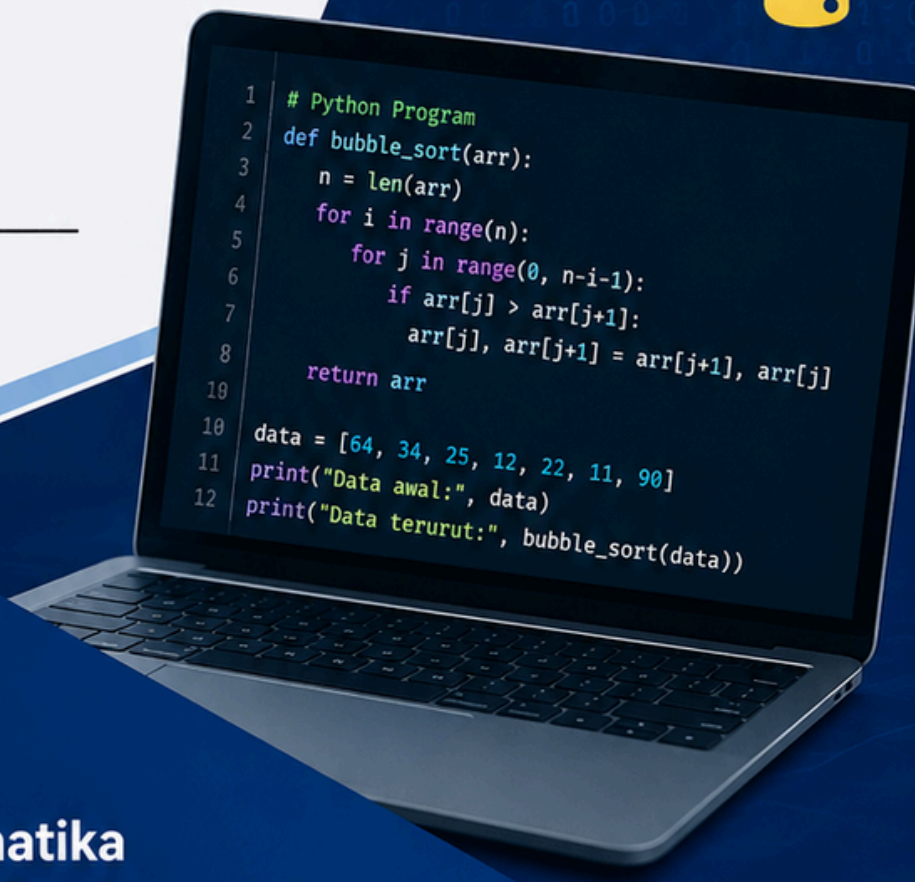
Modul Praktikum

Algoritma dan Pemrograman

Instruktur: **Syahiduz Zaman**

Laboran: _____

Asisten Laboratorium: _____



Teknik Informatika
Fakultas Sains dan Teknologi
UIN Maulana Malik Ibrahim Malang

2026

DAFTAR ISI

MODUL 1

Mengenal Python, Google Colab, dan Cara Berpikir Algoritmik

Mata Kuliah	Praktikum Algoritma dan Pemrograman
Semester	1
Pertemuan	1 dari 10
Durasi	100 menit
Platform	Google Colab
Bahasa	Python 3
Level	Pemula

A. Gambaran Modul

Pertemuan pertama jangan langsung dibebani terlalu banyak sintaks. Target utamanya adalah membuat mahasiswa memahami bahwa pemrograman bukan sekadar "menulis kode", tetapi **menerjemahkan cara berpikir manusia menjadi langkah-langkah yang dapat dijalankan komputer**.

Python digunakan sebagai alat untuk belajar algoritma. Karena itu, mahasiswa sejak awal dibiasakan dengan pola:

masalah → algoritma → kode → eksekusi → kesalahan → perbaikan → hasil

Pendekatan ini sejalan dengan *Think Python* yang memang dimulai dari konsep dasar pemrograman sebelum bergerak ke fungsi, struktur data, dan OOP. Buku tersebut juga menggunakan Jupyter Notebook sebagai lingkungan belajar, sehingga sangat cocok diterapkan melalui Google Colab.

Python Crash Course juga dirancang sebagai pengantar pemrograman berbasis praktik dan problem solving, dengan latihan yang meningkat secara bertahap.

B. Capaian Pembelajaran

Setelah menyelesaikan praktikum ini, mahasiswa diharapkan mampu:

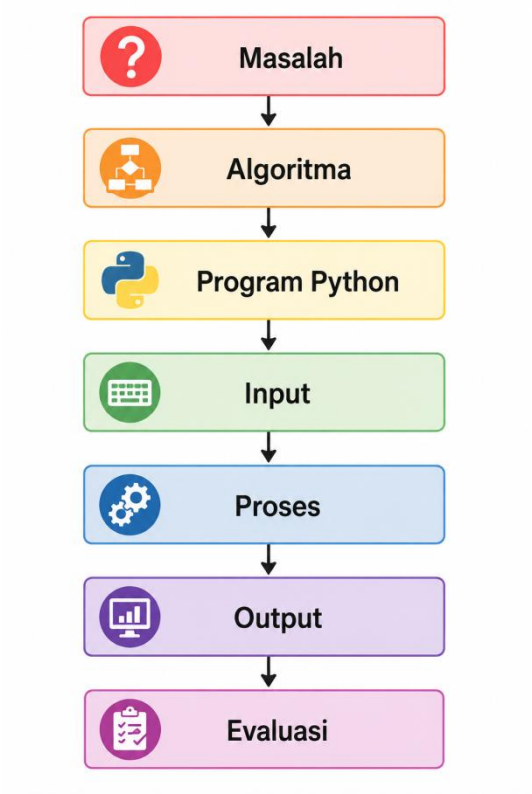
1. Menjelaskan hubungan antara masalah, algoritma, dan program.
2. Menggunakan Google Colab untuk menjalankan program Python.
3. Mengenali struktur dasar sebuah program Python.
4. Menggunakan variabel untuk menyimpan dan mengolah nilai.
5. Mengenali tipe data dasar: `int`, `float`, `str`, dan `bool`.
6. Menggunakan operator aritmetika dasar.

7. Menggunakan `print()` untuk menampilkan hasil.
8. Membuat program sederhana berdasarkan algoritma yang dirancang sendiri.
9. Membaca pesan kesalahan sederhana dari Python.
10. Menjelaskan kembali logika program yang dibuat, bukan hanya menunjukkan bahwa program "berhasil jalan".

Target nomor 10 penting. Mahasiswa tidak boleh sejak pertemuan pertama terbiasa dengan pola **copy–paste–run**.

C. Peta Konsep

Mahasiswa diperkenalkan dengan hubungan:



Contoh:

Masalah: menghitung total belanja.

Algoritmanya:

1. Masukkan harga barang.
2. Masukkan jumlah barang.

3. Kalikan harga dengan jumlah.
4. Tampilkan total.

Kemudian diterjemahkan ke Python.

```
harga = 15000
jumlah = 3

total = harga * jumlah

print(total)
```

Output:

```
45000
```

Mahasiswa kemudian harus mampu menjawab:

“Apa algoritmanya jika harga barang, jumlah barang, dan diskon semuanya berubah?”

Di sinilah pembelajaran algoritma mulai masuk, meskipun sintaks Python masih sangat sederhana.

D. Persiapan Google Colab

Google Colab digunakan sebagai lingkungan praktikum selama satu semester. Mahasiswa tidak perlu menginstal Python di komputer pada tahap awal.

Buka Google Colab dan buat notebook baru.

Struktur notebook yang digunakan selama semester:

```
Praktikum Algoritma dan Pemrograman
Nama:
NIM:
Kelas:

Modul 01 – Python dan Algoritma
```

Kemudian biasakan mahasiswa membedakan dua jenis cell:

Code Cell

digunakan untuk menjalankan Python.

Text/Markdown Cell

digunakan untuk menulis penjelasan, algoritma, kesimpulan, dan refleksi.

Ini penting karena notebook nantinya tidak hanya menjadi tempat menyimpan kode, tetapi menjadi **catatan proses berpikir mahasiswa**.

Think Python edisi ketiga memang dirancang sebagai notebook yang menggabungkan teks dan kode sehingga kode dapat dijalankan, dimodifikasi, dan diuji dalam satu tempat.

E. Skenario Pembelajaran 100 Menit

Waktu	Kegiatan
0–10 menit	Orientasi: apa itu algoritma dan pemrograman
10–20 menit	Mengenal Google Colab
20–35 menit	Program Python pertama
35–50 menit	Variabel dan tipe data
50–65 menit	Operator dan ekspresi
65–80 menit	Latihan algoritma sederhana
80–92 menit	Tantangan logika
92–100 menit	Refleksi dan exit ticket

Komposisinya sengaja lebih banyak praktik daripada ceramah.

F. Materi 1 — Algoritma Itu Apa?

Sebelum menulis Python, mahasiswa diberikan sebuah masalah sederhana:

Anda ingin membuat program untuk menghitung luas sebuah persegi panjang.

Jangan langsung berikan kode.

Minta mahasiswa menjawab:

Input apa yang diperlukan?

Jawaban:

- panjang
- lebar

Proses apa yang dilakukan?

```
luas = panjang × lebar
```

Output apa yang dihasilkan?

- luas persegi panjang

Kemudian tuliskan algoritma:

```
1. Masukkan panjang
2. Masukkan lebar
3. Hitung panjang × lebar
4. Simpan hasil sebagai luas
5. Tampilkan luas
```

Baru setelah itu diterjemahkan menjadi Python.

```
panjang = 10
lebar = 5

luas = panjang * lebar

print(luas)
```

Output:

```
50
```

Tekankan:

Algoritma tidak sama dengan Python.

Algoritma adalah langkah penyelesaian masalah. Python adalah salah satu bahasa untuk mengekspresikan langkah tersebut kepada komputer.

Ini akan menjadi konsep yang terus digunakan sampai Modul 10.

G. Materi 2 — Program Python Pertama

Mulai dengan program paling sederhana:

```
print("Halo, dunia!")
```

Kemudian eksperimen:

```
print("Nama saya Ahmad")
print("Saya mahasiswa semester 1")
print("Saya sedang belajar Python")
```

Tanyakan:

Apa yang terjadi jika tanda kutip dihilangkan?

Misalnya:

```
print(Halo, dunia!)
```

Mahasiswa sengaja diminta menjalankannya.

Tujuannya bukan sekadar mengetahui bahwa program error, tetapi mulai membiasakan diri **tidak takut terhadap error**.

Python Crash Course secara eksplisit menggunakan kesalahan sebagai bagian dari proses belajar dan menunjukkan bagaimana traceback membantu menemukan sumber masalah.

H. Materi 3 — Variabel

Gunakan analogi sederhana:

Variabel adalah **nama yang merujuk kepada sebuah nilai**.

Ini lebih tepat daripada sekadar mengatakan “variabel adalah kotak penyimpanan”.

Think Python mendefinisikan variabel sebagai nama yang merujuk pada sebuah nilai dan menjelaskan assignment statement melalui bentuk:

```
n = 17
```

Contoh:

```
nama = "Budi"
umur = 18
tinggi = 170.5

print(nama)
print(umur)
print(tinggi)
```

Kemudian ubah nilainya:

```
umur = 19

print(umur)
```

Tanyakan:

Apakah `umur` adalah 18 atau 19?

Ini kesempatan pertama memperkenalkan bahwa **assignment bukan persamaan matematika**.

I. Materi 4 — Tipe Data

Kenalkan empat tipe data utama.

```
nama = "Budi"
umur = 18
tinggi = 170.5
mahasiswa_aktif = True
```

Kemudian:

```
print(type(nama))
print(type(umur))
print(type(tinggi))
print(type(mahasiswa_aktif))
```

Hasil:

```
<class 'str'>
<class 'int'>
<class 'float'>
<class 'bool'>
```

Mahasiswa kemudian mencoba:

```
x = 10
y = "10"

print(x)
print(y)
```

Lalu:

```
print(x + 5)
```

dan:

```
print(y + 5)
```

Program kedua akan menghasilkan error.

Diskusikan:

Mengapa angka `10` dan teks `"10"` tidak sama?

Ini sangat penting sebagai fondasi pengolahan data nantinya. Data scientist akan berhadapan dengan data numerik, teks, kategori, tanggal, nilai kosong, dan berbagai representasi data lainnya.

J. Materi 5 — Operator dan Ekspresi

Kenalkan:

```
+    penjumlahan
-    pengurangan
*    perkalian
/    pembagian
//   pembagian bulat
%    modulus/sisa
**   perpangkatan
```

Contoh:

```
a = 10
b = 3

print(a + b)
print(a - b)
print(a * b)
print(a / b)
print(a // b)
print(a % b)
print(a ** b)
```

Kemudian berikan pertanyaan:

Tanpa menjalankan Python, prediksi hasil setiap baris.

Baru setelah mahasiswa memprediksi, jalankan program.

Ini merupakan pola yang sebaiknya dipertahankan sepanjang semester:

Prediksi → Jalankan → Bandingkan → Jelaskan

Think Python sendiri menggunakan latihan semacam ini: mahasiswa diminta memperkirakan tipe atau hasil ekspresi sebelum menggunakan Python untuk memeriksanya.

K. Praktikum Utama — Program Penghitung Nilai

Sekarang mahasiswa membuat program yang sedikit lebih realistis.

Masalah:

Seorang mahasiswa memperoleh nilai tugas 80, UTS 75, dan UAS 90. Nilai akhir dihitung dengan bobot:

Tugas = 20%
UTS = 30%
UAS = 50%

Mahasiswa diminta membuat algoritma terlebih dahulu.

Algoritma

```
1. Tentukan nilai tugas
2. Tentukan nilai UTS
3. Tentukan nilai UAS
4. Hitung kontribusi masing-masing nilai
5. Jumlahkan semua kontribusi
6. Tampilkan nilai akhir
```

Kemudian Python:

```
tugas = 80
uts = 75
uas = 90

nilai_akhir = (
    tugas * 0.20 +
    uts * 0.30 +
    uas * 0.50
)

print(nilai_akhir)
```

Hasil:

```
83.0
```

Kemudian mahasiswa diminta mengubah nilai:

```
tugas = 90
uts = 60
uas = 80
```

dan melihat perubahan hasil.

L. Tantangan 1 — Prediksi Sebelum Eksekusi

Berikan kode:

```
a = 10
b = 3

c = a + b
d = a * b
e = c + d

print(c)
print(d)
print(e)
```

Mahasiswa **tidak boleh menjalankan program terlebih dahulu**.

Tugas:

1. Prediksi nilai `c`.
2. Prediksi nilai `d`.
3. Prediksi nilai `e`.
4. Setelah itu jalankan program.
5. Jika prediksi salah, jelaskan mengapa.

Tujuannya membangun kebiasaan **tracing program**.

M. Tantangan 2 — Tukar Nilai

Diberikan:

```
a = 10
b = 20
```

Bagaimana membuat:

```
a = 20
b = 10
```

tanpa mengetik ulang angka 20 dan 10?

Mahasiswa mencoba terlebih dahulu.

Solusi yang kemudian diperkenalkan:

```
a = 10
b = 20

a, b = b, a

print(a)
print(b)
```

Jangan terlalu lama menjelaskan sintaks `a, b = b, a`. Yang penting mahasiswa menyadari bahwa satu masalah dapat mempunyai beberapa cara penyelesaian.

N. Tantangan 3 — Kalkulator Belanja

Buat program yang menghitung total belanja.

Data:

```
Harga barang = Rp25.000
Jumlah       = 4
Diskon       = 10%
```

Program harus menghasilkan:

```
Harga awal  : Rp100.000
Diskon      : Rp10.000
Total bayar : Rp90.000
```

Mahasiswa harus menentukan sendiri algoritmanya.

Salah satu solusi:

```
harga = 25000
jumlah = 4
diskon = 0.10

total_awal = harga * jumlah
nilai_diskon = total_awal * diskon
total_bayar = total_awal - nilai_diskon

print("Harga awal :", total_awal)
print("Diskon     :", nilai_diskon)
print("Total bayar:", total_bayar)
```

Perhatikan bahwa mahasiswa belum perlu diajari `if`, `for`, function, atau library. Kita sedang membangun fondasi.

O. Tantangan Logika — “Berapa yang Harus Dibayar?”

Sebuah toko menjual buku seharga Rp50.000.

Jika membeli 3 buku:

```
Harga normal = 3 × 50.000
```

Tetapi pelanggan mendapat diskon 15%.

Pertanyaan:

Berapa yang harus dibayar?

Kemudian naikkan level:

Bagaimana jika jumlah buku dan harga buku berubah?

Dan level berikutnya:

Bagaimana jika harga buku, jumlah buku, dan diskon semuanya disimpan dalam variabel?

Tujuan latihan ini adalah membuat mahasiswa mulai melihat bahwa:

```
50.000 × 3 × 0.85
```

bukan sekadar perhitungan matematika, tetapi dapat dibangun menjadi sebuah **algoritma yang dapat digunakan kembali**.

P. Mini Data Challenge

Karena arah mata kuliah sejak awal adalah menuju data science, Modul 1 sudah boleh diberi sentuhan kecil tentang data.

Misalnya data lima mahasiswa:

```
nilai1 = 80
nilai2 = 75
nilai3 = 90
nilai4 = 65
nilai5 = 85
```

Pertanyaan:

1. Berapa jumlah seluruh nilai?
2. Berapa rata-ratanya?
3. Berapa nilai tertinggi?
4. Berapa nilai terendah?

Pada tahap ini mahasiswa **belum boleh menggunakan `list`, `max()`, `min()`, atau NumPy**.

Tujuannya justru agar mereka merasakan betapa tidak efisiennya mengolah data secara manual.

Misalnya:

```
total = nilai1 + nilai2 + nilai3 + nilai4 + nilai5
rata_rata = total / 5

print(rata_rata)
```

Lalu dosen mengatakan:

“Bayangkan datanya bukan lima mahasiswa, tetapi 50.000 mahasiswa. Apakah cara ini masih masuk akal?”

Ini menjadi **teaser untuk Modul 3 dan Modul 7**, ketika mereka belajar struktur data dan pengolahan dataset.

Q. Penggunaan AI: Boleh, tetapi Jangan Menjadi “Mesin Jawaban”

Karena Think Python edisi ketiga memang sudah memasukkan penggunaan virtual assistant/LLM dalam proses belajar, mahasiswa dapat diperkenalkan dengan AI sejak pertemuan pertama. Buku tersebut bahkan menekankan penggunaan LLM untuk mempercepat pembelajaran, testing, dan debugging.

Namun aturan praktikum:

AI boleh digunakan untuk memahami error, tetapi mahasiswa harus mampu menjelaskan kode yang dikumpulkan.

Contoh prompt yang diperbolehkan:

“Saya pemula Python. Jelaskan mengapa kode berikut menghasilkan NameError. Jangan langsung berikan kode perbaikannya.”

atau:

“Berikan tiga kemungkinan penyebab error ini dan jelaskan cara mengeceknya.”

Bukan:

“Kerjakan soal ini.”

Ini sekaligus melatih **AI literacy**, bukan sekadar penggunaan AI.

R. Exit Ticket — 8 Menit Terakhir

Sebelum praktikum selesai, mahasiswa menjawab lima pertanyaan singkat.

1. Apa perbedaan algoritma dan program?

2. Apa perbedaan `10` dan `"10"`?

3. Apa fungsi `print()`?

4. Apa yang dimaksud variabel?

5. Jika program menghasilkan error, apa yang pertama kali harus dilakukan?

Pertanyaan kelima sengaja sederhana. Jawaban yang kita inginkan bukan:

“Tanya ChatGPT.”

Tetapi:

Baca pesan error → cari baris bermasalah → pahami penyebab → perbaiki → jalankan kembali.

S. Tugas Setelah Praktikum

Tugas Individu: “Kalkulator Kehidupan Mahasiswa”

Buat program Python yang menghitung pengeluaran seorang mahasiswa dalam satu minggu.

Data minimal:

```
uang saku per hari
```

```
pengeluaran makan
transportasi
internet
pengeluaran lain
```

Program harus menghasilkan:

```
Total pengeluaran per hari
Total pengeluaran per minggu
Sisa uang
Persentase uang yang digunakan
```

Mahasiswa harus menyerahkan:

1. Algoritma dalam bentuk langkah-langkah.
2. Kode Python.
3. Screenshot/output program.
4. Penjelasan singkat tentang cara kerja program.

T. Rubrik Penilaian

Komponen	Bobot
Algoritma/logika	25%
Ketepatan kode	25%
Pemahaman variabel & tipe data	15%
Kemampuan menjelaskan program	15%
Tantangan logika	10%
Kerapian notebook	10%
Total	**100%**

Saya menyarankan **jangan memberikan bobot terbesar pada "program berhasil dijalankan"**. Kalau tidak, mahasiswa akan belajar bahwa tujuan praktikum adalah menghasilkan output, bukan memahami proses.

U. Bacaan Lanjutan

1. Think Python — Allen B. Downey

Untuk modul ini mahasiswa membaca:

Chapter 1 — Programming as a Way of Thinking

Fokus:

- expression

- value
- type
- operator
- function
- error
- cara berpikir komputasional

Bab ini juga menyediakan latihan prediksi hasil ekspresi dan tipe data.

Kemudian:

Chapter 2 — Variables and Statements

Fokus:

- variables
- assignment
- statements
- `print()`
- `import`
- argument
- module

Khusus untuk Google Colab: mahasiswa juga perlu membaca bagian awal Think Python mengenai Jupyter Notebook karena edisi ketiga memang dirancang untuk bekerja dalam lingkungan notebook.

