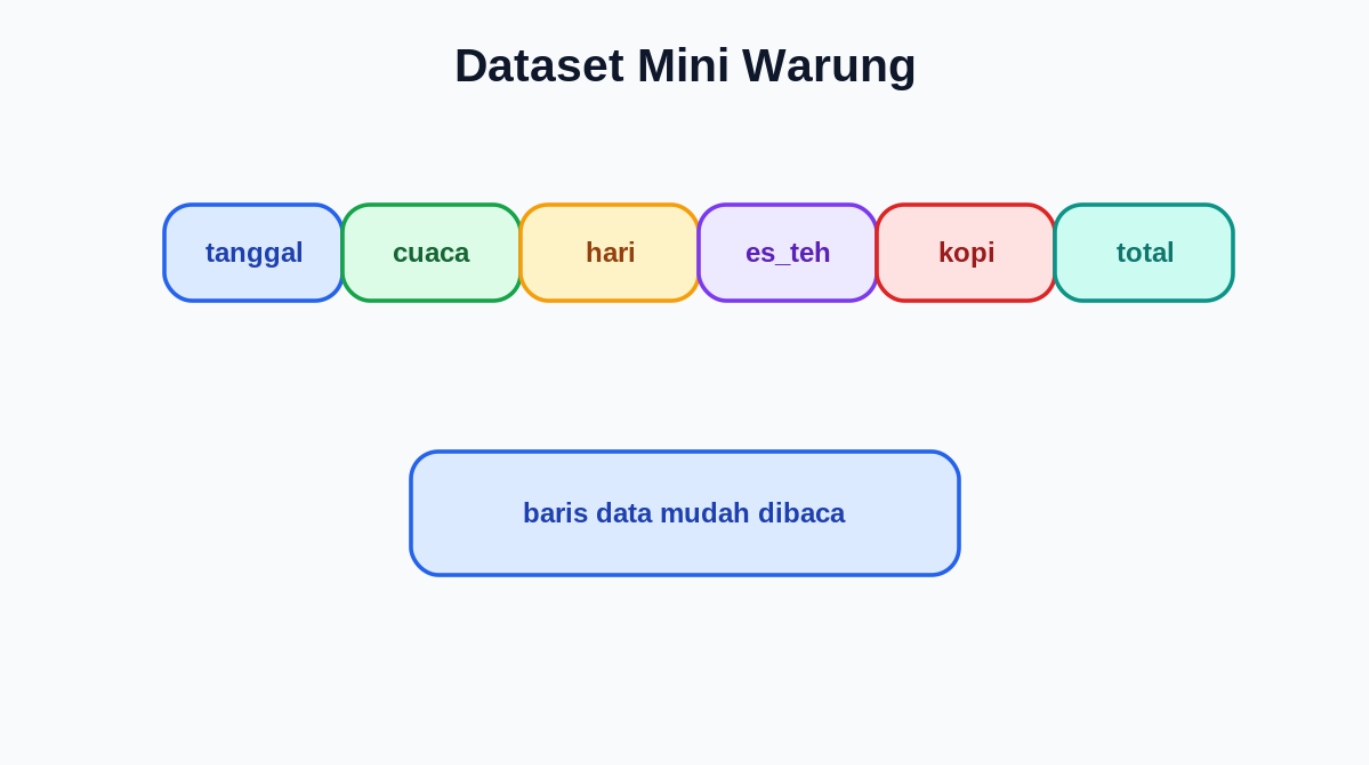
1. Lokal: buka `python_data_lab.ipynb` dengan Jupyter/VS Code.
2. Colab: upload file `.ipynb` atau buka dari GitHub setelah repo publik.
3. Kaggle: buat notebook baru, upload `.ipynb`, lalu jalankan sel dari atas.

Tes cepat subbab 5

1. Kapan Colab lebih nyaman daripada Jupyter lokal?
2. Kapan Jupyter lokal lebih aman?
3. Mengapa data sensitif tidak boleh sembarang diupload?

Subbab 6 — Dataset mini: data warung yang bisa dibaca manusia

Untuk praktikum awal, kita pakai dataset mini penjualan minuman warung. Dataset kecil membuat konsep mudah dilihat tanpa library berat.