2. Python Crash Course — Eric Matthes

Untuk modul ini:

Chapter 1 — Getting Started

dan

Chapter 2 — Variables and Simple Data Types

Fokus Chapter 2:

- variables
- naming variables
- strings
- integers
- floats
- comments
- error sederhana

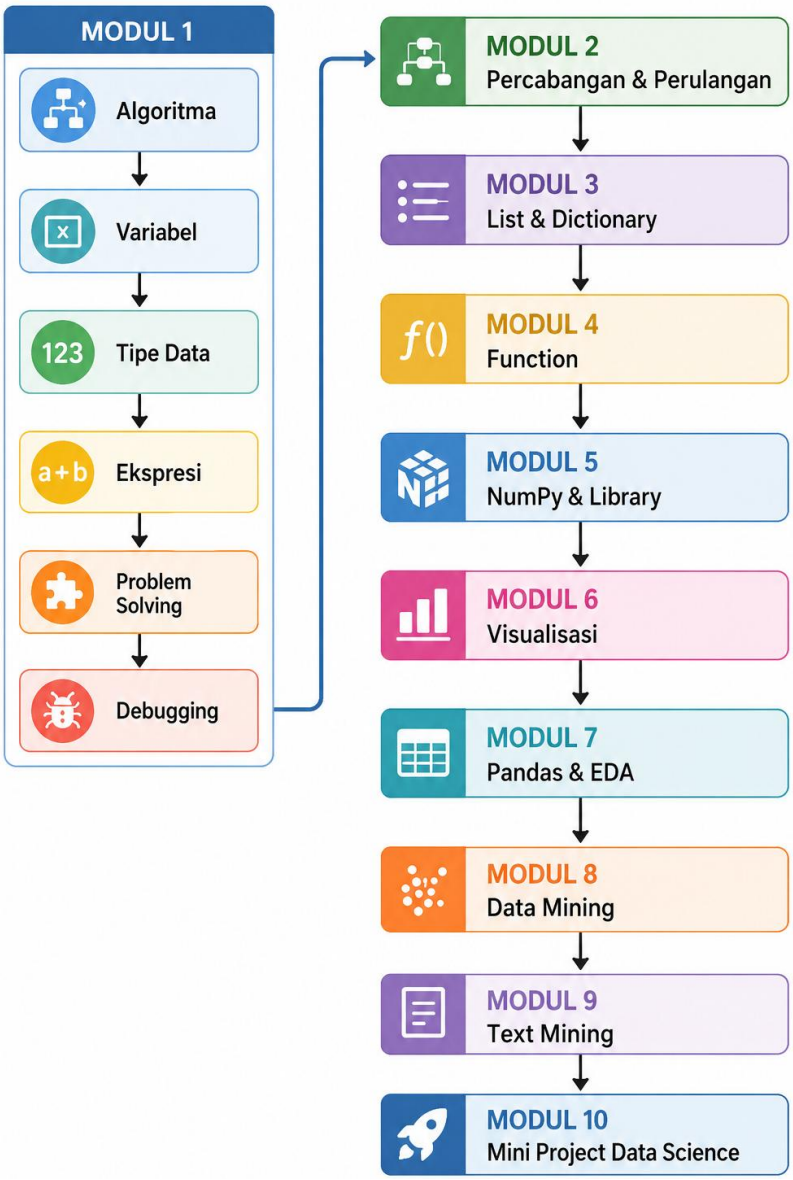
- latihan mandiri

Python Crash Course juga secara eksplisit menggunakan latihan kecil untuk membiasakan mahasiswa belajar melalui praktik.

V. Hubungan Modul 1 dengan 9 Modul Berikutnya

Modul 1 jangan dipandang sebagai “belajar Python dasar” saja.

Ia membangun fondasi:



MODUL 2

Percabangan, Perulangan, dan Logika Program

Mata Kuliah	Praktikum Algoritma dan Pemrograman
Semester	1
Pertemuan	2 dari 10
Durasi	100 menit
Platform	Google Colab
Bahasa	Python 3
Level	Pemula

Modul 2 mulai membawa mahasiswa dari program yang hanya melakukan perhitungan menuju program yang **dapat mengambil keputusan dan mengulang pekerjaan**.

Ini merupakan titik penting dalam pembelajaran algoritma. Mahasiswa mulai memahami bahwa algoritma bukan hanya urutan instruksi, tetapi juga dapat berbentuk:

Jika kondisi tertentu terjadi, lakukan A; jika tidak, lakukan B.

dan:

Ulangi proses tertentu selama kondisi masih terpenuhi.

Think Python menempatkan konsep conditional dan loop pada bagian awal buku. Dalam struktur bukunya, enam bab pertama membangun elemen dasar seperti aritmetika, conditional, loop, fungsi, dan recursion.

A. Tujuan Pembelajaran

Setelah praktikum, mahasiswa mampu:

1. Menjelaskan konsep kondisi dalam algoritma.
2. Menggunakan operator perbandingan.
3. Menggunakan operator logika `and`, `or`, dan `not`.
4. Membuat percabangan menggunakan `if`.
5. Menggunakan `if-else`.
6. Menggunakan `if-elif-else`.
7. Membuat program dengan kondisi bertingkat.
8. Menjelaskan konsep perulangan.
9. Menggunakan `for`.
10. Menggunakan `while`.

11. Menggunakan variabel penghitung (*counter*).
12. Membuat algoritma sederhana yang menggabungkan percabangan dan perulangan.
13. Melakukan *tracing* terhadap program.
14. Memecahkan masalah tanpa langsung menggunakan library.

Target utama modul ini bukan hafalan sintaks.

Mahasiswa harus mulai mampu menjawab:

“Mengapa program mengambil keputusan seperti itu?”

B. Konsep Utama

Modul ini memperkenalkan tiga struktur dasar algoritma:

1. Sequence

Instruksi dijalankan berurutan.

```
A
↓
B
↓
C
```

Contoh:

```
harga = 20000
jumlah = 3
total = harga * jumlah
print(total)
```

2. Selection

Program memilih tindakan berdasarkan kondisi.

```
       kondisi?
       /     \
     ya       tidak
     ↓          ↓
    A           B
```

Dalam Python:

```
if kondisi:
    A
else:
```

```
    B
```

3. Repetition

Program mengulangi instruksi.

```
     kondisi?
     /     \
   ya       tidak
   ↓          ↓
 proses      selesai
   ↓
 kembali
```

Dengan demikian mahasiswa mulai mengenal tiga pola algoritmik fundamental:

Sequence → Selection → Repetition

Ini akan menjadi fondasi hampir seluruh modul setelahnya.

C. Skenario Pembelajaran 100 Menit

Waktu	Kegiatan
0–10 menit	Review Modul 1 dan tracing program
10–25 menit	Konsep kondisi dan operator perbandingan
25–40 menit	`if`, `if-else`, `if-elif-else`
40–50 menit	Operator logika
50–65 menit	Perulangan `for`
65–75 menit	Perulangan `while`
75–88 menit	Challenge algoritma
88–96 menit	Mini Data Challenge
96–100 menit	Exit ticket

D. Review Modul 1: Tracing

Mulai dengan kode berikut.

```
a = 10
b = 5

c = a + b
d = c * 2

print(c)
print(d)
```

Sebelum menjalankan, mahasiswa mengisi:

Variabel	Nilai
a	?
b	?
c	?
d	?

Kemudian jalankan.

Tujuannya adalah mengingat kembali prinsip:

Jangan selalu langsung menjalankan kode. Prediksi dahulu.

Kebiasaan ini sangat penting ketika program mulai kompleks.

E. Materi 1 — Kondisi

Berikan masalah:

Seorang mahasiswa dinyatakan lulus jika nilainya minimal 60.

Secara algoritmik:

```
Jika nilai >= 60
    mahasiswa lulus
Jika tidak
    mahasiswa tidak lulus
```

Python:

```
nilai = 75

if nilai >= 60:
    print("Lulus")
```

Kemudian:

```
nilai = 45

if nilai >= 60:
    print("Lulus")
```

Tidak ada output.

Tanyakan:

Mengapa tidak ada output?

Ini kesempatan menjelaskan bahwa `if` hanya menjalankan blok kode ketika kondisinya bernilai `True`.

F. Operator Perbandingan

Kenalkan:

```
>     lebih besar
<     lebih kecil
>=    lebih besar atau sama dengan
<=    lebih kecil atau sama dengan
==    sama dengan
!=    tidak sama dengan
```

Contoh:

```
umur = 19

print(umur > 17)
print(umur < 17)
print(umur == 19)
print(umur != 19)
```

Hasil berupa:

```
True
False
True
False
```

Tekankan perbedaan:

```
x = 10
```

dengan:

```
x == 10
```

Yang pertama adalah **assignment**.

Yang kedua adalah **perbandingan**.

Ini kesalahan klasik mahasiswa pemula sehingga perlu dibuat latihan khusus.

G. Latihan 1 — Apakah Bilangan Positif?

Buat algoritma:

```
Masukkan sebuah bilangan
Jika bilangan > 0
    tampilkan "Positif"
```

Python:

```
angka = 15

if angka > 0:
    print("Positif")
```

Kemudian mahasiswa menguji:

```
angka = -10
```

dan:

```
angka = 0
```

Pertanyaan:

Apa yang terjadi pada angka 0?

Mahasiswa kemudian diminta memperbaiki algoritma sehingga program dapat membedakan:

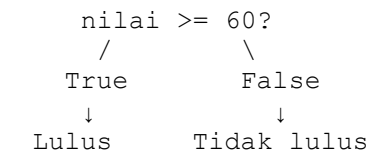
```
Positif
Negatif
Nol
```

H. Materi 2 — `if-else`

Sekarang program memiliki dua kemungkinan.

```
nilai = 55

if nilai >= 60:
    print("Lulus")
else:
    print("Tidak lulus")
```

Visualisasikan:



Tekankan bahwa struktur algoritma ini sangat penting.

I. Latihan 2 — Bilangan Genap atau Ganjil

Gunakan operator modulus `%`.

```
angka = 17

if angka % 2 == 0:
    print("Genap")
else:
    print("Ganjil")
```

Jangan hanya memberikan rumus.

Tanyakan:

Mengapa `angka % 2 == 0` berarti bilangan genap?

Mahasiswa harus menjelaskan konsep **sisa pembagian**.

Kemudian:

```
angka = 28
```

dan:

```
angka = 101
```

J. Materi 3 — `if-elif-else`

Masalah:

Nilai	Grade
≥ 85	A
≥ 75	B
≥ 65	C
≥ 50	D
< 50	E

Algoritma:

```
Jika nilai >= 85
    A
Jika tidak, jika nilai >= 75
    B
Jika tidak, jika nilai >= 65
    C
Jika tidak, jika nilai >= 50
    D
Jika tidak
    E
```

Python:

```
nilai = 78

if nilai >= 85:
    grade = "A"
elif nilai >= 75:
    grade = "B"
elif nilai >= 65:
    grade = "C"
elif nilai >= 50:
    grade = "D"
else:
    grade = "E"

print(grade)
```

Kemudian mahasiswa menguji:

```
95
82
70
55
40
```

K. Challenge: Mengapa Urutannya Penting?

Berikan kode:

```
nilai = 90

if nilai >= 50:
    grade = "D"
elif nilai >= 65:
    grade = "C"
elif nilai >= 75:
    grade = "B"
elif nilai >= 85:
    grade = "A"

print(grade)
```

Tanyakan:

Apakah program menghasilkan A?

Mahasiswa akan menemukan bahwa hasilnya bukan A.

Kemudian diskusikan:

Mengapa?

Ini latihan algoritmik yang lebih penting daripada sekadar menghafalkan `if`.

Mahasiswa harus memahami bahwa **urutan kondisi memengaruhi hasil program**.

L. Materi 4 — Operator Logika

Kenalkan:

```
and
or
not
```

Contoh:

```
umur = 20
nilai = 75

if umur >= 18 and nilai >= 60:
    print("Memenuhi syarat")
```

Artinya:

Syarat pertama **dan** syarat kedua harus benar.

Contoh `or`:

```
hari = "Sabtu"

if hari == "Sabtu" or hari == "Minggu":
    print("Akhir pekan")
```

Contoh `not`:

```
aktif = False

if not aktif:
    print("Mahasiswa tidak aktif")
```

M. Challenge Logika — Beasiswa

Seorang mahasiswa mendapatkan beasiswa jika:

- IPK ≥ 3.50
- dan penghasilan orang tua < Rp5.000.000

Buat algoritmanya.

Kemudian implementasikan:

```
ipk = 3.65
penghasilan = 4000000

if ipk >= 3.50 and penghasilan < 5000000:
    print("Memenuhi syarat beasiswa")
else:
    print("Belum memenuhi syarat")
```

Naikkan tantangannya:

Bagaimana jika mahasiswa boleh mendapatkan beasiswa jika **IPK ≥ 3.75 ATAU** mendapatkan rekomendasi khusus?

Mahasiswa harus merancang kondisi sendiri.

N. Materi 5 — Perulangan

Sekarang berikan masalah:

Tampilkan angka 1 sampai 5.

Cara pertama:

```
print(1)
print(2)
print(3)
print(4)
print(5)
```

Tanyakan:

Kalau sampai 1.000 bagaimana?

Di sinilah mahasiswa melihat kebutuhan perulangan.

O. `for`

Contoh:

```
for i in range(1, 6):
    print(i)
```

Output:

```
1
2
3
4
5
```

Jelaskan konsep:

```
range(1, 6)
```

menghasilkan angka mulai dari 1 sampai sebelum 6.

Kemudian:

```
for i in range(5):
    print(i)
```

Output:

```
0
1
2
3
4
```

Ini penting untuk memahami bahwa Python menggunakan indeks yang dimulai dari 0 pada banyak struktur datanya, konsep yang nantinya akan sangat penting ketika masuk ke list dan data processing.

P. Latihan `for`

Latihan 1

Tampilkan angka 1–10.

```
for i in range(1, 11):
    print(i)
```

Latihan 2

Tampilkan angka genap 2–20.

```
for i in range(2, 21, 2):
    print(i)
```

Latihan 3

Hitung jumlah angka 1–100.

```
total = 0

for i in range(1, 101):
    total = total + i

print(total)
```

Perhatikan bahwa ini mulai memperkenalkan konsep **accumulator**.

Mahasiswa harus memahami:

```
total = 0

total = total + 1
total = total + 2
total = total + 3
...
```

Ini bukan sekadar sintaks `for`.

Ini adalah **algoritma akumulasi**.

Q. Challenge Logika — Jumlah Bilangan

Tanpa menggunakan fungsi `sum()`, hitung:

```
1 + 2 + 3 + ... + 1000
```

Mahasiswa wajib menggunakan `for`.

Kemudian tantangan berikutnya:

Hitung hanya bilangan genap dari 1 sampai 1000.

Dan berikutnya:

Hitung hanya bilangan yang habis dibagi 3.

Dengan demikian mahasiswa mulai belajar menggabungkan:

loop + condition

Contoh:

```
total = 0

for i in range(1, 1001):
    if i % 2 == 0:
        total = total + i

print(total)
```

R. Materi 6 — `while`

Jelaskan bahwa `for` cocok ketika kita mengetahui pola perulangannya.

`while` digunakan ketika pengulangan bergantung pada kondisi.

Contoh:

```
angka = 1

while angka <= 5:
    print(angka)
    angka = angka + 1
```

Hasil:

```
1
2
3
4
5
```

Tekankan bagian paling berbahaya:

```
angka = angka + 1
```

Jika mahasiswa menulis:

```
angka = 1

while angka <= 5:
    print(angka)
```

program tidak akan berhenti.

Ini menjadi kesempatan menjelaskan **infinite loop**.

S. Challenge — Tebak Angka Sederhana

Belum perlu menggunakan input yang kompleks. Gunakan nilai yang sudah ditentukan.

```
angka_rahasia = 7
tebakan = 1

while tebakan != angka_rahasia:
    print("Tebakan belum benar")
    tebakan = tebakan + 1

print("Benar!")
```

Tanyakan:

Berapa kali loop dijalankan?

Kemudian mahasiswa diminta mengubah program.

Target:

```
Tebakan 1
```

```
Tebakan 2
Tebakan 3
...
Tebakan 7
Benar!
```

Ini latihan sederhana, tetapi sudah memperkenalkan konsep **state**, **condition**, dan **iteration**.

T. Challenge Utama — FizzBuzz

Ini challenge yang sangat baik untuk mahasiswa semester 1.

Tampilkan angka 1 sampai 30.

Tetapi:

- jika habis dibagi 3 → `Fizz`
- jika habis dibagi 5 → `Buzz`
- jika habis dibagi 3 dan 5 → `FizzBuzz`
- selain itu → tampilkan angka

Contoh:

```
1
2
Fizz
4
Buzz
Fizz
7
8
Fizz
Buzz
11
Fizz
...
30 → FizzBuzz
```

Mahasiswa harus merancang algoritma terlebih dahulu.

Jangan langsung berikan solusi.

Setelah mereka mencoba, baru bahas salah satu kemungkinan:

```
for i in range(1, 31):
    if i % 3 == 0 and i % 5 == 0:
        print("FizzBuzz")
    elif i % 3 == 0:
```

```
        print("Fizz")
    elif i % 5 == 0:
        print("Buzz")
    else:
        print(i)
```

Challenge ini sangat baik karena menguji:

- `for`
- `if`
- `elif`
- `else`
- modulus
- `and`
- urutan kondisi
- algoritma

U. Mini Data Challenge

Sekarang kita mulai mengarah ke data science, tetapi masih tanpa library.

Data nilai:

```
nilai = [70, 85, 60, 90, 75]
```

Namun **jangan dulu menjelaskan list secara formal**. List baru akan menjadi materi utama Modul 3.

Untuk sementara, dosen dapat memberikan data melalui `range` atau beberapa variabel.

Challenge:

Dari nilai 1 sampai 100, berapa banyak nilai yang termasuk kategori “lulus” jika batas lulus adalah 60?

```
jumlah_lulus = 0

for nilai in range(1, 101):
    if nilai >= 60:
        jumlah_lulus = jumlah_lulus + 1

print(jumlah_lulus)
```

Kemudian:

Berapa banyak nilai yang gagal?

Dan:

Berapa persentase nilai yang lulus?

```
jumlah_lulus = 0

for nilai in range(1, 101):
    if nilai >= 60:
        jumlah_lulus += 1

persentase = jumlah_lulus / 100 * 100

print(persentase)
```

Ini belum benar-benar *data mining*, tetapi mahasiswa mulai melihat pola:

data → kondisi → penghitungan → informasi

Pada modul-modul berikutnya, pola ini akan berkembang menjadi:

dataset → preprocessing → analisis → visualisasi → data mining.

V. Challenge Logika Tingkat Lanjut

Untuk mahasiswa yang lebih cepat, berikan challenge tambahan.

Challenge 1 — Bilangan Prima

Buat program untuk menentukan apakah sebuah bilangan adalah bilangan prima.

Contoh:

```
7 → prima
11 → prima
12 → bukan prima
17 → prima
20 → bukan prima
```

Jangan langsung memberikan kode.

Minta mahasiswa memikirkan:

“Bagaimana komputer mengetahui bahwa 17 hanya dapat dibagi oleh 1 dan 17?”

Kemudian mereka dapat menggunakan perulangan untuk mencoba pembagi.

Ini menjadi latihan algoritma yang sangat baik sebelum mahasiswa mengenal fungsi pada Modul 4.

Challenge 2 — Faktor Bilangan

Input:

```
12
```

Output:

```
1
2
3
4
6
12
```

Mahasiswa mencari semua angka yang dapat membagi 12 tanpa sisa.

Challenge 3 — Statistik Sederhana

Tanpa menggunakan:

```
max()
min()
sum()
```

buat program untuk mencari:

- jumlah
- nilai terbesar
- nilai terkecil
- rata-rata

dari:

```
12, 8, 15, 7, 20, 10
```

Challenge ini sengaja disiapkan untuk **Modul 3**, ketika struktur data akan dibahas secara lebih serius.

W. Debugging Challenge

Berikan program yang salah:

```
nilai = 75

if nilai >= 60
    print("Lulus")
else:
    print("Tidak lulus")
```

Minta mahasiswa menemukan kesalahan.

Kemudian:

```
nilai = 75

if nilai = 60:
    print("Lulus")
```

Apa kesalahannya?

Kemudian:

```
for i in range(1, 10):
print(i)
```

Apa yang salah?

Tujuannya memperkenalkan tiga kategori masalah secara intuitif:

1. **Syntax error**
2. **Runtime error**
3. **Logical error**

Think Python menekankan debugging sebagai keterampilan fundamental, termasuk membedakan jenis kesalahan dan mengembangkan strategi untuk menemukannya.

X. Penggunaan AI dalam Modul 2

Pada tahap ini AI boleh digunakan sebagai **partner debugging dan penguji logika**, bukan pembuat jawaban.

Contoh prompt yang diperbolehkan:

Saya sedang belajar Python. Jelaskan mengapa kondisi pada program berikut menghasilkan output yang salah. Jangan berikan kode perbaikannya.

Atau:

Uji logika algoritma saya untuk menentukan apakah sebuah bilangan prima. Cari kasus yang mungkin membuat algoritma saya gagal.

Bahkan bisa dibuat aturan:

Mahasiswa wajib memberikan algoritma terlebih dahulu sebelum meminta AI membantu memperbaiki kode.

Dengan demikian AI tidak mengambil alih proses berpikir.

Think Python edisi ketiga memang memasukkan virtual assistant sebagai bagian dari pembelajaran, termasuk penggunaan AI untuk memperdalam pemahaman dan debugging.

Y. Tugas Praktikum

“Analisis Nilai Mahasiswa”

Buat program yang menerima nilai mahasiswa dan menentukan kategori:

```
85–100  → A
75–84   → B
65–74   → C
50–64   → D
<50     → E
```

Kemudian program menentukan:

```
Lulus / Tidak Lulus
```

Syarat lulus:

```
nilai >= 60
```

Program harus melakukan pengujian terhadap minimal 10 nilai.

Mahasiswa harus menggunakan:

- `if`
- `elif`
- `else`
- `for`
- operator perbandingan

Bonus

Hitung:

- jumlah mahasiswa lulus
- jumlah mahasiswa tidak lulus
- persentase kelulusan

Tanpa menggunakan library tambahan.

Z. Struktur Notebook yang Dikumpulkan

Mahasiswa wajib menyerahkan notebook dengan struktur:

```
# MODUL 2
# Percabangan dan Perulangan

Nama:
NIM:
Kelas:

## 1. Tujuan Praktikum

## 2. Konsep Algoritma

## 3. Latihan Percabangan

## 4. Latihan Perulangan

## 5. Challenge FizzBuzz

## 6. Challenge Logika

## 7. Tugas Analisis Nilai

## 8. Kesimpulan
```

Setiap challenge harus memiliki:

```
Masalah
↓
Algoritma
↓
Kode
↓
Output
↓
Penjelasan
```

Ini perlu dibiasakan sejak awal karena nanti format tersebut akan sangat berguna ketika mahasiswa mengerjakan proyek data science.

AA. Rubrik Penilaian

Komponen	Bobot
Pemahaman algoritma	20%
Percabangan	15%
Perulangan	20%
Challenge logika	20%
Debugging	10%
Kemampuan menjelaskan kode	10%
Kerapian notebook	5%
Total	**100%**

Sekali lagi, **challenge logika diberi bobot besar**. Mahasiswa yang hanya mampu menyalin contoh tidak boleh memperoleh nilai yang sama dengan mahasiswa yang mampu merancang solusi.

AB. Bacaan Lanjutan dari Dua Buku

Think Python — Allen B. Downey

Untuk Modul 2, rujukan utamanya:

Chapter 5 — Conditional Execution

Pelajari konsep pengambilan keputusan dan conditional execution.

Chapter 6 — Return Values

Bagian ini mulai menghubungkan pengambilan keputusan dengan fungsi dan nilai yang dikembalikan.

Chapter 7 — Iteration and Search

Ini menjadi bacaan penting untuk konsep `for`, `while`, iteration, dan algoritma pencarian.

Struktur Think Python memang menempatkan conditional, loop, fungsi, dan recursion pada enam bab awal sebagai fondasi pemrograman.

Python Crash Course — Eric Matthes

Untuk modul ini:

Chapter 5 — If Statements

Fokus pada:

- conditional test
- `if`
- `if-else`
- `if-elif-else`
- comparison
- Boolean expressions

Kemudian:

Chapter 7 — User Input and While Loops

Fokus pada:

- input
- `while`
- kondisi perulangan
- pengulangan berdasarkan kondisi.

Python Crash Course dirancang bertahap dan menggunakan latihan untuk membangun kemampuan pemrograman; ini sesuai dengan pola praktikum yang kita gunakan.

AC. Ke Mana Setelah Ini?

Modul 2 membangun kemampuan seperti pada gambar berikut.

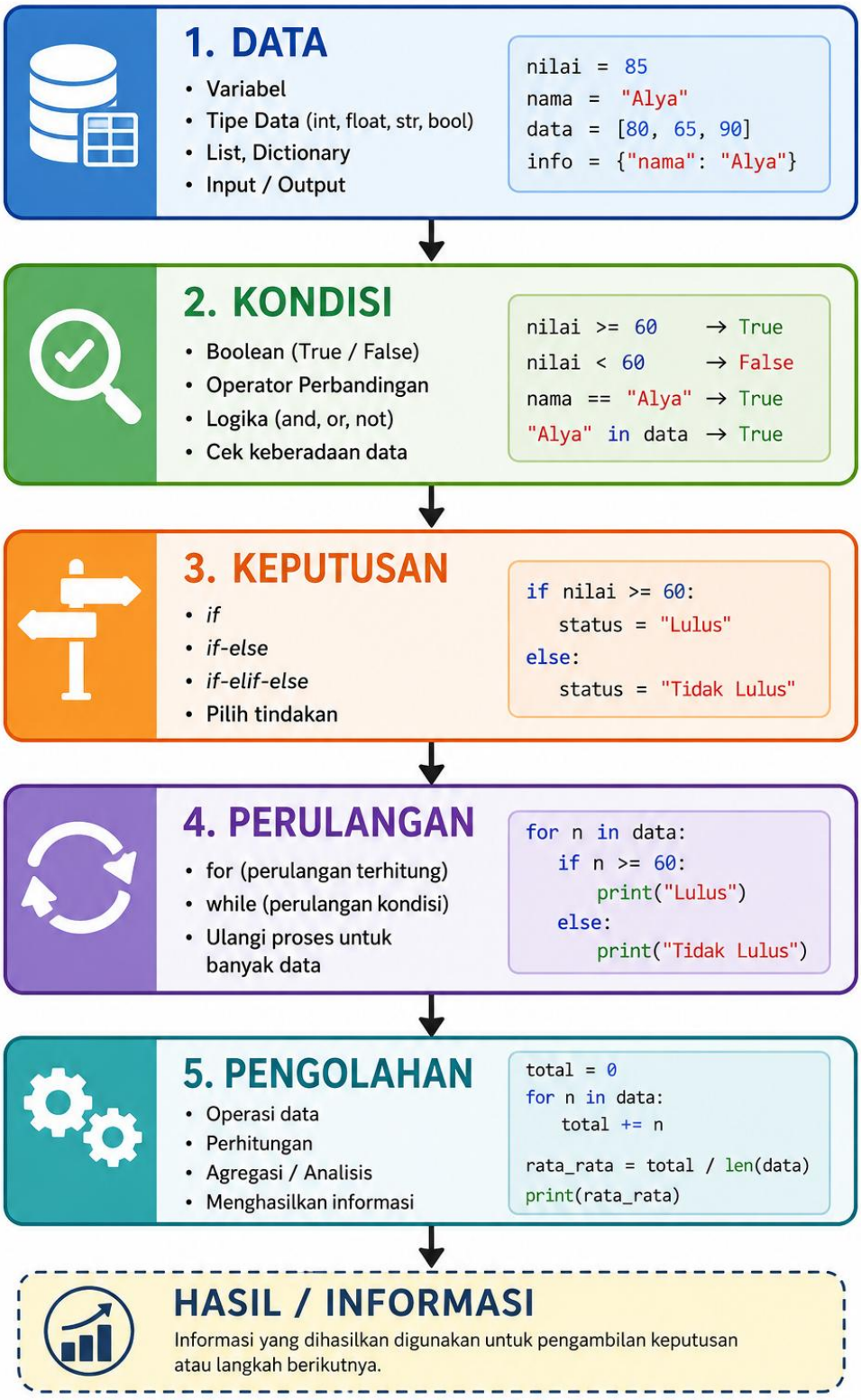
Pada Modul 3 kita mulai menghadapi masalah yang lebih realistis:

“Bagaimana kalau data tidak hanya satu nilai, tetapi 10, 100, atau 10.000 nilai?”

Di situlah kita membutuhkan **struktur data**.

Mahasiswa akan mulai mengenal:

```
List
Dictionary
Index
Searching
Sorting
Counting
```


1. DATA
• Variabel
• Tipe Data (int, float, str, bool)
• List, Dictionary
• Input / Output
nilai = 85
nama = "Alya"
data = [80, 65, 90]
info = {"nama": "Alya"}
2. KONDISI
• Boolean (True / False)
• Operator Perbandingan
• Logika (and, or, not)
• Cek keberadaan data
nilai >= 60 → True
nilai < 60 → False
nama == "Alya" → True
"Alya" in data → True
3. KEPUTUSAN
• if
• if-else
• if-elif-else
• Pilih tindakan
if nilai >= 60:
status = "Lulus"
else:
status = "Tidak Lulus"
4. PERULANGAN
• for (perulangan terhitung)
• while (perulangan kondisi)
• Ulangi proses untuk banyak data
for n in data:
if n >= 60:
print("Lulus")
else:
print("Tidak Lulus")
5. PENGOLAHAN
• Operasi data
• Perhitungan
• Agregasi / Analisis
• Menghasilkan informasi
total = 0
for n in data:
total += n
rata_rata = total / len(data)
print(rata_rata)
HASIL / INFORMASI
Informasi yang dihasilkan digunakan untuk pengambilan keputusan atau langkah berikutnya.

Ini merupakan transisi yang sangat penting menuju data science. Think Python menjelaskan bahwa setelah fondasi awal, buku bergerak ke struktur data inti seperti list, dictionary, dan tuple, yang digunakan untuk menulis program secara lebih efisien.

Jadi alur dua pertemuan pertama sekarang menjadi sangat jelas:

Modul 1:
“Bagaimana komputer menjalankan instruksi?”

Modul 2:
“Bagaimana komputer mengambil keputusan dan mengulang instruksi?”

Modul 3:
“Bagaimana komputer menyimpan dan mengolah banyak data?”

Dan dari sinilah modul-modul berikutnya mulai bergerak semakin kuat ke **data processing, visualisasi, EDA, dan data mining**.

MODUL 3

List, Dictionary, dan Representasi Data

Mata Kuliah	Praktikum Algoritma dan Pemrograman
Semester	1
Pertemuan	3
Durasi	100 menit
Platform	Google Colab
Bahasa	Python 3
Level	Pemula → mulai menuju data processing

Modul 3 adalah salah satu modul yang sangat penting dalam keseluruhan rancangan praktikum ini. Setelah Modul 1 mahasiswa memahami variabel, tipe data, ekspresi, problem solving, dan debugging, kemudian Modul 2 memperkenalkan percabangan dan perulangan, sekarang mahasiswa menghadapi persoalan yang lebih realistis:

Bagaimana menyimpan dan mengolah banyak data sekaligus?

Di sinilah `list` dan `dictionary` mulai menjadi fondasi menuju data science.

Peta jalan modul yang telah kita susun memang menempatkan **Modul 3: List & Dictionary** setelah percabangan dan perulangan, kemudian baru bergerak ke Function, NumPy, Visualisasi, Pandas & EDA, dan Data Mining.

A. Gagasan Besar Modul

Pada Modul 1 mahasiswa bekerja dengan:

```
nilai = 80
```

Satu variabel menyimpan satu nilai.

Pada Modul 2 mahasiswa mulai mengulang:

```
for i in range(1, 11):
    print(i)
```

Tetapi bagaimana kalau kita memiliki:

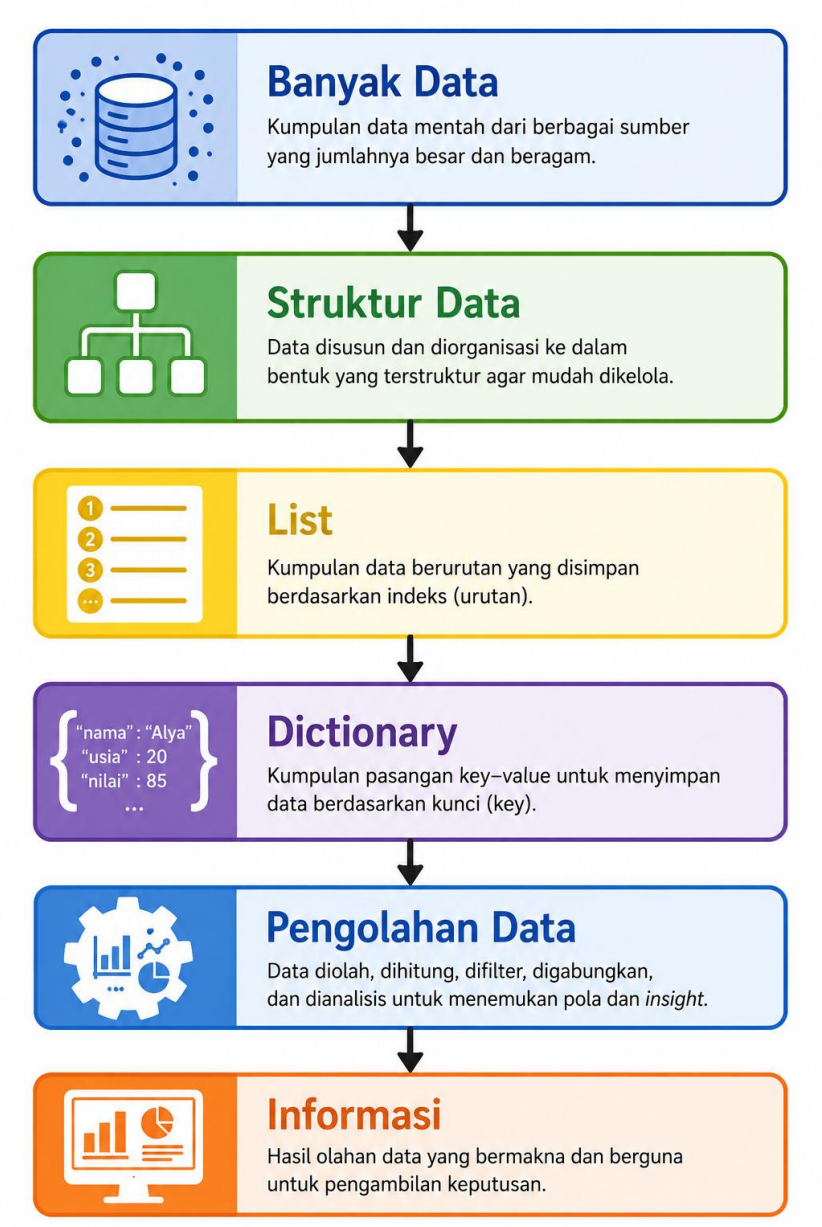
```
78, 85, 90, 67, 72, 88, 95, 60, 76, 83
```

dan ingin:

- menghitung rata-rata,
- mencari nilai tertinggi,
- mencari nilai terendah,
- mencari mahasiswa yang lulus,
- mengurutkan nilai,
- mencari nilai tertentu,
- menghitung frekuensi,
- mengelompokkan data?

Kita membutuhkan **struktur data**.

Konsep utama modul ini:



Think Python menjelaskan list sebagai salah satu tipe bawaan Python yang sangat penting, sedangkan dictionary menjadi struktur untuk pemetaan hubungan antara *key* dan *value*. Bahkan buku tersebut menggunakan dictionary untuk menghitung jumlah kata dan frekuensinya.

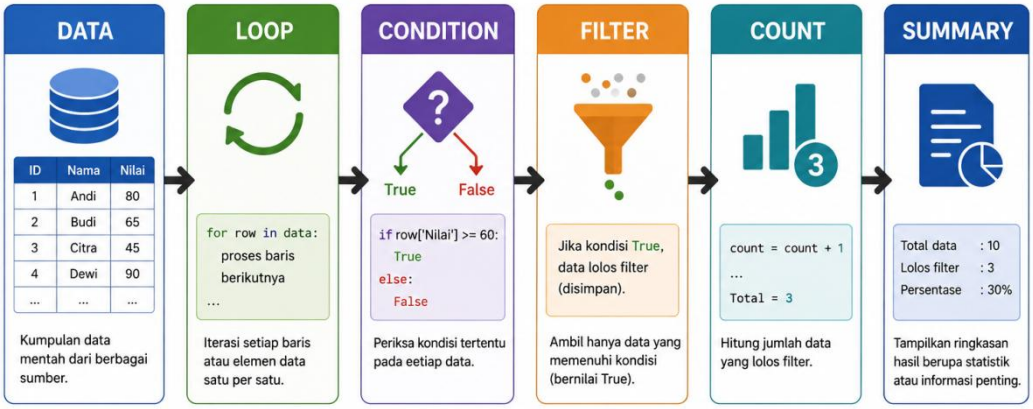
B. Tujuan Pembelajaran

Setelah menyelesaikan modul ini, mahasiswa mampu:

1. Menjelaskan konsep struktur data.
2. Menjelaskan perbedaan variabel tunggal dengan koleksi data.
3. Membuat dan mengakses `list`.
4. Menggunakan indeks pada list.
5. Mengubah elemen list.
6. Menambahkan dan menghapus elemen.
7. Menggunakan `len()`.
8. Melakukan pencarian data dalam list.
9. Menggunakan perulangan pada list.
10. Mengurutkan data.
11. Menggunakan slicing.
12. Membuat list dari hasil perulangan.
13. Memahami list comprehension.
14. Membuat dictionary.
15. Mengakses pasangan `key-value`.
16. Menambah, mengubah, dan menghapus data dictionary.
17. Melakukan iterasi pada dictionary.
18. Menggunakan dictionary untuk merepresentasikan data sederhana.
19. Menggabungkan list, dictionary, percabangan, dan perulangan.
20. Mulai memahami bagaimana data terstruktur direpresentasikan sebelum masuk ke NumPy dan Pandas.

C. Hubungan dengan Modul Sebelumnya

Sehingga mahasiswa mulai mampu melakukan:



Inilah pola yang nantinya muncul berulang kali dalam data science.

D. Skenario Praktikum

Waktu	Kegiatan
0–10 menit	Review Modul 2
10–25 menit	Konsep list dan indexing
25–40 menit	Manipulasi list
40–50 menit	Looping, searching, sorting
50–65 menit	Dictionary
65–75 menit	List of dictionaries
75–88 menit	Data processing challenge
88–96 menit	Mini Data Challenge
96–100 menit	Exit ticket

E. Bagian 1 — Mengapa Kita Membutuhkan List?

Mulai dari cara yang buruk.

```
nilai1 = 80
nilai2 = 75
nilai3 = 90
nilai4 = 65
nilai5 = 88
```

Tanyakan kepada mahasiswa:

Kalau ada 10.000 mahasiswa, apakah kita akan membuat 10.000 variabel?

Tentu tidak.

Kita dapat menggunakan:

```
nilai = [80, 75, 90, 65, 88]
```

Sekarang satu variabel dapat merepresentasikan sekumpulan data.

Python Crash Course memperkenalkan list sebagai kumpulan item yang memiliki urutan tertentu dan menekankan bahwa list dapat menyimpan informasi dalam jumlah sangat besar.

F. Membuat List

```
nilai = [80, 75, 90, 65, 88]

print(nilai)
```

Output:

```
[80, 75, 90, 65, 88]
```

List dapat berisi string:

```
nama = ["Andi", "Budi", "Citra", "Dina"]
```

atau angka:

```
nilai = [80, 75, 90, 65, 88]
```

atau bahkan berbagai tipe data:

```
data = ["Andi", 20, 3.75]
```

Namun, tekankan kepada mahasiswa:

Secara teknis Python mengizinkan berbagai tipe data dalam satu list, tetapi untuk pengolahan data kita biasanya membuat struktur yang konsisten.

Ini merupakan kebiasaan penting menuju Pandas.

G. Indexing

Misalnya:

```
nama = ["Andi", "Budi", "Citra", "Dina"]
```

Posisinya:

```
Index     0       1       2       3
          ↓       ↓       ↓       ↓
        Andi    Budi    Citra   Dina
```

Maka:

```
print(nama[0])
print(nama[2])
```

Output:

```
Andi
Citra
```

Python menggunakan indeks yang dimulai dari 0. Python Crash Course secara khusus menekankan bahwa indeks list dimulai dari 0, bukan 1.

H. Negative Index

Python juga memungkinkan:

```
print(nama[-1])
```

Output:

```
Dina
```

Artinya:

```
-4      -3      -2      -1
 ↓       ↓       ↓       ↓
Andi    Budi    Citra   Dina
```

Ini berguna ketika kita tidak mengetahui panjang list tetapi ingin mengambil elemen terakhir.

I. Latihan 1 — Prediksi Output

Sebelum menjalankan:

```
buah = ["apel", "mangga", "jeruk", "pisang", "melon"]

print(buah[0])
```

```
print(buah[2])
print(buah[-1])
```

Mahasiswa wajib memprediksi output terlebih dahulu.

Kemudian jalankan.

Ini mempertahankan kebiasaan **tracing** dari Modul 1.

J. Memodifikasi List

List bersifat mutable.

```
nilai = [70, 80, 90]

nilai[1] = 85

print(nilai)
```

Output:

```
[70, 85, 90]
```

Think Python secara eksplisit membahas bahwa list bersifat mutable, sehingga elemennya dapat diubah setelah list dibuat.

K. Menambahkan Data

Gunakan `append()`.

```
nilai = [70, 80, 90]

nilai.append(95)

print(nilai)
```

Hasil:

```
[70, 80, 90, 95]
```

Tambahkan beberapa data:

```
nilai.append(65)
nilai.append(88)
```

L. Menyisipkan Data

Gunakan `insert()`.

```
nama = ["Andi", "Citra", "Dina"]

nama.insert(1, "Budi")

print(nama)
```

Hasil:

```
["Andi", "Budi", "Citra", "Dina"]
```

M. Menghapus Data

Dengan indeks:

```
nama = ["Andi", "Budi", "Citra", "Dina"]

del nama[1]

print(nama)
```

Dengan nilai:

```
nama.remove("Citra")
```

Dengan elemen terakhir:

```
nama.pop()
```

Jelaskan perbedaan konseptual:

```
del     → hapus berdasarkan posisi
remove  → hapus berdasarkan nilai
pop     → ambil/hapus elemen
```

Think Python juga membahas `remove()` dan menunjukkan bahwa mencoba menghapus nilai yang tidak terdapat dalam list menghasilkan `ValueError`.

N. Panjang List

```
nilai = [80, 75, 90, 65, 88]

print(len(nilai))
```

Output:

```
5
```

Konsep penting:

```
len(list)
```

berarti:

Berapa banyak elemen yang terdapat dalam list?

Bukan:

Apa indeks terakhir?

Jika panjang list adalah 5, indeks terakhir adalah 4.

Ini menjadi sumber *off-by-one error* yang sangat umum.

Python Crash Course secara khusus menggunakan `len()` untuk mengetahui jumlah elemen list dan membahas `IndexError` akibat salah indeks.

O. Debugging — IndexError

Berikan:

```
nama = ["Andi", "Budi", "Citra"]

print(nama[3])
```

Apa yang terjadi?

```
IndexError: list index out of range
```

Tanyakan:

Mengapa indeks 3 salah?

Karena:

```
0 → Andi
1 → Budi
2 → Citra
```

Tidak ada indeks 3.