Dataset mini warung

Contoh baris data:

tanggal	cuaca	hari	es_teh	kopi	total
2026-01-01	panas	libur	42	18	60
2026-01-02	hujan	kerja	18	31	49

Dalam script Bab 3, dataset disimpan sebagai list of dict:

```
{"tanggal": "2026-01-01", "cuaca": "panas", "hari": "libur", "es_teh": 42, "kopi": 18}
```

Mengapa bukan langsung pandas? Karena kita ingin melihat konsep dasar: list, dict, loop, fungsi, agregasi, dan rata-rata. Setelah konsep kuat, pandas akan terasa seperti alat bantu, bukan sihir.

Latihan ketik manual 6A — membuat dataset sendiri

Ketik di file `latihan_03_dataset.py`:

```
data = [
    {"tanggal": "2026-01-01", "cuaca": "panas", "es_teh": 42, "kopi": 18},
    {"tanggal": "2026-01-02", "cuaca": "hujan", "es_teh": 18, "kopi": 31},
]

print(data)
print(data[0])
print(data[0]["es_teh"])
```

Tambahkan satu baris baru:

```
data.append({"tanggal": "2026-01-03", "cuaca": "panas", "es_teh": 35, "kopi": 22})
print(data)
```

Pertanyaan latihan:

1. Apa arti `data[0]`?

2. Apa arti `data[0]["es_teh"]`?

3. Mengapa setiap baris memakai key yang sama?

Catatan hijau — kata kunci: Data Kecil
Data kecil bukan mainan. Data kecil membantu kita memeriksa logika sebelum memakai dataset besar.

Dataset sebagai list of dict

Dalam Python standard library, dataset tabular kecil bisa ditulis sebagai list of dict:

```
data = [
    {"hari": "Senin", "penjualan": 30, "hujan": 0},
    {"hari": "Selasa", "penjualan": 18, "hujan": 1},
]
```

Setiap dict adalah satu baris. Key adalah nama kolom. Value adalah isi sel.

Contoh hitung manual

Jika penjualan `[30,18,42]`:

```
sum = 30+18+42 = 90
mean = 90/3 = 30
max = 42
min = 18
range = 42-18 = 24
```

Python membantu menghitung, tetapi pembaca tetap harus tahu makna angka.

Tes cepat subbab 6

1. Mengapa dataset kecil berguna untuk belajar?

2. Apa itu list of dict?

3. Tambahkan satu baris data baru ke dataset Bab 3.

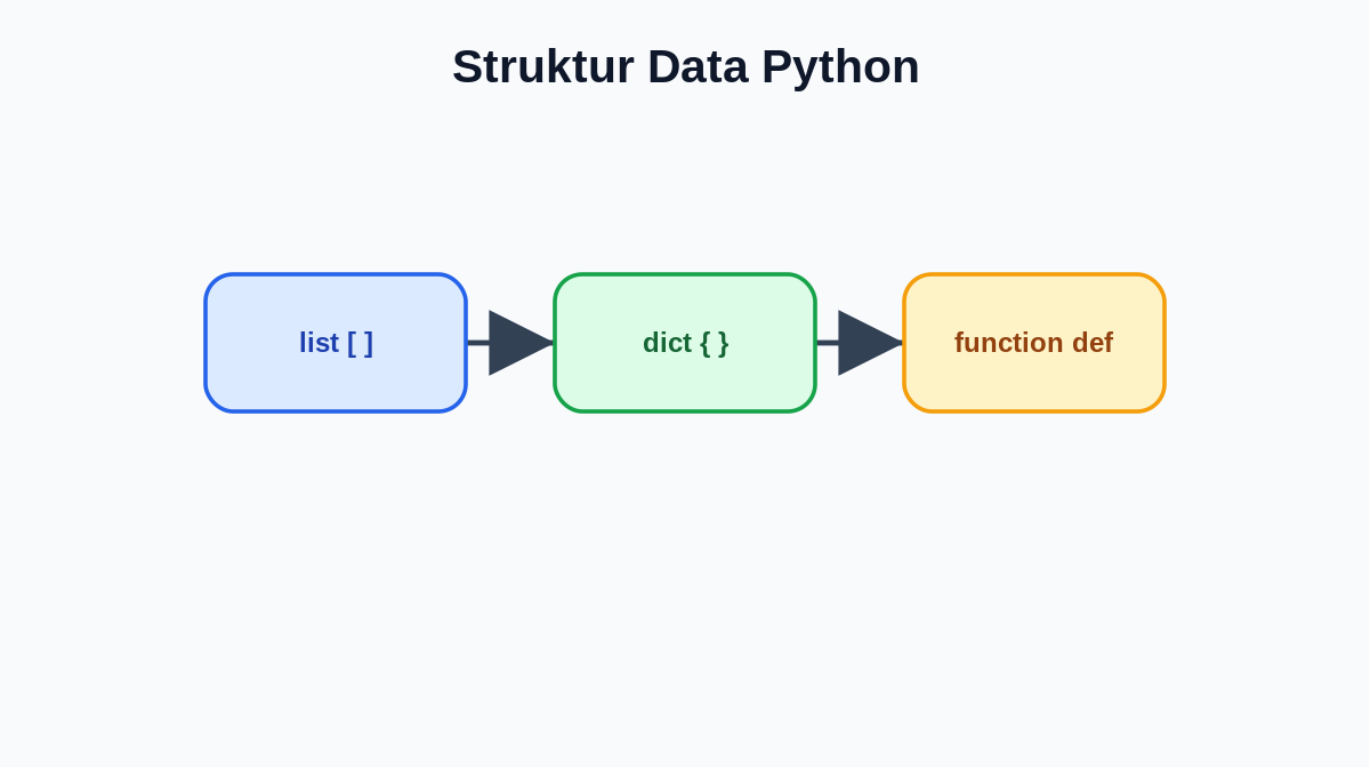
Subbab 7 — Operasi dasar Python untuk data

Tiga struktur Python yang sering dipakai di awal:

• list: kumpulan item berurutan,

• dict: pasangan key-value,

• function: blok kode yang bisa dipanggil ulang.



Struktur data Python

Contoh:

```
penjualan = [42, 18, 35]
rata_rata = sum(penjualan) / len(penjualan)
```

Contoh fungsi:

```
def rata_rata(nilai):
    return sum(nilai) / len(nilai)
```

Latihan ketik manual 7A — loop untuk membaca data

Ketik contoh berikut:

```
data = [
    {"tanggal": "2026-01-01", "es_teh": 42, "kopi": 18},
    {"tanggal": "2026-01-02", "es_teh": 18, "kopi": 31},
    {"tanggal": "2026-01-03", "es_teh": 35, "kopi": 22},
]

for baris in data:
    total = baris["es_teh"] + baris["kopi"]
    print(baris["tanggal"], total)
```

`for baris in data` berarti: ambil satu baris, kerjakan isi loop, lalu lanjut ke baris berikutnya.

Latihan ketik manual 7B — fungsi kecil untuk total

```
def total_minuman(baris):
    return baris["es_teh"] + baris["kopi"]

for baris in data:
    print(baris["tanggal"], total_minuman(baris))
```

Dalam analisis data, fungsi membantu kita menghindari copy-paste. Jika perhitungan salah, kita memperbaiki satu tempat, bukan banyak tempat.

> Catatan kuning — kata kunci: Fungsi
> Fungsi adalah kebiasaan baik sejak awal. AI yang rapi dimulai dari kode kecil yang rapi.

Loop sebagai notasi sigma versi Python

Rumus jumlah:

$$\Sigma x_i$$

Dalam Python:

```
total = 0
for x in data:
    total += x
```

Rumus mean:

$$\text{mean} = (\Sigma x_i) / n$$

Dalam Python:

```
mean = total / len(data)
```

Memahami hubungan rumus dan kode membuat pembaca tidak hanya menyalin syntax.