Kemudian berikan:

```
print(nama[-1])
```

Mahasiswa menemukan cara yang lebih aman untuk mengambil elemen terakhir.

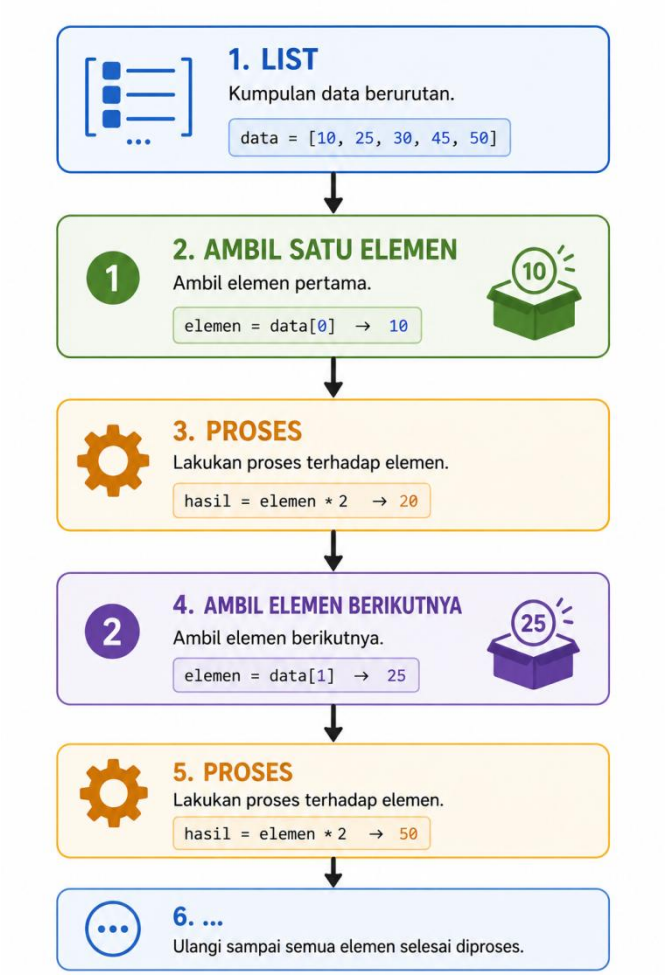
P. Looping List

Sekarang gabungkan Modul 2 dengan Modul 3.

```
nama = ["Andi", "Budi", "Citra", "Dina"]

for orang in nama:
    print(orang)
```

Ini jauh lebih penting daripada sekadar menghafal sintaks.

Mahasiswa harus memahami:



Think Python juga menjadikan *looping through a list* sebagai salah satu bagian utama pembahasan list.

Q. Filtering Data

Sekarang gabungkan:

list + for + if

```
nilai = [45, 60, 75, 90, 55, 80]

for n in nilai:
    if n >= 60:
        print(n)
```

Output:

```
60
75
90
80
```

Ini adalah bentuk sederhana dari **filtering data**.

Dan konsep ini akan muncul kembali pada:

```
Pandas
EDA
Data Mining
Machine Learning
```

R. Menghitung Data

Berapa banyak mahasiswa yang lulus?

```
nilai = [45, 60, 75, 90, 55, 80]

jumlah_lulus = 0

for n in nilai:
    if n >= 60:
        jumlah_lulus += 1

print(jumlah_lulus)
```

Hasil:

```
4
```

Ini merupakan algoritma **counting**.

S. Mengumpulkan Data Hasil Filter

Sekarang jangan hanya mencetak.

Simpan hasilnya.

```
nilai = [45, 60, 75, 90, 55, 80]

lulus = []

for n in nilai:
    if n >= 60:
        lulus.append(n)

print(lulus)
```

Output:

```
[60, 75, 90, 80]
```

Perhatikan perubahan konsep:

```
Data awal
    ↓
Filtering
    ↓
Data baru
```

Ini sudah mulai mendekati pola preprocessing data.

T. Sorting

```
nilai = [75, 90, 60, 85, 70]

nilai.sort()

print(nilai)
```

Hasil:

```
[60, 70, 75, 85, 90]
```

Untuk urutan terbalik:

```
nilai.sort(reverse=True)
```

Atau gunakan:

```
sorted(nilai)
```

Perbedaan yang perlu dipahami:

```
nilai.sort()
```

mengubah list asli.

Sedangkan:

```
sorted(nilai)
```

menghasilkan urutan baru tanpa mengubah list asli.

Python Crash Course membahas sorting permanen dengan `sort()`, sorting sementara dengan `sorted()`, urutan terbalik, dan panjang list dalam Chapter 3.

U. Statistik Sederhana

Python sudah menyediakan:

```
nilai = [70, 80, 90, 65, 88]

print(min(nilai))
print(max(nilai))
print(sum(nilai))
print(len(nilai))
```

Rata-rata:

```
rata_rata = sum(nilai) / len(nilai)

print(rata_rata)
```

Python Crash Course memang memperkenalkan `min()`, `max()`, dan `sum()` sebagai fungsi sederhana untuk statistik list.

Namun ada satu aturan pedagogis penting:

Mahasiswa harus terlebih dahulu memahami cara menghitungnya dengan loop sebelum diperbolehkan menggunakan fungsi bawaan.

Jadi:

```
Algoritma manual
```

```
      ↓
Memahami konsep
      ↓
Fungsi bawaan
      ↓
Efisiensi
```

V. Slicing

Misalnya:

```
nilai = [60, 65, 70, 75, 80, 85, 90]
```

Ambil beberapa data:

```
print(nilai[1:4])
```

Hasil:

```
[65, 70, 75]
```

Pola:

```
list[start:stop]
```

`stop` tidak ikut diambil.

Contoh:

```
nilai[:3]
```

mengambil tiga elemen pertama.

```
nilai[3:]
```

mengambil mulai indeks 3 sampai akhir.

Slicing merupakan bagian eksplisit dari pembahasan list dalam Think Python Chapter 9.

W. List Comprehension

Mahasiswa sebelumnya sudah melakukan:

```
kuadrat = []

for i in range(1, 11):
```

```
    kuadrat.append(i ** 2)
```

Python menyediakan bentuk yang lebih ringkas:

```
kuadrat = [i ** 2 for i in range(1, 11)]
```

Hasil:

```
[1, 4, 9, 16, 25, 36, 49, 64, 81, 100]
```

Python Crash Course memperkenalkan list comprehension sebagai cara menggabungkan loop dan pembuatan list dalam satu ekspresi.

Tetapi jangan menjadikan list comprehension sebagai materi hafalan.

Mahasiswa harus dapat menjelaskan:

```
[i ** 2 for i in range(1, 11)]
```

sebagai:

“Untuk setiap `i` dari 1 sampai 10, masukkan i^2 ke dalam list.”

X. Latihan List Comprehension

Buat:

```
angka = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
```

Kemudian hasilkan:

Kuadrat

```
[i ** 2 for i in angka]
```

Bilangan genap

```
[i for i in angka if i % 2 == 0]
```

Kelipatan 3

```
[i for i in angka if i % 3 == 0]
```

Ini mulai memperlihatkan bagaimana **filtering** dapat ditulis secara ringkas.

Y. Bagian 2 — Dictionary

Sekarang berikan masalah:

Kita memiliki:

```
nama = "Andi"
umur = 20
ipk = 3.65
```

Informasinya berkaitan dengan orang yang sama.

Daripada tiga variabel terpisah, kita dapat membuat:

```
mahasiswa = {
    "nama": "Andi",
    "umur": 20,
    "ipk": 3.65
}
```

Ini adalah dictionary.

Think Python menjelaskan dictionary sebagai *mapping*: berbeda dengan list yang menggunakan indeks, dictionary menghubungkan **key** dengan **value**.

Z. Memahami Key dan Value

```
mahasiswa = {
    "nama": "Andi",
    "umur": 20,
    "ipk": 3.65
}
```

Strukturnya:

```
"nama" → "Andi"
"umur" → 20
"ipk"  → 3.65
```

Bagian kiri:

key

Bagian kanan:

value

AA. Mengakses Dictionary

```
print(mahasiswa["nama"])
print(mahasiswa["ipk"])
```

Hasil:

```
Andi
3.65
```

Perhatikan perbedaannya:

List:

```
nama[0]
```

Dictionary:

```
mahasiswa["nama"]
```

Ini merupakan perbedaan konseptual yang harus benar-benar dipahami.

```
LIST
posisi → data

DICTIONARY
key → data
```

AB. Menambah Data

```
mahasiswa["semester"] = 2
```

Sekarang:

```
print(mahasiswa)
```

AC. Mengubah Data

```
mahasiswa["ipk"] = 3.75
```

AD. Menghapus Data

```
del mahasiswa["umur"]
```

AE. Dictionary dan `get()`

Berikan contoh:

```
print(mahasiswa["alamat"])
```

Jika key tidak ada, Python menghasilkan `KeyError`.

Alternatif:

```
print(mahasiswa.get("alamat"))
```

Hasil:

```
None
```

Ini menjadi teknik penting ketika bekerja dengan data yang mungkin tidak lengkap.

AF. Looping Dictionary

```
mahasiswa = {
    "nama": "Andi",
    "umur": 20,
    "ipk": 3.75
}

for key, value in mahasiswa.items():
    print(key, value)
```

Output:

```
nama Andi
umur 20
ipk 3.75
```

Kemudian:

```
for key in mahasiswa.keys():
    print(key)
```

dan:

```
for value in mahasiswa.values():
    print(value)
```

AG. Mengapa Dictionary Penting untuk Data Science?

Karena data dunia nyata hampir selalu memiliki atribut.

Misalnya satu mahasiswa:

```
{
    "nama": "Andi",
    "usia": 20,
    "prodi": "Informatika",
    "ipk": 3.75
}
```

Satu produk:

```
{
    "produk": "Laptop",
    "harga": 8500000,
    "stok": 12
}
```

Satu transaksi:

```
{
    "tanggal": "2026-08-25",
    "produk": "Buku",
    "jumlah": 3,
    "total": 150000
}
```

Ini mulai menyerupai **record** dalam database.

Dan ketika kita memiliki banyak record:

```
mahasiswa = [
    {
        "nama": "Andi",
        "ipk": 3.75
    },
    {
        "nama": "Budi",
        "ipk": 3.20
    },
    {
        "nama": "Citra",
        "ipk": 3.90
    }
]
```

kita sudah memiliki bentuk sederhana dari **dataset**.

AH. List of Dictionaries

Ini merupakan bagian penting modul.

```
mahasiswa = [
    {"nama": "Andi", "ipk": 3.75},
    {"nama": "Budi", "ipk": 3.20},
    {"nama": "Citra", "ipk": 3.90},
    {"nama": "Dina", "ipk": 3.45}
]
```

Visualisasinya:

```
LIST
 │
 ├── Dictionary 1
 │       ├── nama
 │       └── ipk
 │
 ├── Dictionary 2
 │       ├── nama
 │       └── ipk
 │
 ├── Dictionary 3
 │       ├── nama
 │       └── ipk
 │
 └── Dictionary 4
         ├── nama
         └── ipk
```

Ini jauh lebih dekat dengan cara kita berpikir tentang dataset.

AI. Mengolah Dataset Sederhana

Sekarang kita mulai benar-benar melakukan data processing.

```
mahasiswa = [
    {"nama": "Andi", "ipk": 3.75},
    {"nama": "Budi", "ipk": 3.20},
    {"nama": "Citra", "ipk": 3.90},
    {"nama": "Dina", "ipk": 3.45}
]
```

Tampilkan mahasiswa dengan IPK ≥ 3.50:

```
for m in mahasiswa:
    if m["ipk"] >= 3.50:
```

```
        print(m["nama"], m["ipk"])
```

Output:

```
Andi 3.75
Citra 3.9
```

Perhatikan:

```
LIST
  ↓
LOOP
  ↓
DICTIONARY
  ↓
KEY
  ↓
CONDITION
  ↓
FILTER
```

Ini sudah merupakan miniatur proses analisis data.

AJ. Menghitung Rata-rata IPK

```
total = 0

for m in mahasiswa:
    total += m["ipk"]

rata_rata = total / len(mahasiswa)

print(rata_rata)
```

Kemudian:

Berapa mahasiswa yang IPK-nya di atas rata-rata?

```
jumlah = 0

for m in mahasiswa:
    if m["ipk"] > rata_rata:
        jumlah += 1

print(jumlah)
```

Ini merupakan latihan yang sangat baik karena mahasiswa harus menggabungkan beberapa konsep sekaligus.

AK. Data Challenge — Analisis Nilai

Sekarang gunakan:

```
data = [
    {"nama": "Andi", "nilai": 80},
    {"nama": "Budi", "nilai": 65},
    {"nama": "Citra", "nilai": 90},
    {"nama": "Dina", "nilai": 55},
    {"nama": "Eko", "nilai": 75}
]
```

Mahasiswa harus membuat program untuk memperoleh:

1. jumlah mahasiswa;
2. nilai tertinggi;
3. nilai terendah;
4. rata-rata;
5. jumlah mahasiswa lulus;
6. nama mahasiswa dengan nilai ≥ 80.

Belum boleh menggunakan Pandas.

Belum boleh menggunakan NumPy.

Belum boleh menggunakan library.

Hanya:

```
list
dictionary
for
if
variabel
```

Tujuannya justru untuk memperkuat **algoritma sebelum library**.

AL. Challenge — Mencari Nilai Tertinggi

Mahasiswa tidak boleh langsung menggunakan:

```
max()
```

Buat algoritma manual.

Petunjuk:

```
nilai_tertinggi = data[0]["nilai"]

for m in data:
    if m["nilai"] > nilai_tertinggi:
        nilai_tertinggi = m["nilai"]
```

Kemudian kembangkan agar program juga mendapatkan **nama mahasiswa**.

```
Citra → 90
```

Di sini mahasiswa belajar konsep **tracking nilai maksimum**.

Ini penting karena algoritma semacam ini tetap berguna bahkan ketika nanti mereka menggunakan library.

AM. Challenge — Frequency Counting

Sekarang mulai mendekati pola data mining.

Data:

```
kategori = [
    "A", "B", "A", "C", "B",
    "A", "A", "C", "B", "A"
]
```

Pertanyaan:

Berapa kali A, B, dan C muncul?

Mahasiswa diarahkan membuat:

```
frekuensi = {}
```

Kemudian mengisi dictionary.

Target:

```
{
    "A": 5,
    "B": 3,
    "C": 2
}
```

Ini merupakan algoritma **frequency counting**.

Dan ini bukan latihan main-main.

Think Python Chapter 10 memang menggunakan dictionary sebagai struktur untuk menghitung jumlah kemunculan elemen.

Konsep ini nantinya akan muncul dalam:

- analisis frekuensi,
- text mining,
- categorical data,
- klasifikasi,
- association analysis,
- dan berbagai teknik data mining.

AN. Mini Project Modul 3

“Mini Sistem Analisis Nilai Mahasiswa”

Gunakan dataset:

```
mahasiswa = [
    {"nama": "Andi", "uts": 80, "uas": 85},
    {"nama": "Budi", "uts": 70, "uas": 65},
    {"nama": "Citra", "uts": 90, "uas": 95},
    {"nama": "Dina", "uts": 60, "uas": 70},
    {"nama": "Eko", "uts": 75, "uas": 80},
    {"nama": "Fajar", "uts": 50, "uas": 55}
]
```

Bobot:

```
UTS = 40%
UAS = 60%
```

Program harus menghitung:

```
nilai akhir = UTS × 40% + UAS × 60%
```

Kemudian menentukan:

```
≥ 85 → A
≥ 75 → B
≥ 65 → C
≥ 60 → D
< 60 → E
```

Program menghasilkan informasi seperti:

```
Andi   83.0   B
Budi   67.0   C
Citra  93.0   A
...
```

Kemudian hitung:

- rata-rata kelas;
- nilai tertinggi;
- nilai terendah;
- jumlah mahasiswa lulus;
- jumlah mahasiswa tidak lulus;
- mahasiswa dengan nilai tertinggi.

AO. Tantangan Tambahan

Untuk mahasiswa yang lebih cepat:

Urutkan mahasiswa berdasarkan nilai akhir dari tertinggi ke terendah.

Ini lebih sulit karena kita mulai membutuhkan teknik pengurutan terhadap data yang terstruktur.

Mahasiswa dapat diarahkan menggunakan:

```
sorted()
```

dengan `key`.

Contoh setelah konsep dijelaskan:

```
mahasiswa_sorted = sorted(
    mahasiswa,
    key=lambda m: m["nilai_akhir"],
    reverse=True
)
```

Tetapi `lambda` **tidak perlu dijelaskan secara mendalam pada Modul 3**.

Cukup katakan:

“Ini akan kita bongkar pada Modul 4 ketika mempelajari Function.”

Dengan demikian materi tidak melompat terlalu jauh.

AP. Bonus — Tuple

Tuple dikenalkan secara singkat, bukan sebagai fokus utama.

```
koordinat = (10, 20)
```

Perbedaan:

```
List
→ mutable

Tuple
→ immutable
```

Contoh:

```
koordinat[0] = 15
```

akan menghasilkan error.

Think Python Chapter 11 memang membahas tuple sebagai struktur yang mirip list tetapi immutable, kemudian menghubungkannya dengan *packing*, *unpacking*, `zip`, dan pengurutan.

Untuk Modul 3 cukup sampai konsep:

Gunakan list ketika data perlu dimodifikasi; gunakan tuple ketika sekumpulan nilai sebaiknya tidak berubah.

`zip()` boleh diperkenalkan sebagai bonus:

```
nama = ["Andi", "Budi", "Citra"]
nilai = [80, 75, 90]

for n, v in zip(nama, nilai):
    print(n, v)
```

Think Python menggunakan `zip()` untuk memasangkan elemen dari dua sequence dan melakukan pemrosesan secara berpasangan.

AQ. Debugging Challenge

Berikan kode berikut:

```
nilai = [70, 80, 90]
```

```
for i in range(len(nilai)):
    print(nilai[i + 1])
```

Apa masalahnya?

Mahasiswa harus menemukan bahwa ketika `i = 2`, program meminta:

```
nilai[3]
```

yang tidak tersedia.

Kemudian perbaiki:

```
for i in range(len(nilai)):
    print(nilai[i])
```

Lalu tanyakan:

Mengapa sebenarnya kita tidak perlu menggunakan indeks di sini?

Jawaban yang lebih Pythonic:

```
for n in nilai:
    print(n)
```

Ini kesempatan memperkenalkan gagasan:

Gunakan struktur program yang paling sederhana untuk masalah yang sedang diselesaikan.

Prinsip ini juga sejalan dengan filosofi Python yang diperkenalkan Python Crash Course melalui *Zen of Python*: kesederhanaan dan keterbacaan harus menjadi pertimbangan dalam menulis kode.

AR. Penggunaan AI pada Modul 3

AI mulai boleh digunakan untuk **menguji pemahaman struktur data**, tetapi bukan untuk langsung mengerjakan tugas.

Contoh prompt:

Jelaskan perbedaan list dan dictionary menggunakan contoh data mahasiswa. Jangan berikan kode terlebih dahulu.

Atau:

Saya memiliki algoritma untuk mencari nilai tertinggi dalam list. Berikan tiga kasus uji yang dapat membuat algoritma saya gagal.

Lebih menarik lagi:

Saya menggunakan dictionary untuk menghitung frekuensi kemunculan data. Jangan berikan solusi. Tanyakan kepada saya pertanyaan yang membantu saya menemukan algoritmanya.

Dengan pendekatan ini, AI berfungsi sebagai **tutor Socratic**, bukan mesin pembuat jawaban.

AS. Tugas Praktikum

Dataset Penjualan Sederhana

Gunakan data:

```
transaksi = [
    {"produk": "Buku", "kategori": "Pendidikan", "harga": 50000,
"jumlah": 2},
    {"produk": "Pensil", "kategori": "ATK", "harga": 5000,
"jumlah": 10},
    {"produk": "Buku", "kategori": "Pendidikan", "harga": 50000,
"jumlah": 3},
    {"produk": "Pulpen", "kategori": "ATK", "harga": 7000,
"jumlah": 5},
    {"produk": "Buku", "kategori": "Pendidikan", "harga": 50000,
"jumlah": 1}
]
```

Hitung untuk setiap transaksi:

```
total = harga × jumlah
```

Kemudian tentukan:

1. total seluruh penjualan;
2. produk dengan jumlah terjual paling banyak;
3. produk dengan omzet terbesar;
4. jumlah transaksi kategori Pendidikan;
5. jumlah transaksi kategori ATK;
6. produk yang paling sering muncul.

Aturan

Tidak boleh menggunakan:

```
NumPy
Pandas
Matplotlib
library eksternal
```

Hanya gunakan:

```
list
dictionary
for
if
variabel
fungsi bawaan Python
```

Ini sengaja dibuat sebagai **jembatan menuju Modul 5 dan Modul 7**.

AT. Format Notebook Google Colab

Notebook mahasiswa:

```
MODUL 3
LIST, DICTIONARY, DAN DATA PROCESSING

Nama:
NIM:
Kelas:

1. Tujuan Praktikum

2. Review Modul 2

3. List
   3.1 Membuat List
   3.2 Indexing
   3.3 Modifikasi
   3.4 Append
   3.5 Remove
   3.6 Sorting
   3.7 Slicing

4. Dictionary
   4.1 Key dan Value
   4.2 Menambah Data
   4.3 Mengubah Data
   4.4 Looping Dictionary

5. List of Dictionaries

6. Data Processing Challenge

7. Mini Project Analisis Nilai

8. Tugas Penjualan
```

```
9. Debugging

10. Refleksi
```

Untuk setiap challenge:

```
Masalah
   ↓
 Data
   ↓
Algoritma
   ↓
 Kode
   ↓
Output
   ↓
Interpretasi
```

Bagian **interpretasi** sengaja ditambahkan.