Tes cepat subbab 7

1. Apa perbedaan list dan dict?

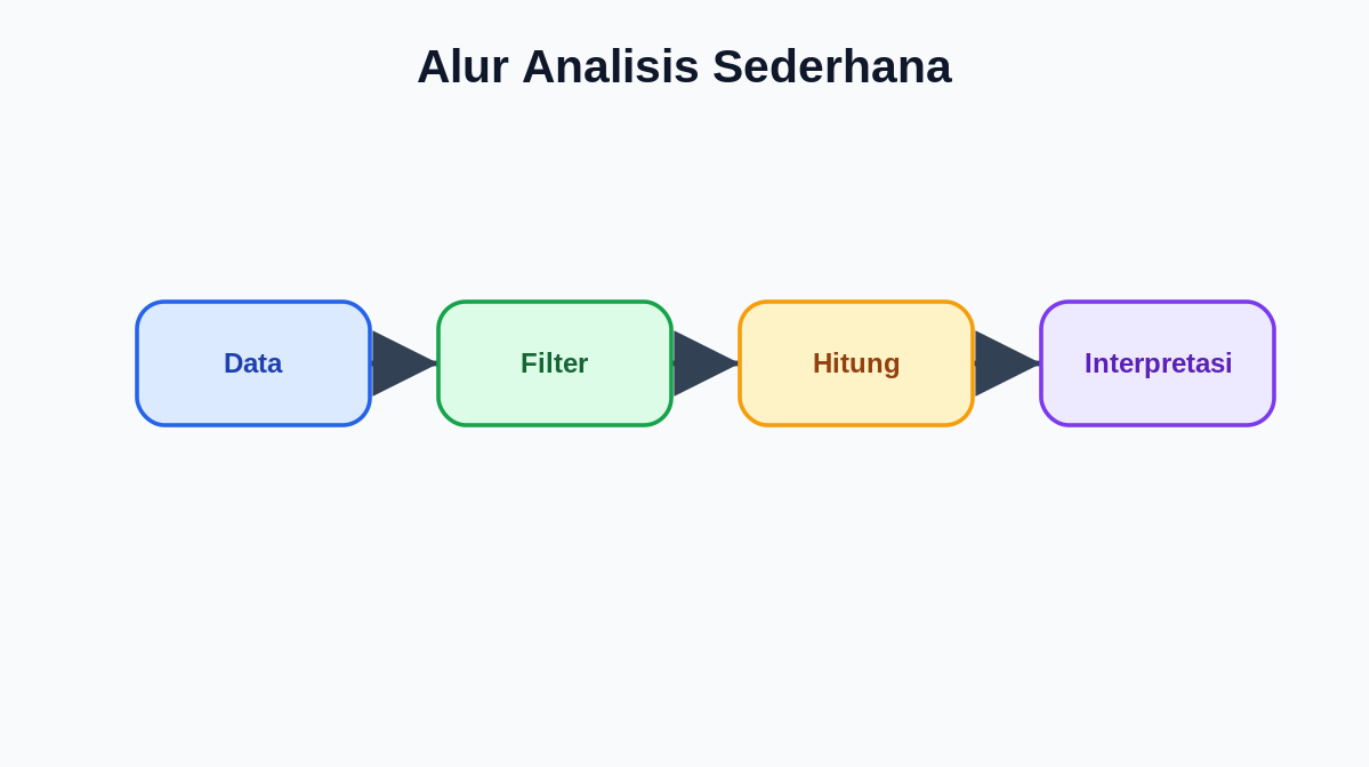
2. Mengapa fungsi mengurangi copy-paste?

3. Buat fungsi `total_minuman(es_teh, kopi)`.

Subbab 8 — Analisis sederhana: rata-rata, maksimum, dan segmentasi

Sebelum model, kita harus mengenali data. Pertanyaan sederhana:

- Rata-rata penjualan es teh berapa?
- Hari apa penjualan paling tinggi?
- Apakah cuaca panas meningkatkan es teh?
- Apakah hari libur berbeda dari hari kerja?



Analisis sederhana

Script Bab 3 menghitung ringkasan:

```
Rata-rata es teh: ...
Rata-rata kopi: ...
Hari total tertinggi: ...
Rata-rata es teh saat panas: ...
```

Latihan ketik manual 8A — menghitung rata-rata tanpa library

```
es_teh = [42, 18, 35, 30, 16]
rata = sum(es_teh) / len(es_teh)
print(rata)
```

Latihan ketik manual 8B — filter data panas

```
data = [
    {"cuaca": "panas", "es_teh": 42},
    {"cuaca": "hujan", "es_teh": 18},
    {"cuaca": "panas", "es_teh": 35},
]

es_teh_panas = []
for baris in data:
    if baris["cuaca"] == "panas":
        es_teh_panas.append(baris["es_teh"])

print(es_teh_panas)
```

```
print(sum(es_teh_panas) / len(es_teh_panas))
```

Ini belum Machine Learning, tetapi sangat penting. Banyak proyek gagal karena orang melatih model sebelum memahami data.

> Catatan biru — kata kunci: EDA
> EDA atau Exploratory Data Analysis adalah proses bertanya kepada data sebelum membuat model.

Contoh segmentasi manual

Data penjualan:

```
penjualan = [18, 30, 42, 35]
```

Mean:

```
(18+30+42+35)/4 = 125/4 = 31,25
```

Segmentasi sederhana:

```
jika penjualan >= mean → ramai
jika penjualan < mean → sepi
```

Hasil:

```
18 sepi, 30 sepi, 42 ramai, 35 ramai
```

Ini adalah bentuk awal rule-based classifier. Nanti Bab 6 dan 7 membuatnya lebih formal.

Tes cepat subbab 8

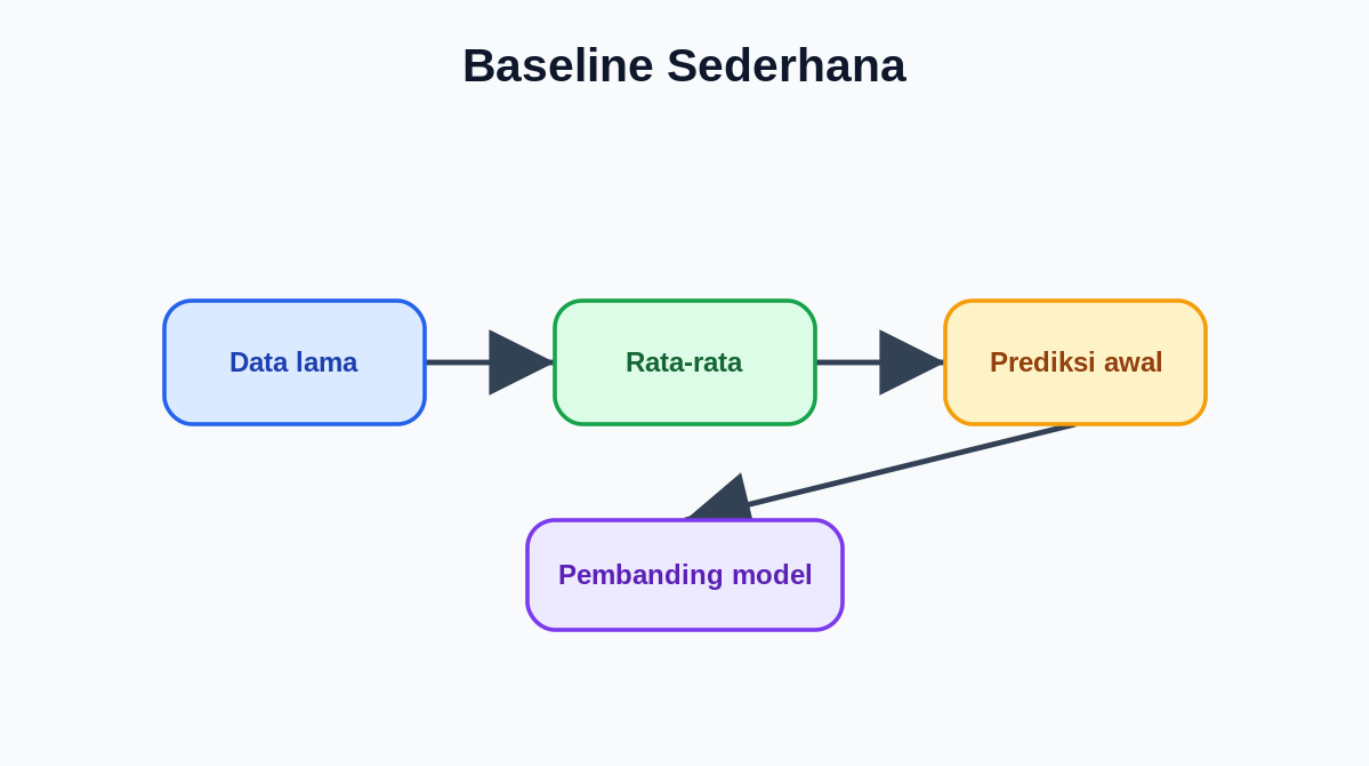
1. Apa tujuan EDA?

2. Mengapa rata-rata saja kadang tidak cukup?

3. Apa pertanyaan EDA tambahan untuk dataset warung?

Subbab 9 — Baseline: model sederhana sebagai lawan tanding

Baseline adalah solusi sederhana yang menjadi pembanding. Untuk prediksi stok es teh, baseline bisa berupa rata-rata 7 hari terakhir atau nilai hari sebelumnya. Jika model canggih tidak mengalahkan baseline, model itu belum berguna.



Baseline sederhana

Contoh baseline:

```
prediksi_besok = round(rata_rata([baris["es_teh"] for baris in data]))
```

Latihan ketik manual 9A — baseline dari rata-rata

Ketik pelan-pelan:

```
data_es_teh = [42, 18, 35, 30, 16, 38, 45]
rata = sum(data_es_teh) / len(data_es_teh)
prediksi_besok = round(rata)
print("Prediksi es teh besok:", prediksi_besok)
```

Latihan ketik manual 9B — baseline hari sebelumnya

```
prediksi_besok = data_es_teh[-1]
print("Prediksi berdasarkan hari terakhir:", prediksi_besok)
```

Bandingkan dua baseline: rata-rata historis vs hari terakhir. Mana lebih masuk akal jika cuaca besok panas? Mengapa?