Mahasiswa tidak cukup menampilkan:

```
Total = 850000
```

Mereka harus mampu menulis:

“Total penjualan dari seluruh transaksi adalah Rp850.000.”

Ini merupakan latihan kecil menuju cara menulis hasil analisis data.

AU. Rubrik Penilaian

Komponen	Bobot
Pemahaman list	15%
Manipulasi dan indexing	10%
Looping dan filtering	15%
Dictionary	15%
List of dictionaries	15%
Data processing challenge	15%
Debugging	5%
Interpretasi hasil	5%
Total	**100%**

Yang penting, **list of dictionaries dan data processing mendapat bobot besar** karena bagian tersebut merupakan jembatan menuju data science.

AV. Bacaan dari Dua Buku

Untuk modul ini, pembagian bacaan sebaiknya dibuat sangat jelas.

Python Crash Course, 3rd Edition

Chapter 3 — Introducing Lists

Pelajari:

- list;
- indexing;
- modifying;
- adding;
- removing;
- sorting;
- `len()`;
- index errors.

Chapter 4 — Working with Lists

Pelajari:

- looping;
- numerical lists;
- `range()`;
- statistik sederhana;
- list comprehension;
- slicing;
- copying list;
- tuple.

Daftar isi buku menunjukkan bahwa Chapter 3 dan 4 memang dirancang khusus untuk membangun kemampuan bekerja dengan list, dari indexing sampai list comprehension dan slicing.

Chapter 6 — Dictionaries

Pelajari:

- key-value;
- akses data;
- menambah dan mengubah data;
- `get()`;
- looping dictionary;

- nesting.

Ini sangat relevan dengan bagian **list of dictionaries** pada modul.

Think Python, 3rd Edition

Bacaan utama:

Chapter 9 — Lists

Bagian yang relevan:

- A List Is a Sequence
- Lists Are Mutable
- List Slices
- List Operations
- List Methods
- Lists and Strings
- Looping Through a List
- Sorting Lists
- Objects and Values
- Aliasing
- List Arguments

Daftar isi Think Python menempatkan seluruh topik tersebut dalam Chapter 9, halaman 121–136.

Chapter 10 — Dictionaries

Fokus:

- dictionary sebagai mapping;
- creating dictionaries;
- `in`;
- counting;
- looping;
- lists and dictionaries;
- accumulating a list.

Ini justru sangat cocok dengan tujuan kita untuk mulai mengenalkan mahasiswa pada **frequency counting** dan pengolahan dataset.

Chapter 11 — Tuples

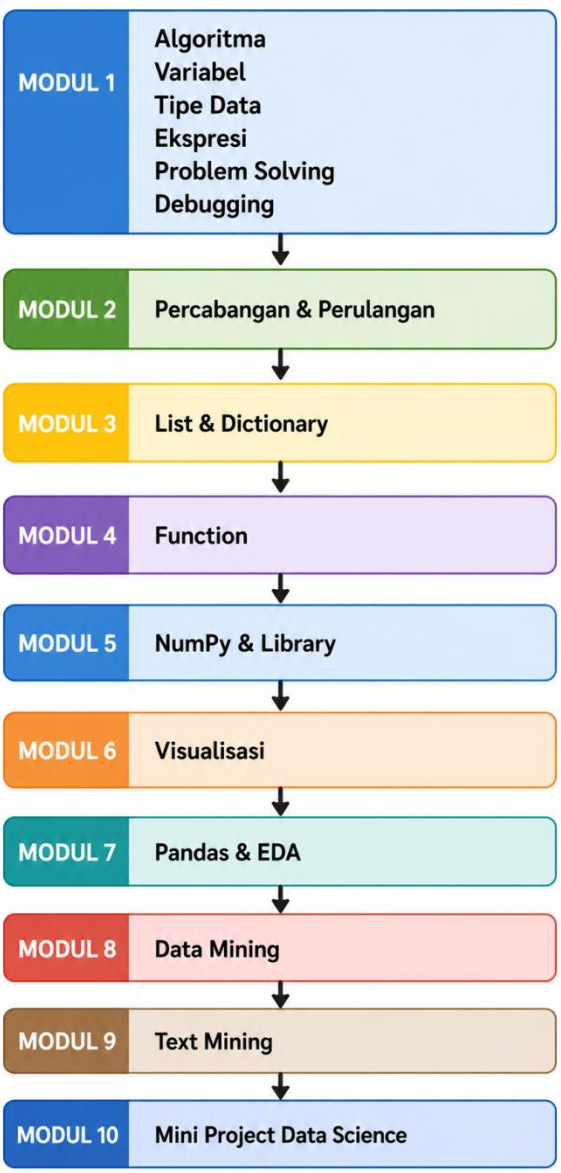
Baca bagian:

- tuple;
- immutability;
- tuple assignment;
- `zip`;
- comparing and sorting.

Untuk Modul 3, ini cukup sebagai pengayaan, tidak perlu menjadi kompetensi utama.

AW. Posisi Modul 3 dalam Roadmap

Sekarang struktur pembelajaran mulai terlihat lebih kuat:



Peta jalan yang kita buat memang menempatkan List & Dictionary sebagai fondasi sebelum Function, kemudian NumPy, Visualisasi, Pandas & EDA, dan Data Mining.

Catatan untuk Aslab:

Yang menurut saya sangat penting untuk dipertahankan: **Modul 3 jangan terlalu cepat memakai NumPy atau Pandas.** Mahasiswa perlu mengalami dulu bagaimana "repotnya" mengolah data secara manual menggunakan list, dictionary, loop, dan kondisi.

Baru pada Modul 5 mereka akan melihat:

"Oh, ternyata pekerjaan yang tadi membutuhkan 10 baris kode bisa dilakukan jauh lebih efektif dengan NumPy."

Dan ketika sampai Modul 7:

"Oh, ternyata dataset seperti list of dictionaries tadi dapat dikelola dengan jauh lebih kuat menggunakan Pandas."

Dengan pola tersebut, library tidak terasa sebagai **sihir**, tetapi sebagai alat untuk menyelesaikan masalah yang sebelumnya sudah mereka pahami secara algoritmik. Itu yang akan membuat transisi dari **mahasiswa belajar Python → mahasiswa belajar data science** jauh lebih masuk akal.

MODUL 4

Function: Membuat Program Menjadi Modular dan Reusable

Mata Kuliah	Praktikum Algoritma dan Pemrograman
Semester	1
Pertemuan	4
Durasi	100 menit
Platform	Google Colab
Bahasa	Python 3
Level	Pemula → pengolahan data terstruktur

Modul 4 menjadi titik penting dalam roadmap. Setelah mahasiswa belajar mengolah data menggunakan `list`, `dictionary`, `for`, dan `if` pada Modul 3, sekarang mereka belajar memindahkan proses-proses tersebut ke dalam **function**.

Peta jalan yang digunakan memang menempatkan **Modul 4 — Function** setelah **Modul 3 — List & Dictionary**, sebelum mahasiswa masuk ke **NumPy & Library** pada Modul 5.

Perubahan cara berpikir yang ingin dibangun adalah:

```
MODUL 3

  Data
   ↓
 Loop
   ↓
Condition
   ↓
Processing
   ↓
 Output
```

menjadi:

```
MODUL 4

  Data
   ↓
Function
   ↓
Processing
   ↓
 Return
   ↓
Data baru
```

Dengan kata lain, mahasiswa mulai belajar bahwa program tidak harus berupa satu blok kode panjang.

A. Gagasan Besar Modul

Bayangkan mahasiswa mempunyai kode:

```
nilai = [70, 80, 90, 65, 85]

total = 0

for n in nilai:
    total += n

rata_rata = total / len(nilai)

print(rata_rata)
```

Kode tersebut bekerja.

Tetapi bagaimana kalau kita ingin menghitung rata-rata untuk:

- kelas A,
- kelas B,
- kelas C,
- kelas D,
- data tahun 2024,
- data tahun 2025?

Apakah kode harus ditulis ulang?

Tidak.

Kita membuat:

```
def hitung_rata_rata(data):
    return sum(data) / len(data)
```

Kemudian:

```
nilai_a = [70, 80, 90, 65, 85]
nilai_b = [75, 88, 92, 70, 80]

print(hitung_rata_rata(nilai_a))
print(hitung_rata_rata(nilai_b))
```

Di sinilah konsep **reusability** mulai benar-benar terasa.

Think Python menyebut kemampuan membuat function sendiri sebagai salah satu fondasi programming, dan menjelaskan bahwa function dapat digunakan untuk mengenkapsulasi pekerjaan serta membuat program lebih mudah dikembangkan.

B. Tujuan Pembelajaran

Setelah menyelesaikan modul ini, mahasiswa mampu:

1. Menjelaskan konsep function.
2. Menjelaskan alasan penggunaan function.
3. Membuat function dengan `def`.
4. Memanggil function.
5. Menggunakan parameter.
6. Membedakan parameter dan argument.
7. Menggunakan positional argument.
8. Menggunakan keyword argument.
9. Menggunakan default parameter.
10. Menggunakan `return`.
11. Membedakan `print()` dan `return`.
12. Membuat function untuk pengolahan data.
13. Membuat function yang menerima list.
14. Membuat function yang menerima dictionary.
15. Membuat function yang mengembalikan list atau dictionary.
16. Membuat boolean function.
17. Memahami variabel lokal dalam function.
18. Menggabungkan beberapa function.
19. Melakukan refactoring program.
20. Membuat mini program analisis data yang modular.

C. Hubungan dengan Modul 3

Pada Modul 3 mahasiswa membuat:

```
mahasiswa = [
    {"nama": "Andi", "nilai": 80},
    {"nama": "Budi", "nilai": 65},
    {"nama": "Citra", "nilai": 90}
]
```

Kemudian:

```
total = 0

for m in mahasiswa:
    total += m["nilai"]
```

```
rata_rata = total / len(mahasiswa)
```

Pada Modul 4, kode tersebut diubah menjadi:

```
def hitung_rata_rata(data):
    total = 0

    for m in data:
        total += m["nilai"]

    return total / len(data)
```

Sekarang kita dapat menggunakan:

```
rata_rata = hitung_rata_rata(mahasiswa)
print(rata_rata)
```

Perubahannya bukan sekadar sintaks.

Mahasiswa mulai memahami:

Function adalah cara membungkus sebuah proses agar proses tersebut dapat digunakan kembali.

D. Struktur Praktikum

Waktu	Kegiatan
0–10 menit	Review Modul 3
10–25 menit	Konsep function
25–40 menit	Parameter dan argument
40–55 menit	Return value
55–65 menit	Boolean function
65–75 menit	Function dengan list/dictionary
75–85 menit	Function composition
85–95 menit	Mini project
95–100 menit	Exit ticket

E. Bagian 1 — Function Paling Sederhana

Mulai dengan:

```
def salam():
    print("Selamat belajar Python!")
```

Perhatikan bahwa function tersebut **belum dijalankan**.

Untuk menjalankannya:

```
salam()
```

Output:

```
Selamat belajar Python!
```

Think Python menjelaskan bahwa definisi function menentukan nama function dan rangkaian statement yang akan dijalankan ketika function dipanggil.

Strukturnya:

```
      def
       ↓
nama_function
       ↓
      ()
       ↓
      :
       ↓
body function
```

F. Definisi vs Pemanggilan Function

Ini harus diperjelas sejak awal.

```
def salam():
    print("Halo")
```

adalah **definisi**.

Sedangkan:

```
salam()
```

adalah **pemanggilan**.

Visualisasikan:

```
    DEFINISI
       ↓
Membuat function
       ↓
BELUM menjalankan

      CALL
```

```
        ↓
Menjalankan function
```

Kesalahan umum mahasiswa pemula adalah mengira `def` langsung menjalankan kode.

G. Mengapa Function?

Berikan kode:

```
print("Selamat datang")
print("Selamat datang")
print("Selamat datang")
print("Selamat datang")
```

Kemudian:

```
def selamat_datang():
    print("Selamat datang")
```

Lalu:

```
selamat_datang()
selamat_datang()
selamat_datang()
selamat_datang()
```

Tanyakan:

Mana yang lebih mudah dipelihara jika teks berubah menjadi “Selamat belajar”?

Inilah awal konsep:

- **reuse**
- **abstraction**
- **modularity**
- **maintainability**

H. Bagian 2 — Parameter

Sekarang function harus menerima data.

```
def salam(nama):
    print(f"Halo, {nama}!")
```

Panggil:

```
salam("Andi")
salam("Budi")
salam("Citra")
```

Output:

```
Halo, Andi!
Halo, Budi!
Halo, Citra!
```

`nama` disebut **parameter**.

Sedangkan `"Andi"` disebut **argument**.

Python Crash Course menjelaskan parameter sebagai informasi yang dibutuhkan function, sedangkan argument adalah informasi yang diberikan ketika function dipanggil.

Visual:

```
def salam(nama):
            ↑
        parameter


salam("Andi")
       ↑
    argument
```

Ini harus menjadi istilah wajib dalam Modul 4.

I. Multiple Parameters

```
def biodata(nama, umur):
    print(f"Nama: {nama}")
    print(f"Umur: {umur}")
```

Pemanggilan:

```
biodata("Andi", 20)
```

Output:

```
Nama: Andi
Umur: 20
```

Function dapat menerima lebih dari satu parameter.

J. Positional Argument

```
def biodata(nama, umur):
    print(nama, umur)

biodata("Andi", 20)
```

Python mencocokkan argument berdasarkan posisi.

```
nama ← "Andi"
umur ← 20
```

Tetapi:

```
biodata(20, "Andi")
```

secara sintaks tetap dapat berjalan, tetapi maknanya menjadi salah.

Ini contoh bagus untuk membedakan:

Syntax benar ≠ program benar.

K. Keyword Argument

Python juga memungkinkan:

```
biodata(
    nama="Andi",
    umur=20
)
```

Keuntungannya:

- lebih mudah dibaca;
- posisi tidak menjadi masalah;
- lebih jelas ketika parameter banyak.

Python Crash Course membahas positional arguments dan keyword arguments sebagai dua cara utama memberikan informasi kepada function.

L. Default Parameter

Misalnya:

```
def salam(nama, bahasa="Indonesia"):
```

```
    print(f"Halo {nama}, selamat belajar {bahasa}")
```

Panggil:

```
salam("Andi")
```

Output:

```
Halo Andi, selamat belajar Indonesia
```

Tetapi:

```
salam("Andi", "Python")
```

Output:

```
Halo Andi, selamat belajar Python
```

Default parameter membuat argument tertentu menjadi opsional. Python Crash Course secara eksplisit membahas penggunaan default value untuk membuat argument opsional.

M. Latihan 1

Buat function:

```
def luas_persegi(sisi):
    ...
```

Function harus menerima panjang sisi dan mengembalikan luas.

Uji:

```
print(luas_persegi(5))
print(luas_persegi(10))
```

Target:

```
25
100
```

Jangan langsung memberikan solusi kepada mahasiswa.

Minta mereka menyusun:

```
Input
  ↓
Process
  ↓
Output
```

terlebih dahulu.

N. Bagian 3 — `print()` vs `return`

Ini adalah **konsep terpenting Modul 4**.

Mulai dengan:

```
def luas_persegi(sisi):
    print(sisi * sisi)
```

Kemudian:

```
hasil = luas_persegi(5)
print(hasil)
```

Output:

```
25
None
```

Mengapa?

Karena function hanya melakukan:

```
print(...)
```

bukan:

```
return ...
```

Sekarang ubah:

```
def luas_persegi(sisi):
    return sisi * sisi
```

Kemudian:

```
hasil = luas_persegi(5)

print(hasil)
```

Output:

```
25
```

O. Konsep Return

`return` mengirimkan hasil dari function kembali kepada bagian program yang memanggilnya.

Visual:

```
          5
          ↓
    luas_persegi()
          ↓
        25
          ↓
        return
          ↓
        hasil
```

Think Python menjelaskan bahwa return value dapat diberikan kepada variabel atau digunakan sebagai bagian dari expression. Contoh `circle_area()` digunakan baik untuk assignment maupun sebagai bagian dari perhitungan lain.

P. Mengapa `return` Lebih Penting daripada `print()`?

Bandingkan:

```
def hitung_luas(sisi):
    print(sisi * sisi)
```

dengan:

```
def hitung_luas(sisi):
    return sisi * sisi
```

Yang kedua dapat digunakan:

```
luas = hitung_luas(5)

if luas > 20:
    print("Luas cukup besar")
```

Bahkan:

```
total = hitung_luas(5) + hitung_luas(10)
```

Sedangkan function yang hanya `print()` tidak memberikan nilai yang bisa digunakan kembali.

Prinsip praktis:

Gunakan `print()` ketika tujuan function memang menampilkan sesuatu.
Gunakan `return` ketika hasil function akan diproses lebih lanjut.

Q. Latihan 2 — Function Matematika

Buat:

```
def luas_persegi(sisi):
    ...

def luas_persegi_panjang(panjang, lebar):
    ...

def luas_lingkaran(radius):
    ...
```

Kemudian:

```
a = luas_persegi(5)
b = luas_persegi_panjang(10, 5)
c = luas_lingkaran(7)
```

Mahasiswa mulai terbiasa bahwa function menghasilkan **data**, bukan sekadar menampilkan teks.

R. Function sebagai Unit Algoritma

Sekarang kita mulai mengubah algoritma Modul 3.

Kode awal:

```
nilai = [70, 80, 90, 65, 85]

total = 0

for n in nilai:
    total += n

rata_rata = total / len(nilai)

print(rata_rata)
```

Refactoring:

```
def hitung_rata_rata(nilai):
    total = 0

    for n in nilai:
        total += n

    return total / len(nilai)
```

Kemudian:

```
nilai = [70, 80, 90, 65, 85]

rata_rata = hitung_rata_rata(nilai)

print(rata_rata)
```

Sekarang algoritma rata-rata mempunyai nama:

```
hitung_rata_rata()
```

Itulah **abstraction**.

S. Function dengan List

```
def nilai_tertinggi(data):
    tertinggi = data[0]

    for n in data:
        if n > tertinggi:
            tertinggi = n

    return tertinggi
```

Uji:

```
nilai = [70, 85, 90, 65, 88]

print(nilai_tertinggi(nilai))
```

Output:

```
90
```

Perhatikan bahwa mahasiswa sekarang memiliki algoritma yang reusable.

T. Function untuk Filtering

Dari Modul 3:

```
nilai = [45, 60, 75, 90, 55, 80]

lulus = []

for n in nilai:
    if n >= 60:
        lulus.append(n)
```

Sekarang:

```
def filter_lulus(nilai):
    lulus = []

    for n in nilai:
        if n >= 60:
            lulus.append(n)

    return lulus
```

Gunakan:

```
hasil = filter_lulus(nilai)

print(hasil)
```

Output:

```
[60, 75, 90, 80]
```

Di sini mahasiswa mulai melihat bahwa function dapat **menerima data dan menghasilkan data baru**.

Ini sangat penting untuk persiapan Pandas nanti.

U. Function dengan Dictionary

Gunakan dataset Modul 3:

```
mahasiswa = {
    "nama": "Andi",
    "uts": 80,
    "uas": 90
}
```

Buat:

```
def hitung_nilai_akhir(data):
    return data["uts"] * 0.4 + data["uas"] * 0.6
```

Kemudian:

```
nilai_akhir = hitung_nilai_akhir(mahasiswa)

print(nilai_akhir)
```

Output:

```
86.0
```

Sekarang function memahami struktur data yang sudah dipelajari pada Modul 3.

V. Function untuk Menentukan Grade

```
def tentukan_grade(nilai):
    if nilai >= 85:
        return "A"
    elif nilai >= 75:
        return "B"
    elif nilai >= 65:
        return "C"
    elif nilai >= 60:
        return "D"
    else:
        return "E"
```

Kemudian:

```
grade = tentukan_grade(86)

print(grade)
```

Output:

```
A
```

Perhatikan bahwa function ini hanya mempunyai satu tugas:

Mengubah nilai menjadi grade.

Ini merupakan prinsip penting:

Satu function sebaiknya memiliki tanggung jawab yang jelas.

W. Function Composition

Sekarang gabungkan:

```
nilai_akhir = hitung_nilai_akhir(mahasiswa)
grade = tentukan_grade(nilai_akhir)

print(nilai_akhir)
print(grade)
```

Visual:

```
Data Mahasiswa
      ↓
hitung_nilai_akhir()
      ↓
Nilai Akhir
      ↓
tentukan_grade()
      ↓
Grade
```

Program mulai dibangun dari function-function kecil.

Think Python Chapter 4 menekankan desain interface function dan bagaimana function dapat dibangun dari function lain. Buku tersebut bahkan menggunakan function seperti `polyline`, `arc`, dan `circle` untuk menunjukkan bagaimana program dapat disusun dari komponen yang lebih kecil.

X. Boolean Function

Function tidak harus mengembalikan angka atau string.

Ia dapat mengembalikan:

```
True
```

atau:

```
False
```

Contoh:

```
def lulus(nilai):
    return nilai >= 60
```

Uji:

```
print(lulus(80))
print(lulus(50))
```

Output:

```
True
False
```

Kemudian:

```
if lulus(80):
    print("Mahasiswa lulus")
```

Think Python menunjukkan boolean function seperti `is_divisible()` yang mengembalikan `True` atau `False`, dan kemudian hasilnya langsung digunakan dalam conditional statement.

Ini pola yang sangat penting:

```
function
 ↓
True / False
 ↓
if
```

Y. Latihan 3 — Boolean Function

Buat:

```
def is_genap(angka):
    ...
```

Kemudian:

```
print(is_genap(10))
print(is_genap(7))
```

Buat juga:

```
def is_lulus(nilai):
    ...
```

dan:

```
def is_genap(angka):
    ...
```

Kemudian gabungkan:

```
if is_lulus(nilai) and is_genap(nilai):
    ...
```

Mahasiswa mulai belajar bahwa function dapat menjadi bagian dari ekspresi logika.