Baseline memberi tiga manfaat:

1. memberi angka pembanding,

2. mencegah overengineering,

3. membantu menjelaskan apakah model baru benar-benar menambah nilai.

Catatan merah — kata kunci: Jangan Langsung Canggih
Model paling canggih yang tidak mengalahkan baseline sederhana belum layak dibanggakan.

Baseline mean dan MAE

Aktual penjualan `[18,30,42,35]`. Baseline prediksi mean = 31,25.

Error absolut:

```
|31,25-18| = 13,25
|31,25-30| = 1,25
|31,25-42| = 10,75
|31,25-35| = 3,75
MAE = (13,25+1,25+10,75+3,75)/4 = 7,25
```

Model AI berikutnya harus mengalahkan baseline ini. Jika tidak, model kompleks belum berguna.

Tes cepat subbab 9

1. Apa itu baseline?

2. Mengapa baseline penting untuk proyek AI?

3. Buat satu baseline untuk prediksi penjualan kopi.

Subbab 10 — Reproducibility: seed, dependency, dan catatan run

Eksperimen AI harus punya catatan. Minimal:

• versi Python,

• file yang dijalankan,

• tanggal eksperimen,

• dependency,

• seed bila ada random,

• data yang dipakai,

• hasil ringkas.

Checklist reproducibility

Untuk random experiment, gunakan seed:

```
import random
random.seed(42)
```

Untuk dependency, simpan `requirements.txt` bila memakai paket eksternal:

```
python -m pip freeze > requirements.txt
python -m pip install -r requirements.txt
```

Catatan ungu — kata kunci: Catatan Run
Catatan run membuat eksperimen bisa dipercaya. Tanpa catatan, hasil bagus sulit diperiksa ulang.

Seed dan eksperimen acak

Jika eksperimen memakai random, seed membuat hasil bisa diulang.

```
import random
```

```
random.seed(42)
print(random.randint(1, 6))
```

Tanpa seed, hasil bisa berubah setiap run. Perubahan itu tidak selalu salah, tetapi harus disadari.

Template catatan run

```
tanggal:
file:
seed:
input data:
perubahan kode:
hasil utama:
catatan error:
kesimpulan:
```

Tes cepat subbab 10

1. Apa fungsi seed?

2. Mengapa dependency perlu dicatat?

3. Tulis format catatan run sederhana untuk praktikum Bab 3.

Subbab 11 — Git: tombol save untuk proyek AI

Git menyimpan riwayat perubahan. Untuk proyek AI, Git membantu melacak perubahan naskah, kode, notebook, konfigurasi, dan dokumentasi. Dataset besar biasanya tidak langsung disimpan di Git; gunakan data kecil, metadata, atau alat khusus seperti DVC bila perlu nanti.

Alur Git sederhana

Perintah minimal:

```
git status
git add file.py
git commit -m "add first data lab"
git log --oneline
```

Kebiasaan baik:

- commit perubahan kecil tetapi bermakna,
- tulis pesan commit jelas,
- jangan commit data rahasia,
- jangan commit file .env, token, private key,
- untuk notebook, jalankan ulang sebelum commit.

Catatan hijau — kata kunci: Riwayat
Git bukan hanya untuk programmer senior. Git adalah buku riwayat proyek agar kita tahu apa yang berubah dan mengapa.

Git sebagai riwayat eksperimen

Git menyimpan perubahan kode dan naskah. Dalam proyek AI, Git membantu menjawab: versi kode mana yang menghasilkan angka ini? kapan dataset/lab berubah? siapa mengubah instruksi?

```
git status  → lihat perubahan
git diff    → lihat isi perubahan
git commit  → simpan snapshot bermakna
```

Contoh pesan commit yang baik

```
feat: add baseline penjualan warung
fix: correct MAE calculation
 docs: explain Colab workflow
```

Commit bukan formalitas; commit adalah catatan berpikir.

Tes cepat subbab 11

1. Apa fungsi `git status`?

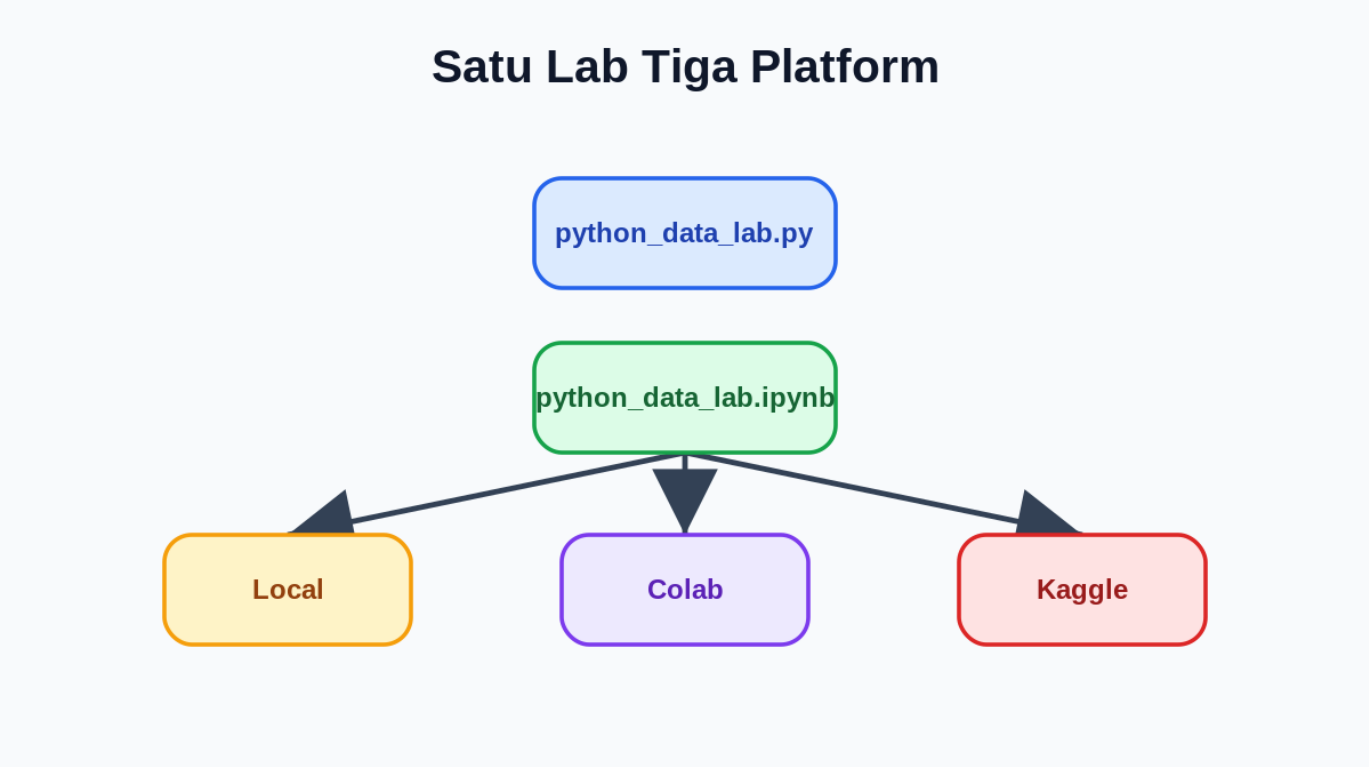
2. Mengapa tidak boleh commit token/API key?

3. Kapan sebaiknya membuat commit?

Subbab 12 — Praktikum penutup: satu lab, tiga cara jalan

Di akhir bab ini, kamu menjalankan lab yang sama dengan tiga cara:

1. file `.py` di terminal/VS Code,
2. notebook `.ipynb` di Jupyter lokal,
3. notebook `.ipynb` di Colab atau Kaggle.



Satu lab tiga platform

File yang disediakan:

- `code/python_data_lab.py`
- `code/python_data_lab.ipynb`

Langkah lokal `.py`:

```
cd zero-to-hero-menaklukkan-ai/chapters/03-python-data-lab/code
python3 python_data_lab.py
```

Langkah Jupyter lokal:

```
python -m pip install notebook
jupyter notebook python_data_lab.ipynb
```

Langkah Colab/Kaggle:

1. buka Colab atau Kaggle Notebook,
2. upload `python_data_lab.ipynb`,
3. jalankan menu Runtime → Run all,
4. simpan salinan hasil eksperimen.