Z. Variabel Lokal

Perhatikan:

```
def hitung():
    x = 10
    print(x)

hitung()
```

Tetapi:

```
print(x)
```

akan menghasilkan:

```
NameError
```

Mengapa?

Karena `x` dibuat di dalam function.

```
GLOBAL
──────────────────

hitung()
   │
   └── x = 10
```

`x` merupakan **variabel lokal**.

Think Python membahas parameter dan variabel lokal sebagai bagian penting dari cara function bekerja, termasuk bagaimana variabel dalam function memiliki ruang lingkupnya sendiri.

Untuk mahasiswa semester awal, tidak perlu masuk terlalu jauh ke LEGB secara formal. Cukup tanamkan:

Variabel yang dibuat di dalam function tidak otomatis menjadi variabel global.

AA. Kesalahan Umum: Mengandalkan Variabel Global

Contoh buruk:

```
nilai = [70, 80, 90]

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

Lebih baik:

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

Kemudian:

```
rata_rata(nilai)
```

Mengapa lebih baik?

Karena function kedua tidak bergantung pada variabel tertentu di luar dirinya.

Ia menjadi lebih **general**.

Think Python membahas *generalization* sebagai salah satu tujuan penting dalam perancangan function: function sebaiknya dibuat cukup umum sehingga dapat digunakan untuk berbagai kasus, bukan hanya satu kasus tertentu.

AB. Dari Spesifik ke General

Kode yang terlalu spesifik:

```
def hitung():
    nilai = [70, 80, 90]
    return sum(nilai) / len(nilai)
```

Hanya bisa digunakan untuk satu dataset.

Lebih baik:

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

Sekarang:

```
hitung_rata_rata([70, 80, 90])
hitung_rata_rata([60, 75, 88])
hitung_rata_rata([90, 95, 100])
```

Satu function → banyak data.

Inilah yang dimaksud **generalization**.

AC. Function dengan List of Dictionaries

Ini bagian yang sengaja menghubungkan Modul 3 dan Modul 4.

Gunakan:

```
mahasiswa = [
    {"nama": "Andi", "nilai": 80},
    {"nama": "Budi", "nilai": 65},
    {"nama": "Citra", "nilai": 90},
    {"nama": "Dina", "nilai": 55}
]
```

Buat function:

```
def mahasiswa_lulus(data):
    hasil = []

    for m in data:
        if m["nilai"] >= 60:
            hasil.append(m)

    return hasil
```

Kemudian:

```
lulus = mahasiswa_lulus(mahasiswa)

print(lulus)
```

Sekarang function mengubah:

```
Dataset
   ↓
Filtering
   ↓
Dataset baru
```

Ini sudah merupakan bentuk sederhana dari **data transformation**.

AD. Function Mengembalikan Dictionary

Misalnya kita ingin membuat ringkasan.

```
def ringkasan_nilai(data):
    hasil = {
        "jumlah": len(data),
        "tertinggi": max(data),
        "terendah": min(data),
        "rata_rata": sum(data) / len(data)
    }

    return hasil
```

Gunakan:

```
nilai = [70, 80, 90, 65, 85]

ringkasan = ringkasan_nilai(nilai)

print(ringkasan)
```

Hasil:

```
{
    'jumlah': 5,
    'tertinggi': 90,
    'terendah': 65,
    'rata_rata': 78.0
}
```

Perhatikan transformasinya:

```
LIST
 ↓
FUNCTION
 ↓
DICTIONARY
```

Mahasiswa mulai memahami bahwa function dapat menjadi **mesin transformasi data**.

AE. Function Mengembalikan Banyak Nilai

Python memungkinkan:

```
def statistik(data):
    return min(data), max(data), sum(data) / len(data)
```

Kemudian:

```
minimum, maksimum, rata_rata = statistik(
    [70, 80, 90, 65, 85]
)
```

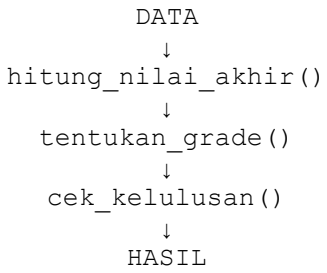
Ini sebenarnya memanfaatkan tuple secara implisit.

Karena tuple baru saja diperkenalkan sebagai materi pengayaan pada Modul 3, bagian ini cocok sebagai penghubung.

Think Python juga membahas *packing* dan *unpacking* tuple sebagai cara mengelola beberapa nilai sekaligus.

AF. Function sebagai Pipeline Data

Sekarang kita dapat membuat pipeline:



Contoh:

```
def hitung_nilai_akhir(uts, uas):
    return uts * 0.4 + uas * 0.6


def tentukan_grade(nilai):
    if nilai >= 85:
        return "A"
    elif nilai >= 75:
        return "B"
```

```
    elif nilai >= 65:
        return "C"
    elif nilai >= 60:
        return "D"
    else:
        return "E"


def lulus(nilai):
    return nilai >= 60
```

Kemudian:

```
nilai = hitung_nilai_akhir(80, 90)
grade = tentukan_grade(nilai)

print(nilai)
print(grade)
print(lulus(nilai))
```

Ini jauh lebih terstruktur daripada satu blok program besar.

AG. Refactoring

Sekarang berikan mahasiswa program Modul 3 yang panjang.

Misalnya:

```
mahasiswa = [
    {"nama": "Andi", "uts": 80, "uas": 85},
    {"nama": "Budi", "uts": 70, "uas": 65},
    {"nama": "Citra", "uts": 90, "uas": 95},
    {"nama": "Dina", "uts": 60, "uas": 70}
]

for m in mahasiswa:
    nilai = m["uts"] * 0.4 + m["uas"] * 0.6

    if nilai >= 85:
        grade = "A"
    elif nilai >= 75:
        grade = "B"
    elif nilai >= 65:
        grade = "C"
    elif nilai >= 60:
        grade = "D"
    else:
        grade = "E"

    print(m["nama"], nilai, grade)
```

Tugas mahasiswa:

Jangan mengubah hasil program. Ubah struktur program menggunakan function.

Hasil yang diharapkan:

```
def hitung_nilai_akhir(uts, uas):
    return uts * 0.4 + uas * 0.6


def tentukan_grade(nilai):
    if nilai >= 85:
        return "A"
    elif nilai >= 75:
        return "B"
    elif nilai >= 65:
        return "C"
    elif nilai >= 60:
        return "D"
    else:
        return "E"
```

Kemudian:

```
for m in mahasiswa:
    nilai = hitung_nilai_akhir(m["uts"], m["uas"])
    grade = tentukan_grade(nilai)

    print(m["nama"], nilai, grade)
```

Inilah **refactoring**.

Think Python menjelaskan refactoring sebagai perubahan struktur kode yang memperbaiki organisasi program tanpa mengubah perilakunya.

AH. Debugging Function

Function juga menghasilkan error.

Contoh:

```
def hitung_luas(panjang, lebar):
    return panjang * lebar

print(hitung_luas(10))
```

Python akan memberi informasi bahwa ada argument yang kurang.

Python Crash Course menunjukkan bahwa traceback Python dapat memberi tahu function yang bermasalah sekaligus nama argument yang belum diberikan.

Mahasiswa harus belajar membaca:

```
TypeError
    ↓
function mana?
    ↓
berapa argument dibutuhkan?
    ↓
berapa argument diberikan?
```

AI. Kesalahan yang Lebih Berbahaya

Tidak semua kesalahan function menghasilkan error.

Contoh:

```
def hitung_luas(panjang, lebar):
    return panjang + lebar
```

Program berjalan.

Tetapi hasilnya salah.

Ini:

```
Syntax error    → kode tidak dapat dijalankan
Runtime error   → program berhenti ketika berjalan
Logic error     → program berjalan tetapi hasil salah
```

Modul 4 harus memperkuat kebiasaan debugging yang sudah diperkenalkan pada Modul 1.

AJ. Incremental Development

Jangan langsung membuat function besar.

Misalnya target:

Membuat function analisis nilai mahasiswa.

Jangan langsung menulis 50 baris.

Mulai:

```
def hitung_nilai_akhir(uts, uas):
    return uts * 0.4 + uas * 0.6
```

Tes:

```
print(hitung_nilai_akhir(80, 90))
```

Kemudian tambahkan:

```
def tentukan_grade(nilai):
    ...
```

Tes.

Kemudian:

```
def lulus(nilai):
    ...
```

Tes.

Baru digabungkan.

Think Python secara eksplisit menyarankan **incremental development**: mulai dengan program yang bekerja, lakukan perubahan kecil, dan uji setelah setiap perubahan. Buku tersebut juga menyarankan penggunaan variabel sementara dan print sebagai *scaffolding* ketika melakukan debugging, lalu menghapusnya setelah program selesai.

AK. Mini Challenge — Function Statistik

Buat function:

```
def statistik_nilai(data):
    ...
```

Function harus mengembalikan:

```
jumlah data
nilai minimum
nilai maksimum
rata-rata
```

Contoh:

```
hasil = statistik_nilai([70, 80, 90, 60, 85])

print(hasil)
```

Kemudian buat versi yang menghasilkan dictionary:

```
{
    "jumlah": ...,
    "minimum": ...,
    "maksimum": ...,
    "rata_rata": ...
}
```

Ini latihan yang sangat penting karena mahasiswa belajar bahwa desain output function juga merupakan bagian dari desain algoritma.

AL. Mini Project Modul 4

“Reusable Student Data Analyzer”

Gunakan dataset:

```
mahasiswa = [
    {"nama": "Andi", "uts": 80, "uas": 85},
    {"nama": "Budi", "uts": 70, "uas": 65},
    {"nama": "Citra", "uts": 90, "uas": 95},
    {"nama": "Dina", "uts": 60, "uas": 70},
    {"nama": "Eko", "uts": 75, "uas": 80},
    {"nama": "Fajar", "uts": 50, "uas": 55}
]
```

Mahasiswa wajib membuat minimal:

```
hitung_nilai_akhir()
tentukan_grade()
cek_kelulusan()
hitung_rata_rata()
nilai_tertinggi()
nilai_terendah()
filter_lulus()
```

Kemudian function-function tersebut digunakan untuk menghasilkan laporan.

AM. Output yang Diharapkan

Program menghasilkan semacam:

```
================================
ANALISIS NILAI MAHASISWA
================================

Andi
```

```
Nilai Akhir : 83.0
Grade       : B
Status      : Lulus

Budi
Nilai Akhir : 67.0
Grade       : C
Status      : Lulus

...

================================
RINGKASAN KELAS
================================

Jumlah mahasiswa : 6
Rata-rata        : 74.83
Nilai tertinggi  : 93.00
Nilai terendah   : 53.00
Jumlah lulus     : 5
Jumlah tidak lulus : 1
```

Yang dinilai bukan hanya output.

Mahasiswa harus mampu menunjukkan bahwa programnya tersusun dari function-function yang memiliki tugas jelas.

AN. Challenge untuk Mahasiswa Lebih Maju

Buat:

```
def analisis_mahasiswa(data):
    ...
```

Function tersebut mengembalikan dictionary:

```
{
    "jumlah": ...,
    "rata_rata": ...,
    "tertinggi": ...,
    "terendah": ...,
    "jumlah_lulus": ...,
    "jumlah_tidak_lulus": ...
}
```

Tetapi ada aturan:

`analisis_mahasiswa()` tidak boleh mengerjakan semua perhitungan sendiri.

Ia harus memanggil function lain.

Misalnya:

```
analisis_mahasiswa()
        │
        ├── hitung_rata_rata()
        ├── nilai_tertinggi()
        ├── nilai_terendah()
        ├── hitung_lulus()
        └── hitung_tidak_lulus()
```

Ini mulai mengenalkan **hierarki function**.

AO. Function sebagai Fondasi Data Science

Di sinilah hubungan Modul 4 dengan roadmap data science harus dibuat eksplisit.

Mahasiswa sekarang sudah memiliki:

```
LIST
DICTIONARY
LOOP
IF
FUNCTION
```

Mereka dapat membuat:

```
Function untuk membersihkan data
Function untuk menghitung statistik
Function untuk melakukan filtering
Function untuk transformasi data
Function untuk validasi data
Function untuk membuat kategori
```

Contohnya:

```
def bersihkan_nama(nama):
    return nama.strip().title()
```

atau:

```
def validasi_nilai(nilai):
    return 0 <= nilai <= 100
```

atau:

```
def kategori_nilai(nilai):
    if nilai >= 80:
        return "Tinggi"
    elif nilai >= 60:
        return "Sedang"
    else:
        return "Rendah"
```

Ini merupakan embrio dari **data preprocessing**.

Nanti ketika mahasiswa belajar Pandas, konsepnya berubah menjadi operasi terhadap kolom dan DataFrame. Tetapi pola berpikirnya sudah mereka kenal.

AP. AI dalam Modul 4

Pada modul ini AI dapat digunakan lebih serius, tetapi tetap dengan aturan.

Contoh tugas:

Saya memiliki function berikut. Jangan perbaiki kodenya. Buatkan lima test case yang dapat digunakan untuk menguji function tersebut.

Atau:

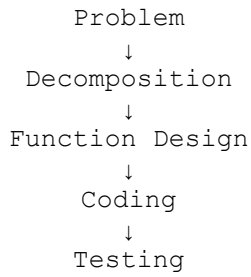
Jelaskan mengapa function saya menghasilkan `None`. Jangan berikan kode perbaikannya.

Atau:

Saya ingin memecah program analisis nilai menjadi beberapa function. Jangan tuliskan programnya. Bantu saya menentukan tanggung jawab masing-masing function.

Ini sangat cocok dengan pendekatan Think Python yang dalam bagian latihan bahkan mendorong penggunaan virtual assistant untuk membantu memecah persoalan besar menjadi beberapa function kecil dan mengujinya.

Jadi mulai Modul 4, mahasiswa dapat diarahkan menggunakan AI sebagai:



```
        ↓
   Debugging
```

bukan:

```
      Problem
        ↓
"AI, buatkan kode"
        ↓
    Copy Paste
```

AQ. Tugas Praktikum

“Reusable Sales Analyzer”

Gunakan dataset Modul 3:

```
transaksi = [
    {"produk": "Buku", "kategori": "Pendidikan", "harga": 50000,
"jumlah": 2},
    {"produk": "Pensil", "kategori": "ATK", "harga": 5000,
"jumlah": 10},
    {"produk": "Buku", "kategori": "Pendidikan", "harga": 50000,
"jumlah": 3},
    {"produk": "Pulpen", "kategori": "ATK", "harga": 7000,
"jumlah": 5},
    {"produk": "Buku", "kategori": "Pendidikan", "harga": 50000,
"jumlah": 1}
]
```

Buat function:

```
hitung_total_transaksi()
hitung_total_penjualan()
produk_terlaris()
produk_omzet_terbesar()
filter_kategori()
```

Program utama harus menggunakan function-function tersebut.

Tidak boleh membuat satu blok program panjang.

AR. Ketentuan Tugas

Mahasiswa wajib:

1. menggunakan minimal 5 function;
2. setiap function memiliki satu tugas utama;

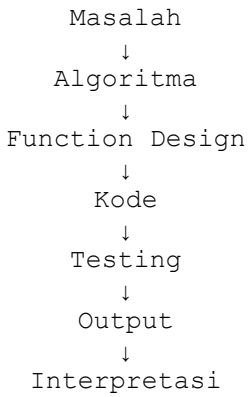
3. menggunakan parameter;
4. menggunakan `return`;
5. minimal satu function mengembalikan dictionary;
6. minimal satu function mengembalikan list;
7. minimal satu boolean function;
8. melakukan pengujian terhadap function;
9. memberikan minimal dua test case untuk setiap function penting;
10. menuliskan interpretasi hasil.

AS. Format Notebook Google Colab

```
MODUL 4
FUNCTION

Nama:
NIM:
Kelas:

1. Tujuan Praktikum

2. Review Modul 3

3. Function Dasar
   3.1 Definisi
   3.2 Pemanggilan
   3.3 Parameter
   3.4 Argument

4. Return
   4.1 print vs return
   4.2 Return angka
   4.3 Return string
   4.4 Return list
   4.5 Return dictionary

5. Boolean Function

6. Function dengan List

7. Function dengan Dictionary

8. Function Composition

9. Refactoring

10. Debugging Function

11. Mini Project

12. Testing

13. Interpretasi Hasil
```

14. Refleksi

Struktur notebook tetap mengikuti pola:



AT. Rubrik Penilaian

Komponen	Bobot
Konsep function	10%
Parameter & argument	10%
Return	15%
Function dengan list/dictionary	15%
Boolean function	10%
Function composition	10%
Refactoring	10%
Testing & debugging	10%
Mini project	10%
Total	**100%**

AU. Bacaan dari Dua Buku

Untuk **Python Crash Course, 3rd Edition**, bacaan utama adalah:

Chapter 8 — Functions

Fokus:

- defining a function;
- passing information;
- arguments dan parameters;
- positional arguments;
- keyword arguments;
- default values;

- return values;
- passing list;
- passing arbitrary number of arguments.

Buku tersebut secara eksplisit menyusun Chapter 8 dari definisi function, parameter/argument, berbagai cara passing argument, default argument, hingga return values.

Untuk modul ini **belum perlu masuk mendalam ke Chapter 9 tentang classes**.

Untuk **Think Python, 3rd Edition**, justru bacaan perlu dibuat agak tersebar karena buku tersebut membangun konsep function secara bertahap.

Chapter 3 — Functions

Pelajari:

- defining new functions;
- function calls;
- parameters;
- function composition;
- local variables;
- stack diagrams;
- traceback.

Chapter 3 memperkenalkan function sebagai fondasi programming dan membahas bagaimana satu function dapat memanggil function lain.

Chapter 4 — Functions and Interfaces

Pelajari:

- function interface;
- parameter;
- function composition;
- generalization;
- refactoring.

Chapter 4 sangat relevan untuk filosofi desain Modul 4 karena buku tersebut menggunakan function kecil yang kemudian digabungkan menjadi function yang lebih kompleks, serta membahas refactoring.

Chapter 6 — Return Values

Ini bacaan **sangat penting**.

Pelajari:

- return value;
- boolean functions;
- function sebagai bagian expression;
- return beberapa nilai;
- debugging function.

Think Python menjelaskan bahwa return value dapat disimpan dalam variabel maupun digunakan langsung dalam expression.

AV. Posisi Modul 4 dalam Roadmap

Setelah empat modul, kemampuan mahasiswa sudah berkembang seperti ini:

```
MODUL 1
Algoritma
Variabel
Tipe Data
Ekspresi
Problem Solving
Debugging
    ↓
MODUL 2
Percabangan & Perulangan
    ↓
MODUL 3
List & Dictionary
    ↓
MODUL 4
Function
    ↓
```

Dan sekarang mahasiswa sudah dapat berpikir:

```
DATA
  ↓
STRUCTURE
  ↓
ALGORITHM
  ↓
FUNCTION
  ↓
REUSABLE PROCESS
  ↓
RESULT
```

Barulah setelah ini **Modul 5: NumPy & Library** menjadi masuk akal.

Karena pada Modul 5 mahasiswa akan mulai melihat sesuatu yang menarik:

```
Yang sebelumnya dibuat manual
            ↓
         Function
            ↓
         Library
            ↓
          NumPy
```

Catatan Aslab:

Mereka akan belajar bahwa library bukan pengganti pemahaman algoritma. Library adalah **abstraksi dari algoritma dan operasi yang sudah mereka pahami**.

Itu juga alasan saya menyarankan agar Modul 4 tidak terlalu cepat memasukkan `lambda`, decorator, recursion, `*args/**kwargs`, atau pembuatan package. Materi tersebut memang ada dalam ekosistem Python, tetapi belum diperlukan untuk tujuan roadmap ini. Fokus Modul 4 sebaiknya sangat kuat pada **parameter → return → modularisasi → function composition → refactoring → data processing**.

Dengan fondasi ini, ketika masuk **Modul 5 — NumPy & Library**, mahasiswa sudah siap berpindah dari "saya bisa membuat algoritma" ke "saya bisa menggunakan alat komputasi yang tepat untuk mengolah data."

MODUL 5

NumPy dan Library Python untuk Pengolahan Data

Mata Kuliah	Praktikum Algoritma dan Pemrograman
Semester	1
Pertemuan	5 dari 10
Durasi	100 menit
Platform	Google Colab
Bahasa	Python 3
Level	Pemula → pengantar scientific computing dan data science

Modul 5 merupakan **titik transisi** dalam mata kuliah ini.

Pada Modul 1–4 mahasiswa belajar membangun logika program sendiri:

```
Algoritma
    ↓
Percabangan
    ↓
Perulangan
    ↓
List & Dictionary
    ↓
Function
```

Mulai Modul 5, mahasiswa diperkenalkan pada gagasan baru:

Tidak semua algoritma harus kita bangun dari nol.

Python memiliki ekosistem library yang memungkinkan pekerjaan komputasi dilakukan dengan lebih efektif.

Fokus pertama adalah **NumPy (Numerical Python)**.

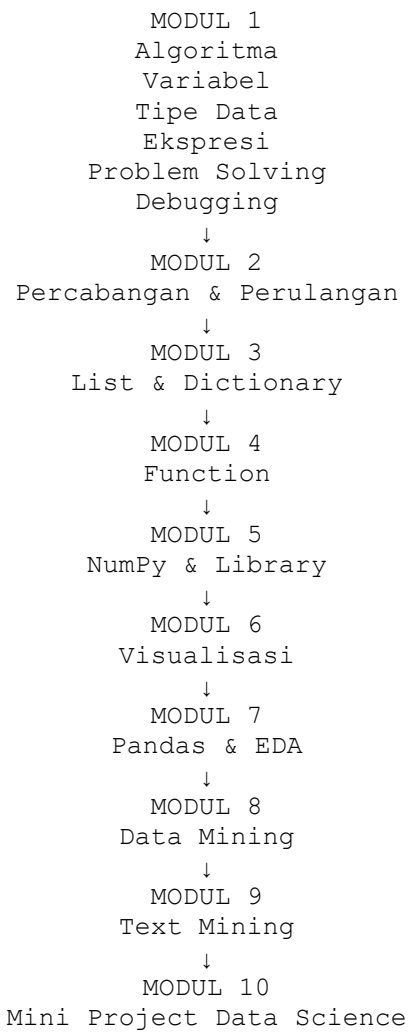
Namun ada satu prinsip yang harus dijaga:

Mahasiswa harus memahami algoritma manual terlebih dahulu, kemudian melihat bagaimana library membuatnya lebih sederhana dan efisien.

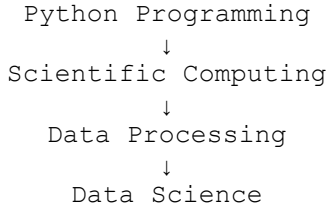
Dengan demikian NumPy tidak diperkenalkan sebagai kumpulan fungsi yang harus dihafalkan.

A. Posisi Modul 5 dalam Roadmap

Roadmap praktikum:



Mulai modul ini mahasiswa mulai memasuki wilayah:



B. Tujuan Pembelajaran

Setelah menyelesaikan modul ini, mahasiswa mampu:

1. Menjelaskan konsep library Python.
2. Menggunakan `import`.
3. Menjelaskan perbedaan library dan fungsi yang dibuat sendiri.
4. Mengenal NumPy sebagai library komputasi numerik.
5. Membuat NumPy array.
6. Mengakses elemen array.
7. Memahami dimensi array.
8. Membuat array satu dimensi dan dua dimensi.
9. Menggunakan `shape`, `size`, dan `dtype`.
10. Melakukan operasi aritmetika pada array.
11. Memahami perbedaan operasi list dan array.
12. Menggunakan fungsi statistik NumPy.
13. Menggunakan slicing pada array.
14. Melakukan filtering sederhana terhadap array.
15. Menggunakan operasi agregasi.
16. Memahami konsep *vectorization* secara sederhana.
17. Membandingkan pendekatan loop dengan pendekatan NumPy.
18. Mulai memahami mengapa library penting dalam data science.

C. Mengapa Tidak Langsung Pandas?

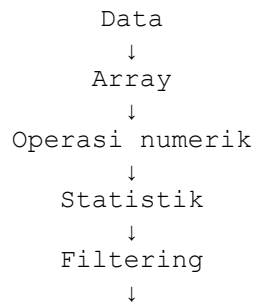
Mahasiswa mungkin bertanya:

“Kalau tujuan akhirnya data science, mengapa tidak langsung Pandas?”

Jawabannya pedagogis.

Pandas memang sangat kuat untuk mengolah dataset, tetapi di balik banyak operasi data terdapat konsep numerik dan array.

Kita ingin mahasiswa memahami terlebih dahulu:



```
Vectorization
```

Baru kemudian:

```
DataFrame
    ↓
  Pandas
    ↓
   EDA
```

Dengan cara ini mahasiswa tidak sekadar mengetahui bahwa:

```
df["nilai"].mean()
```

menghasilkan rata-rata.

Mereka memahami terlebih dahulu apa yang sebenarnya dimaksud dengan **mean**, bagaimana menghitungnya, dan bagaimana komputer memproses sekumpulan angka.

D. Skenario Praktikum 100 Menit

Waktu	Kegiatan
0–10 menit	Review Function
10–20 menit	Konsep library dan `import`
20–35 menit	NumPy array
35–50 menit	Operasi array
50–65 menit	Statistik NumPy
65–75 menit	Filtering dan Boolean array
75–85 menit	Vectorization
85–95 menit	Data Challenge
95–100 menit	Exit ticket

E. Bagian 1 — Apa Itu Library?

Selama Modul 1–4 mahasiswa membuat sendiri:

```
def hitung_rata_rata(data):
    return sum(data) / len(data)
```

Sekarang kita menemukan bahwa Python dan ekosistemnya sudah menyediakan berbagai fungsi.

Misalnya:

```
import math
```

Kemudian:

```
print(math.sqrt(25))
```

Hasil:

```
5.0
```

Jelaskan:

```
library
   ↓
module
   ↓
function
```

Mahasiswa tidak perlu menghafalkan terminologi secara terlalu formal, tetapi perlu memahami bahwa library berisi kode yang dapat kita manfaatkan kembali.

F. Menggunakan `import`

Cara sederhana:

```
import math

print(math.sqrt(25))
```

Atau:

```
from math import sqrt

print(sqrt(25))
```

Untuk praktikum ini, biasakan bentuk:

```
import numpy as np
```

Kemudian gunakan:

```
np.sqrt(25)
```

`np` adalah alias yang umum digunakan untuk NumPy.

G. Mengapa Menggunakan Library?

Bandingkan.

Tanpa library:

```
akar = 25 ** 0.5
```

Dengan library:

```
import math

akar = math.sqrt(25)
```

Keduanya dapat menghasilkan nilai yang sama.

Tetapi library menyediakan banyak operasi yang sudah diuji dan dioptimalkan.

Untuk data science, library menjadi sangat penting karena kita akan bekerja dengan:

- ribuan;
- jutaan;
- bahkan jutaan baris data.

Kita tidak ingin menulis seluruh algoritma dasar dari nol setiap kali melakukan analisis.

H. Bagian 2 — Mengenal NumPy

Di Google Colab:

```
import numpy as np
```

Kemudian:

```
print(np.__version__)
```

Mahasiswa dapat melihat versi NumPy yang tersedia di lingkungan Colab.

Tidak perlu menginstal NumPy secara manual.

I. List vs NumPy Array

Mulai dengan list:

```
nilai = [70, 80, 90, 60]
```

Kemudian:

```
nilai_array = np.array(nilai)
```

Sekarang:

```
print(nilai_array)
```

Output kira-kira:

```
[70 80 90 60]
```

Konsep:

```
Python List
     ↓
np.array()
     ↓
NumPy Array
```

J. Mengapa Array?

Perhatikan:

```
nilai = [70, 80, 90]
```

Jika kita melakukan:

```
print(nilai * 2)
```

hasilnya adalah pengulangan list:

```
[70, 80, 90, 70, 80, 90]
```

Bukan:

```
[140, 160, 180]
```

Sekarang:

```
nilai = np.array([70, 80, 90])

print(nilai * 2)
```

Hasil:

```
[140 160 180]
```

Ini adalah salah satu perbedaan paling penting.

Pada NumPy:

Operasi dapat diterapkan kepada seluruh array sekaligus.

K. Konsep Vectorization

Inilah istilah yang mulai diperkenalkan:

vectorization

Tanpa NumPy:

```
nilai = [70, 80, 90]

hasil = []

for n in nilai:
    hasil.append(n * 2)
```

Dengan NumPy:

```
nilai = np.array([70, 80, 90])

hasil = nilai * 2
```

Secara konsep:

```
LIST
  ↓
for
  ↓
elemen satu per satu
  ↓
hasil
```

sedangkan NumPy:

```
ARRAY
  ↓
```

```
OPERASI
   ↓
SELURUH ELEMEN
```

Jangan mengatakan kepada mahasiswa bahwa NumPy “tidak menggunakan loop”. Secara implementasi internal tetap ada operasi berulang, tetapi banyak pekerjaan dilakukan oleh operasi array yang dioptimalkan di balik layar.

Yang perlu dipahami mahasiswa:

Kita tidak perlu selalu menulis loop Python secara eksplisit untuk operasi numerik terhadap array.

L. Operasi Aritmetika Array

```
import numpy as np

nilai = np.array([70, 80, 90, 60])
```

Coba:

```
print(nilai + 5)
print(nilai - 5)
print(nilai * 2)
print(nilai / 2)
```

Hasil:

```
[75 85 95 65]
[65 75 85 55]
[140 160 180 120]
[35. 40. 45. 30.]
```

Kemudian:

```
print(nilai ** 2)
```

M. Operasi Antararray

Misalnya:

```
uts = np.array([70, 80, 90, 60])
uas = np.array([80, 85, 95, 70])
```

Hitung rata-rata dua nilai:

```
rata_rata = (uts + uas) / 2
```

```
print(rata_rata)
```

Hasil:

```
[75.  82.5 92.5 65. ]
```

Ini sangat menarik untuk mahasiswa.

Kita tidak menulis:

```
for i in range(...):
```

tetapi NumPy melakukan operasi elemen demi elemen.

N. Menghitung Nilai Akhir

Gunakan bobot:

```
UTS = 40%
UAS = 60%
```

Kode:

```
nilai_akhir = uts * 0.4 + uas * 0.6

print(nilai_akhir)
```

Ini sudah menjadi contoh nyata pengolahan data numerik.

O. Array Satu Dimensi

```
nilai = np.array([70, 80, 90, 60, 85])
```

Akses:

```
print(nilai[0])
print(nilai[2])
print(nilai[-1])
```

Konsep indexing tetap sama dengan list.

Ini sengaja dilakukan agar mahasiswa melihat:

NumPy tidak membuang konsep yang sudah dipelajari. Ia memperluasnya.

P. Slicing Array

```
print(nilai[1:4])
```

Misalnya menghasilkan:

```
[80 90 60]
```

Juga:

```
print(nilai[:3])
print(nilai[2:])
```

Konsep slicing yang dipelajari pada Modul 3 tetap berlaku.

Q. Array Dua Dimensi

Sekarang mulai menarik.

Buat data nilai:

```
nilai = np.array([
    [80, 85, 90],
    [70, 75, 80],
    [90, 95, 88]
])
```

Visual:

```
           UTS  UAS  Tugas
Andi        80   85    90
Budi        70   75    80
Citra       90   95    88
```

Secara matematis kita dapat melihatnya sebagai matriks.

R. `shape`

```
print(nilai.shape)
```

Hasil:

```
(3, 3)
```

Artinya:

```
3 baris
3 kolom
```

Ini merupakan konsep penting untuk data science.

Mahasiswa mulai mengenal:

dimensi data

S. `ndim` dan `size`

```
print(nilai.ndim)
print(nilai.size)
```

Hasil:

```
2
9
```

Artinya:

- `ndim` → jumlah dimensi;
- `size` → jumlah seluruh elemen.

Kemudian:

```
print(nilai.dtype)
```

Mahasiswa mulai diperkenalkan dengan konsep tipe data array.

T. Mengakses Baris

```
print(nilai[0])
```

Hasil:

```
[80 85 90]
```

Mengakses elemen:

```
print(nilai[0, 1])
```

Hasil:

```
85
```

Format:

```
array[baris, kolom]
```

Visual:

```
nilai[0,1]

          kolom
            ↓
[80  85  90]
 ↑
baris
```

U. Mengakses Kolom

Ambil kolom kedua:

```
print(nilai[:, 1])
```

Hasil:

```
[85 75 95]
```

Ini konsep penting.

```
:
```

berarti:

ambil seluruh baris.

Sehingga:

```
nilai[:, 1]
```

berarti:

semua baris pada kolom indeks 1.

V. Latihan — Data Nilai Mahasiswa

Gunakan:

```
nilai = np.array([
    [80, 85, 90],
    [70, 75, 80],
```

```
    [90, 95, 88],
    [60, 70, 75],
    [85, 80, 90]
])
```

Mahasiswa diminta mencari:

1. semua nilai mahasiswa pertama;
2. semua nilai UTS;
3. semua nilai UAS;
4. nilai UAS mahasiswa ketiga;
5. dua mahasiswa pertama;
6. dua kolom pertama.

W. Fungsi Statistik NumPy

Sekarang masuk ke bagian yang sangat relevan dengan data science.

```
nilai = np.array([70, 80, 90, 60, 85])
```

Gunakan:

```
print(np.mean(nilai))
print(np.median(nilai))
print(np.min(nilai))
print(np.max(nilai))
print(np.sum(nilai))
```

Juga:

```
print(np.std(nilai))
```

Mahasiswa diperkenalkan secara sederhana:

- `mean` → rata-rata;
- `median` → nilai tengah;
- `min` → minimum;
- `max` → maksimum;
- `sum` → jumlah;
- `std` → standar deviasi.

Belum perlu membahas rumus statistik secara mendalam. Itu akan menjadi bagian dari mata kuliah berikutnya atau konteks EDA.

X. Perbandingan dengan Algoritma Manual

Ini bagian yang **wajib**.

Sebelumnya mahasiswa membuat:

```
def hitung_rata_rata(data):
    total = 0

    for n in data:
        total += n

    return total / len(data)
```

Sekarang:

```
np.mean(data)
```

Tanyakan:

Apa yang kita dapatkan dari NumPy?

Jawaban:

- kode lebih singkat;
- fungsi sudah tersedia;
- operasi numerik lebih terstruktur;
- dapat bekerja dengan array;
- mendukung operasi numerik skala besar.

Tetapi jangan berhenti pada “kode lebih pendek”.

Tekankan:

Kita baru boleh menggunakan `np.mean()` karena sebelumnya kita sudah memahami algoritma rata-rata.

Y. Agregasi pada Array 2D

Gunakan:

```
nilai = np.array([
    [80, 85, 90],
    [70, 75, 80],
```

```
    [90, 95, 88]
])
```

Rata-rata seluruh data:

```
print(np.mean(nilai))
```

Rata-rata berdasarkan kolom:

```
print(np.mean(nilai, axis=0))
```

Rata-rata berdasarkan baris:

```
print(np.mean(nilai, axis=1))
```

Ini menjadi pengenalan awal konsep:

```
axis=0
↓
operasi sepanjang baris
→ menghasilkan statistik per kolom

axis=1
↓
operasi sepanjang kolom
→ menghasilkan statistik per baris
```

Bagian ini memang perlu diberi waktu karena `axis` sering membingungkan mahasiswa pemula.

Z. Visualisasi Konsep `axis`

Gunakan:

```
            Kolom
        0       1       2

  ┌─────┬───────┬───────┬───────┐
  0     │  80   │  85   │  90   │
  ├─────┼───────┼───────┼───────┤
  1     │  70   │  75   │  80   │
  ├─────┼───────┼───────┼───────┤
  2     │  90   │  95   │  88   │
  └─────┴───────┴───────┴───────┘
```

`axis=0`:

```
↓       ↓       ↓
kolom   kolom   kolom
```

`axis=1`:

→
→
→

Jangan terlalu cepat menggunakan istilah matematika yang abstrak. Gunakan visual dan contoh data.

AA. Boolean Array

Ini bagian yang sangat penting sebagai jembatan ke filtering data.

```
nilai = np.array([45, 60, 75, 90, 55, 80])
```

Coba:

```
print(nilai >= 60)
```

Hasil:

```
[False  True  True  True False  True]
```

Sekarang:

```
print(nilai[nilai >= 60])
```

Hasil:

```
[60 75 90 80]
```

Perhatikan transformasinya:

```
DATA
  ↓
KONDISI
  ↓
BOOLEAN ARRAY
  ↓
FILTER
  ↓
DATA TERPILIH
```

Ini adalah konsep yang akan menjadi **sangat penting dalam Pandas**.

AB. Filtering Data

Contoh:

```
nilai = np.array([45, 60, 75, 90, 55, 80])

lulus = nilai[nilai >= 60]

print(lulus)
```

Kemudian:

```
print(nilai[nilai >= 80])
```

Hasil:

```
[80 90]
```

Mahasiswa mulai melihat bahwa filtering yang pada Modul 3 membutuhkan:

```
for
if
append
```

dapat dilakukan dengan ekspresi array.

AC. Filtering dengan Dua Kondisi

Misalnya kita ingin nilai antara 70 dan 90.

```
hasil = nilai[(nilai >= 70) & (nilai <= 90)]

print(hasil)
```

Perhatikan bahwa pada NumPy kita menggunakan:

```
&
```

bukan:

```
and
```

untuk menggabungkan kondisi element-wise.

Ini perlu dijelaskan dengan hati-hati karena mahasiswa akan mudah keliru.

AD. Challenge — Data Nilai

Gunakan:

```
nilai = np.array([
    55, 72, 80, 90, 65,
    88, 45, 76, 92, 60,
    84, 70
])
```

Cari:

1. nilai lulus;
2. nilai ≥ 80;
3. nilai antara 70 dan 85;
4. jumlah mahasiswa lulus;
5. persentase mahasiswa lulus;
6. rata-rata mahasiswa lulus.

Contoh:

```
lulus = nilai[nilai >= 60]

persentase = len(lulus) / len(nilai) * 100

print(persentase)
```

AE. Operasi Matematika pada Seluruh Data

Misalnya nilai asli:

```
nilai = np.array([60, 70, 80, 90])
```

Kita ingin memberikan bonus 5 poin:

```
nilai_baru = nilai + 5
```

Hasil:

```
[65 75 85 95]
```

Tetapi bagaimana jika nilai maksimum tidak boleh melebihi 100?

Kita dapat menggunakan:

```
nilai_baru = np.minimum(nilai + 5, 100)
```

Ini contoh sederhana bagaimana library membantu menyelesaikan aturan data tanpa membuat loop panjang.

AF. Random Data

Mulai mengenalkan pembuatan data sintetis.

```
np.random.seed(42)

nilai = np.random.randint(50, 101, 20)

print(nilai)
```

Mahasiswa akan memperoleh 20 nilai acak antara 50–100.

Kemudian:

```
print(np.mean(nilai))
print(np.min(nilai))
print(np.max(nilai))
```

Ini menarik karena mahasiswa mulai dapat membuat **dataset simulasi**.

AG. Mengapa Data Sintetis Berguna?

Jelaskan:

Dalam data science, kita tidak selalu memiliki data nyata ketika sedang mengembangkan algoritma.

Kita dapat membuat data simulasi untuk:

- testing;
- eksperimen;
- pembelajaran;
- pengujian algoritma;
- simulasi kondisi tertentu.

Tetapi tekankan:

Data sintetis bukan pengganti data nyata untuk menarik kesimpulan tentang dunia nyata.

AH. Challenge Random Data

Buat 100 nilai ujian:

```
np.random.seed(10)

nilai = np.random.randint(40, 101, 100)
```

Cari:

- rata-rata;
- median;
- minimum;
- maksimum;
- jumlah lulus;
- persentase lulus;
- jumlah nilai ≥ 80.

Kemudian:

Apakah distribusi nilai tersebut terlihat seperti distribusi nilai mahasiswa sungguhan?

Jawaban tidak perlu dicari dengan metode statistik rumit.

Tujuannya adalah mulai membangun **sikap kritis terhadap data**.

AI. Data Science Challenge

Sekarang buat dua kelompok data:

```
kelompok_A = np.random.randint(60, 91, 30)
kelompok_B = np.random.randint(50, 101, 30)
```

Bandingkan:

```
print(np.mean(kelompok_A))
print(np.mean(kelompok_B))
```

Kemudian:

```
print(np.std(kelompok_A))
print(np.std(kelompok_B))
```

Pertanyaan:

Apakah kelompok dengan rata-rata lebih tinggi selalu lebih konsisten?

Ini pertanyaan konseptual yang sangat bagus.

Mahasiswa mulai diperkenalkan bahwa:

Satu angka statistik tidak selalu cukup untuk menggambarkan data.

Ini menjadi pengantar yang sangat baik menuju visualisasi dan EDA.

AJ. Mini Data Challenge

Gunakan dataset nilai:

```
nilai = np.array([
    78, 85, 90, 67, 72,
    88, 95, 60, 76, 83,
    91, 55, 69, 87, 74,
    82, 93, 64, 71, 89
])
```

Mahasiswa membuat analisis:

Statistik dasar

```
Jumlah data
Rata-rata
Median
Minimum
Maksimum
Standar deviasi
```

Filtering

```
Nilai ≥ 80
Nilai < 60
Nilai 70–85
```

Persentase

```
Persentase lulus
Persentase nilai ≥ 80
```

Pertanyaan interpretasi

1. Apakah rata-rata dan median dekat?
2. Apakah terdapat nilai yang sangat rendah?

3. Apakah sebagian besar mahasiswa berada di atas 70?
4. Apakah data terlihat menyebar jauh atau relatif rapat?

Mahasiswa belum perlu membuat grafik. Grafik akan menjadi fokus Modul 6.

AK. Challenge Algoritmik

Berikan tugas:

Buat function menggunakan Python biasa untuk menghitung rata-rata.

Kemudian:

Buat versi menggunakan NumPy.

Versi pertama:

```
def rata_rata_manual(data):
    total = 0

    for x in data:
        total += x

    return total / len(data)
```

Versi kedua:

```
def rata_rata_numpy(data):
    return np.mean(data)
```

Bandingkan:

```
data = np.array([70, 80, 90, 65, 85])

print(rata_rata_manual(data))
print(rata_rata_numpy(data))
```

Tujuan latihan:

Library tidak menggantikan algoritma. Library mengabstraksikan dan mengoptimalkan operasi yang umum digunakan.

AL. Mini Benchmark

Untuk mahasiswa yang lebih cepat, perkenalkan perbandingan sederhana.