Panduan Colab super-pemula

Jika kamu benar-benar mulai dari nol:

1. Buka browser.

2. Ketik `colab.research.google.com`.

3. Login dengan akun Google.

4. Klik tab Upload.

5. Klik Choose File.

6. Pilih file `python_data_lab.ipynb` dari folder proyek.

7. Setelah terbuka, klik menu Runtime.

8. Pilih Run all.

9. Jika muncul peringatan, baca dulu. Untuk notebook buku ini, kode hanya memakai Python standard library.

10. Setelah selesai, pilih File → Save a copy in Drive agar hasilmu tersimpan.

Jika repo sudah berada di GitHub publik, cara lain:

1. Buka Colab.

2. Pilih tab GitHub.

3. Masukkan URL repo.

4. Pilih notebook `zero-to-hero-menaklukkan-ai/chapters/03-python-data-lab/code/python_data_lab.ipynb`.

5. Klik buka, lalu Runtime → Run all.

Catatan biru — kata kunci: Satu Sumber, Banyak Platform
Praktikum dibuat agar pembaca bisa belajar lewat alat yang paling nyaman: lokal, Jupyter, Colab, atau Kaggle.

Tes cepat subbab 12

1. Jalankan file `.py` dan catat output-nya.

2. Jalankan notebook lokal dengan Restart & Run All.

3. Upload notebook ke Colab atau Kaggle dan jalankan semua sel.

4. Tambahkan 2 baris data baru dan bandingkan hasil rata-rata.

5. Tulis catatan run: platform, waktu, versi Python, dan hasil utama.

Latihan hitung dan coding terstruktur Bab 3

1. Hitung manual mean, min, max, dan range untuk `[18,30,42,35]`.

2. Tulis kode Python untuk menghitung total dengan loop, bukan `sum()`.

3. Buat list of dict berisi 3 hari penjualan warung. Hitung rata-ratanya.

4. Buat rule segmentasi: `ramai` jika penjualan >= mean, selain itu `sepi`.

5. Hitung MAE baseline mean untuk aktual `[18,30,42,35]`.

6. Jalankan kode dengan seed 42 dan catat hasil random pertama.

7. Tulis catatan run mini dengan template reproducibility.

Latihan ini menjadikan Python bukan sekadar syntax, tetapi alat menghitung dan memeriksa ide AI.

Ringkasan Bab 3

- Python adalah bahasa kerja utama praktikum AI dalam buku ini.
- File `.py` cocok untuk resep kode yang bisa dijalankan ulang.
- Notebook cocok untuk eksplorasi, penjelasan, dan laporan eksperimen.
- Colab/Kaggle memudahkan belajar dari browser, tetapi data sensitif sebaiknya tetap lokal.
- Dataset kecil membantu memeriksa logika sebelum masuk model kompleks.
- EDA membantu kita bertanya kepada data sebelum membuat model.
- Baseline adalah pembanding wajib untuk model AI.
- Reproducibility membutuhkan catatan run, dependency, seed, dan struktur folder.
- Git membantu menyimpan riwayat proyek, tetapi jangan commit rahasia.
- Praktikum yang baik bisa dijalankan di lokal, notebook, Colab, dan Kaggle.

Referensi utama bab

[R1] Python Software Foundation. Dokumentasi Python resmi. Digunakan untuk perintah dasar dan prinsip menjalankan Python. [R2] Project Jupyter. Dokumentasi Jupyter Notebook/JupyterLab. Digunakan untuk konsep notebook dan instalasi lokal. [R3] Google Colab documentation. Digunakan untuk alur menjalankan notebook di browser. [R4] Kaggle Notebooks documentation. Digunakan untuk alur menjalankan notebook dan dataset di Kaggle. [R5] Pro Git book. Digunakan untuk konsep Git dasar; tidak disalin. [R6] Géron, A. *Hands-On Machine Learning*. Digunakan sebagai inspirasi reproducible ML workflow; tidak disalin. [R7] Wilson et al. “Good Enough Practices in Scientific Computing.” Digunakan sebagai inspirasi prinsip reproducibility; tidak disalin.

Catatan validasi internal v0.3

Aspek	Status	Catatan
File `.py`	Siap	Script hanya memakai standard library agar mudah jalan.
Notebook `.ipynb`	Siap	Notebook dibuat setara dengan script dan dapat diupload ke Colab/Kaggle.
Instalasi Jupyter	Tervalidasi awal	Instruksi memakai `pip install notebook/jupyterlab`.
Reproducibility	Tervalidasi awal	Ada seed, catatan run, checklist, dan baseline.
Risiko data	Tervalidasi awal	Ada peringatan data sensitif untuk platform cloud.