```
import time
import numpy as np

data = np.random.randint(1, 100, 1000000)
```

Manual:

```
start = time.time()

total = 0

for x in data:
    total += x

hasil = total / len(data)

end = time.time()

print("Manual:", end - start)
```

NumPy:

```
start = time.time()

hasil = np.mean(data)

end = time.time()

print("NumPy:", end - start)
```

Jelaskan bahwa hasil waktu dapat berbeda-beda tergantung lingkungan komputer.

Tidak perlu menjadikan benchmark sebagai penilaian utama.

Tujuannya hanya memberikan pengalaman:

“Oh, library bukan hanya membuat kode pendek, tetapi dapat memberikan keuntungan komputasi.”

AM. Debugging Challenge

Berikan:

```
import numpy as np

nilai = np.array([70, 80, 90])

print(nilai[3])
```

Tanyakan:

Error apa yang muncul?

Kemudian:

```
print(nilai * [2, 3])
```

Apa yang terjadi?

Diskusikan bahwa operasi array memiliki aturan tertentu dan tidak semua bentuk array dapat dioperasikan secara sembarangan.

Ini menjadi pengantar sangat ringan terhadap konsep **shape compatibility**.

AN. Bonus — Broadcasting

Jika:

```
nilai = np.array([70, 80, 90])
```

kemudian:

```
print(nilai + 5)
```

angka `5` diterapkan ke semua elemen.

Secara konseptual:

```
[70, 80, 90]
+
[ 5,  5,  5]
=
[75, 85, 95]
```

Padahal kita tidak membuat array kedua secara eksplisit.

Ini disebut **broadcasting**.

Untuk semester 1, cukup berikan pemahaman intuitif:

NumPy dapat memperluas nilai tertentu agar kompatibel dengan operasi array.

Tidak perlu masuk ke aturan broadcasting multidimensi yang kompleks.

AO. Function + NumPy

Sekarang gabungkan Modul 4 dan Modul 5.

Buat:

```
def analisis_nilai(data):
    return {
        "jumlah": data.size,
        "rata_rata": np.mean(data),
        "median": np.median(data),
        "minimum": np.min(data),
        "maksimum": np.max(data),
        "standar_deviasi": np.std(data)
    }
```

Kemudian:

```
nilai = np.array([
    78, 85, 90, 67, 72,
    88, 95, 60, 76, 83
])

hasil = analisis_nilai(nilai)

print(hasil)
```

Sekarang mahasiswa telah menggabungkan:

```
Modul 1 → tipe data
Modul 2 → loop & condition
Modul 3 → struktur data
Modul 4 → function
Modul 5 → NumPy
```

Ini penting sebagai checkpoint kurikulum.

AP. Mini Project Modul 5

“Analisis Nilai dengan NumPy”

Gunakan dataset sintetis:

```
np.random.seed(123)

nilai = np.random.randint(40, 101, 100)
```

Buat function:

```
analisis_nilai()
```

yang menghasilkan:

```
Jumlah data
Rata-rata
Median
Minimum
Maksimum
Standar deviasi
Jumlah lulus
Persentase lulus
Jumlah nilai ≥ 80
Persentase nilai ≥ 80
```

Kemudian buat function kedua:

```
kategori_nilai()
```

yang mengelompokkan:

```
≥ 80    → Tinggi
60–79   → Sedang
< 60    → Rendah
```

Belum perlu menggunakan Pandas.

AQ. Tantangan Lebih Sulit

Buat dua kelompok:

```
kelompok_A = np.random.randint(60, 91, 50)
kelompok_B = np.random.randint(50, 101, 50)
```

Program harus menentukan:

```
Kelompok dengan rata-rata lebih tinggi
Kelompok dengan median lebih tinggi
Kelompok dengan variasi lebih besar
```

Kemudian mahasiswa menjawab:

Jika Anda menjadi dosen, apakah hanya berdasarkan rata-rata Anda akan menyimpulkan kelompok mana yang lebih baik?

Pertanyaan ini sengaja mulai menggeser pembelajaran dari:

coding

menuju:

data reasoning.

AR. Penggunaan AI

Pada Modul 5 AI mulai dapat digunakan untuk eksplorasi library.

Contoh prompt yang diperbolehkan:

Jelaskan perbedaan Python list dan NumPy array menggunakan contoh nilai mahasiswa.

Atau:

Saya mendapatkan error ketika menggunakan NumPy array. Jelaskan arti error tersebut tanpa langsung memperbaiki kodenya.

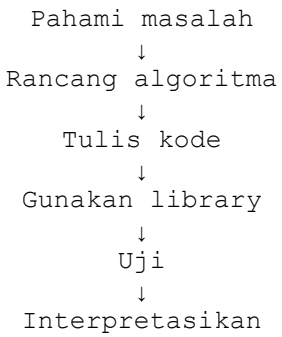
Atau:

Saya ingin menghitung rata-rata, median, dan standar deviasi menggunakan NumPy. Jelaskan fungsi yang tersedia dan kapan masing-masing digunakan.

Untuk mahasiswa yang lebih maju:

Buatkan lima pertanyaan pengujian untuk function analisis NumPy saya. Jangan berikan solusinya.

Prinsipnya tetap:



AS. Tugas Praktikum

“Analisis Data Penjualan”

Buat data sintetis:

```
np.random.seed(42)

harga = np.random.randint(10000, 100000, 100)
jumlah = np.random.randint(1, 10, 100)
```

Hitung:

```
total_penjualan = harga * jumlah
```

Kemudian analisis:

1. total omzet;
2. rata-rata nilai transaksi;
3. transaksi terbesar;
4. transaksi terkecil;
5. median transaksi;
6. standar deviasi transaksi;
7. jumlah transaksi di atas rata-rata;
8. persentase transaksi di atas rata-rata.

Gunakan NumPy.

AT. Tantangan Tambahan

Mahasiswa yang lebih cepat diminta membuat:

```
def analisis_penjualan(harga, jumlah):
    ...
```

Function harus mengembalikan dictionary:

```
{
    "total_omzet": ...,
    "rata_rata": ...,
    "median": ...,
    "minimum": ...,
    "maksimum": ...,
    "standar_deviasi": ...
}
```

Kemudian:

```
hasil = analisis_penjualan(harga, jumlah)
```

Dengan demikian mahasiswa menggabungkan **Function + NumPy**.

AU. Struktur Notebook Google Colab

```
MODUL 5
NUMPY DAN LIBRARY PYTHON

Nama:
NIM:
Kelas:

1. Tujuan Praktikum

2. Review Function

3. Mengenal Library
   3.1 import
   3.2 module
   3.3 function

4. Mengenal NumPy
   4.1 np.array()
   4.2 indexing
   4.3 slicing
   4.4 shape
   4.5 ndim
   4.6 size
   4.7 dtype

5. Operasi Array

6. Array 2 Dimensi

7. Statistik NumPy

8. Boolean Array

9. Filtering

10. Vectorization

11. Random Data

12. Function + NumPy

13. Mini Data Challenge

14. Tugas Analisis Penjualan

15. Interpretasi

16. Refleksi
```

AV. Rubrik Penilaian

Komponen	Bobot
Konsep library	10%
NumPy array	15%
Indexing & slicing	10%
Operasi array	15%
Statistik	15%
Filtering	10%
Function + NumPy	10%
Data challenge	10%
Interpretasi	5%
Total	**100%**

AW. Bacaan dari Dua Buku

Untuk **Python Crash Course, 3rd Edition**, bagian yang relevan untuk modul ini terutama:

Chapter 2 — Variables and Simple Data Types

Untuk memperkuat tipe data dan operasi dasar Python.

Kemudian bagian mengenai **modules** dan penggunaan library pada bagian lanjutan buku dapat digunakan sebagai pengayaan.

Namun perlu ditegaskan: **Python Crash Course bukan buku khusus NumPy.** Jadi jangan memaksakan buku tersebut sebagai sumber utama untuk detail NumPy.

Untuk **Think Python, 3rd Edition**, bagian yang relevan adalah pembahasan mengenai:

- modules;
- import;
- sequences;
- arrays/list;
- numerical operations;
- penggunaan library.

Tetapi sama seperti Python Crash Course, buku ini bukan buku khusus NumPy.

Karena itu, untuk bagian teknis NumPy, modul praktikum ini sebaiknya menggunakan **dokumentasi resmi NumPy sebagai bacaan tambahan**, sedangkan dua buku utama tetap berfungsi sebagai fondasi konsep Python.

Dengan kata lain:

```
Think Python
+
Python Crash Course
↓
Fondasi Python & algoritma
↓
Dokumentasi NumPy
↓
Scientific computing
```

Ini lebih sehat daripada menjadikan dua buku dasar tersebut seolah-olah merupakan manual NumPy.

AX. Exit Ticket

Mahasiswa menjawab sebelum praktikum selesai:

1. **Apa perbedaan list dan NumPy array?**
2. **Apa fungsi `shape`?**
3. **Apa arti `axis=0` pada array dua dimensi?**
4. **Mengapa `nilai[nilai >= 60]` dapat digunakan untuk filtering?**
5. **Apa keuntungan vectorization?**
6. **Mengapa kita belajar algoritma manual sebelum menggunakan NumPy?**

Pertanyaan terakhir merupakan pertanyaan inti modul.

Jawaban yang diharapkan:

Karena library seharusnya digunakan setelah kita memahami masalah dan algoritmanya, sehingga kita tahu apa yang dilakukan library dan dapat memeriksa apakah hasilnya masuk akal.

AY. Jembatan ke Modul 6

Setelah Modul 5 mahasiswa sudah dapat melakukan:

```
DATA
  ↓
NUMPY ARRAY
  ↓
OPERASI NUMERIK
  ↓
STATISTIK
  ↓
FILTERING
```

Tetapi ada masalah:

Kita masih sulit melihat bentuk data hanya dari angka.

Misalnya:

```
[78, 85, 90, 67, 72, 88, 95, 60, 76, 83, ...]
```

- Kita bisa menghitung rata-rata.
- Kita bisa mencari maksimum.

Tetapi kita belum bisa dengan mudah menjawab:

- Apakah sebagian besar data berkumpul di sekitar nilai tertentu?
- Apakah ada nilai yang sangat ekstrem?
- Apakah distribusinya simetris?
- Apakah dua kelompok data memiliki pola yang berbeda?

Maka Modul 6 akan memperkenalkan:

MODUL 6 — VISUALISASI DATA

Dengan alur:

```
NUMPY
  ↓
DATA
  ↓
MATPLOTLIB
  ↓
LINE PLOT
  ↓
BAR CHART
```

```
        ↓
  SCATTER PLOT
        ↓
   HISTOGRAM
        ↓
MEMBACA POLA DATA
```

Dan di sinilah pembelajaran mulai menjadi lebih menyenangkan secara visual.

Mahasiswa tidak hanya melihat:

```
Mean = 74.83
```

tetapi mulai melihat **mengapa angka tersebut demikian** melalui grafik.

Ini kemudian menjadi fondasi langsung menuju **Modul 7 — Pandas & Exploratory Data Analysis**, sebelum akhirnya mahasiswa masuk ke **Data Mining pada Modul 8**.

MODUL 6
Visualisasi Data dengan Python: Mengubah Angka Menjadi Cerita

Mata Kuliah	Praktikum Algoritma dan Pemrograman
Semester	1
Pertemuan	6 dari 10
Durasi	100 menit
Platform	Google Colab
Bahasa	Python 3
Library	NumPy, Pandas, Matplotlib
Level	Pemula → pengantar Data Science

Modul 6 menjadi jembatan penting antara kemampuan pemrograman dasar dan dunia data science.

Pada Modul 5 mahasiswa sudah mengenal NumPy dan konsep library. Sekarang mereka mulai menggunakan Python untuk menjawab pertanyaan yang lebih menarik:

“Apa yang sebenarnya sedang terjadi di dalam data?”

Data yang awalnya hanya berupa angka:

```
70
82
65
91
88
75
```

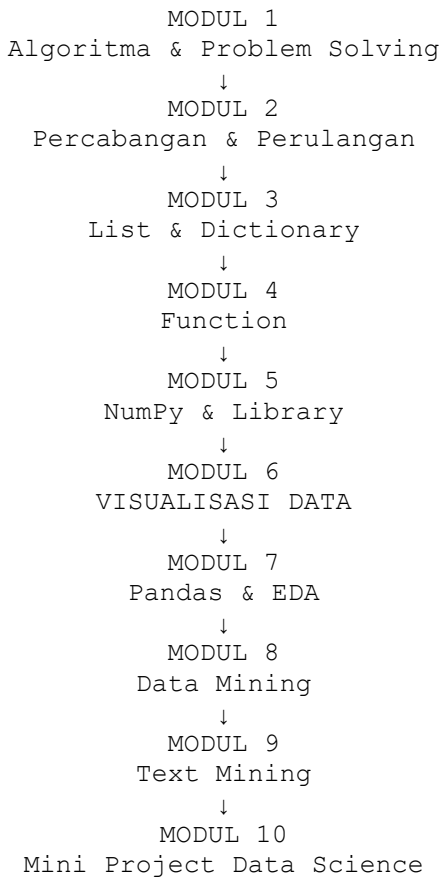
diubah menjadi informasi visual:

```
DATA
 ↓
PENGOLAHAN
 ↓
VISUALISASI
 ↓
POLA
 ↓
INSIGHT
```

Karena itu visualisasi tidak diajarkan sebagai kegiatan “membuat grafik supaya bagus”. Mahasiswa diarahkan memahami bahwa **grafik adalah alat untuk berpikir**.

A. Posisi Modul 6 dalam Roadmap

Urutan 10 modul:

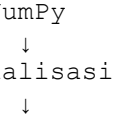


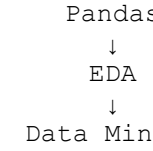
Modul 6 sengaja diletakkan **sebelum Pandas & EDA**.

Alasannya sederhana.

Mahasiswa sebaiknya sudah mampu membaca grafik ketika pada Modul 7 mereka mulai melakukan eksplorasi dataset dengan Pandas.

Jadi:





B. Mengapa Visualisasi Masuk Praktikum Algoritma?

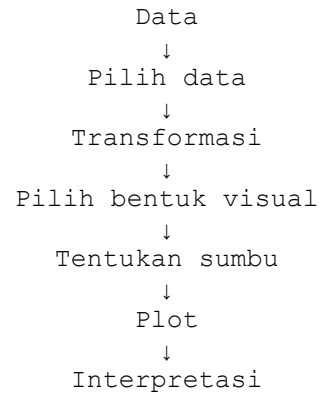
Ini perlu dijelaskan kepada mahasiswa.

Mungkin mereka bertanya:

“Bukankah membuat grafik itu bukan algoritma?”

Jawabannya:

Visualisasi memang bukan inti algoritma, tetapi proses membuat visualisasi tetap melibatkan kemampuan algoritmik:



Bahkan lebih penting lagi:

Mahasiswa belajar memilih representasi yang tepat untuk suatu masalah.

Ini merupakan bagian dari computational thinking.

C. Tujuan Pembelajaran

Setelah menyelesaikan modul ini, mahasiswa mampu:

1. Menjelaskan fungsi visualisasi data.
2. Mengenal Matplotlib.

3. Membuat line chart.
4. Membuat bar chart.
5. Membuat scatter plot.
6. Membuat histogram.
7. Memberi judul dan label grafik.
8. Mengatur ukuran grafik.
9. Membuat beberapa visualisasi dari satu dataset.
10. Memilih jenis grafik berdasarkan jenis pertanyaan.
11. Membaca pola dari visualisasi.
12. Menghindari visualisasi yang menyesatkan.
13. Menggabungkan NumPy dengan Matplotlib.
14. Membuat visualisasi sederhana menggunakan Pandas.
15. Menjelaskan insight berdasarkan grafik.
16. Menggunakan visualisasi sebagai bagian dari proses analisis data.

D. Skenario Praktikum 100 Menit

Waktu	Kegiatan
0–10 menit	Pengantar visualisasi
10–20 menit	Matplotlib & konsep figure
20–35 menit	Line chart
35–48 menit	Bar chart
48–60 menit	Scatter plot
60–70 menit	Histogram
70–80 menit	Visualisasi dengan Pandas
80–90 menit	Data Challenge
90–97 menit	Mini Project
97–100 menit	Exit Ticket

E. Bagian 1 — Data vs Informasi

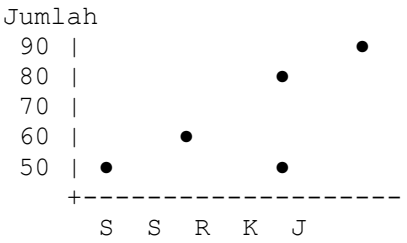
Mulai dengan dua bentuk penyajian.

Bentuk 1

```
Senin     50
Selasa    60
Rabu      55
Kamis     80
Jumat     90
```

Bentuk 2

Grafik:



Tanyakan:

Mana yang lebih cepat menunjukkan bahwa terjadi kenaikan pada Kamis dan Jumat?

Mahasiswa biasanya akan menjawab grafik.

Kemudian jelaskan:

Visualisasi membantu manusia menemukan pola yang sulit terlihat dari tabel angka.

F. Visualisasi Bukan Sekadar Hiasan

Tekankan:

```
Grafik yang indah
≠
Grafik yang informatif
```

Grafik harus menjawab pertanyaan.

Misalnya:

Bagaimana perubahan penjualan dari waktu ke waktu?

Gunakan:

```
LINE CHART
```

Pertanyaan:

Produk mana yang paling banyak terjual?

Gunakan:

```
BAR CHART
```

Pertanyaan:

Apakah ada hubungan antara jam belajar dan nilai?

Gunakan:

```
SCATTER PLOT
```

Pertanyaan:

Bagaimana distribusi nilai?

Gunakan:

```
HISTOGRAM
```

Inilah konsep utama Modul 6.

G. Bagian 2 — Mengenal Matplotlib

Import:

```
import matplotlib.pyplot as plt
```

Jelaskan:

```
matplotlib
    ↓
library visualisasi

pyplot
    ↓
modul yang sering digunakan

plt
 ↓
alias
```

Mahasiswa tidak perlu menghafal sejarah Matplotlib.

Yang penting:

```
import matplotlib.pyplot as plt
```

kemudian:

```
plt.plot(...)
plt.bar(...)
```

```
plt.scatter(...)
plt.hist(...)
```

H. Grafik Pertama

```
import matplotlib.pyplot as plt

x = [1, 2, 3, 4, 5]
y = [10, 20, 15, 30, 40]

plt.plot(x, y)

plt.show()
```

Mahasiswa melihat hubungan:

```
          x
          ↓
sumbu horizontal

          y
          ↓
 sumbu vertikal
```

I. Menambahkan Judul

```
plt.plot(x, y)

plt.title("Perubahan Nilai")

plt.show()
```

Tetapi grafik masih belum lengkap.

Tambahkan:

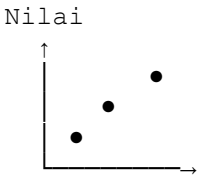
```
plt.xlabel("Hari")
plt.ylabel("Nilai")
```

Kode lengkap:

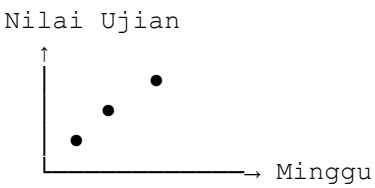
```
plt.plot(x, y)

plt.title("Perubahan Nilai")
plt.xlabel("Hari")
plt.ylabel("Nilai")

plt.show()
```

J. Mengapa Label Penting?

Bandingkan:



dengan:



Grafik kedua jauh lebih informatif.

Jelaskan kepada mahasiswa:

Grafik tanpa konteks dapat membuat pembaca menebak-nebak.

K. Line Chart

Line chart cocok untuk data yang memiliki urutan, terutama waktu.

Contoh:

```
hari = [
    "Senin",
    "Selasa",
    "Rabu",
    "Kamis",
    "Jumat"
]

penjualan = [
    50,
    60,
    55,
    80,
    90
]

plt.plot(hari, penjualan)
```

```
plt.title("Penjualan Mingguan")
plt.xlabel("Hari")
plt.ylabel("Jumlah Penjualan")

plt.show()
```

Pertanyaan:

Pada hari apa penjualan tertinggi?

Mahasiswa membaca grafik.

L. Marker

Tambahkan titik:

```
plt.plot(
    hari,
    penjualan,
    marker="o"
)
```

Sekarang setiap observasi terlihat jelas.

Konsep:

```
line
+
marker
=
lebih mudah membaca perubahan
```

M. Challenge Line Chart

Berikan data:

```
bulan = [
    "Jan",
    "Feb",
    "Mar",
    "Apr",
    "Mei",
    "Jun"
]

pengunjung = [
    120,
    150,
```

```
    140,
    180,
    210,
    200
]
```

Tugas:

Buat grafik perkembangan jumlah pengunjung.

Kemudian mahasiswa menjawab:

1. Bulan tertinggi?
2. Bulan terendah?
3. Apakah tren secara umum meningkat?
4. Apakah setiap bulan selalu meningkat?

Pertanyaan keempat penting.

Karena:

```
tren meningkat
≠
setiap titik meningkat
```

N. Bagian 3 — Bar Chart

Sekarang pertanyaan berubah:

Kategori mana yang paling tinggi?

Gunakan:

```
kategori = [
    "Python",
    "Java",
    "R",
    "C++"
]

jumlah = [
    80,
    65,
    40,
    30
]

plt.bar(
    kategori,
```

```
    jumlah
)

plt.title("Jumlah Mahasiswa")
plt.xlabel("Bahasa")
plt.ylabel("Jumlah")

plt.show()
```

Bar chart cocok untuk membandingkan kategori.

O. Bar Chart Horizontal

Jika nama kategorinya panjang:

```
plt.barh(
    kategori,
    jumlah
)

plt.title("Jumlah Mahasiswa")
plt.xlabel("Jumlah")
plt.ylabel("Bahasa")

plt.show()
```

Mahasiswa diperkenalkan pada prinsip:

Bentuk grafik dapat dipilih berdasarkan kebutuhan membaca data.

P. Challenge Bar Chart

Dataset:

```
mata_kuliah = [
    "Algoritma",
    "Basis Data",
    "Matematika",
    "Jaringan",
    "Pemrograman Web"
]

nilai = [
    78,
    82,
    75,
    80,
    85
]
```

Tugas:

Buat bar chart dan tentukan mata kuliah dengan nilai rata-rata tertinggi.

Kemudian:

Apakah grafik tersebut membuktikan bahwa mata kuliah tersebut paling mudah?

Jawabannya:

Tidak.

Grafik hanya menunjukkan perbedaan nilai pada dataset.

Q. Bagian 4 — Scatter Plot

Sekarang masuk ke grafik yang sangat penting untuk Data Science.

Pertanyaan:

Apakah dua variabel memiliki pola hubungan?

Contoh:

```
jam_belajar = [
    1, 2, 3, 4, 5,
    6, 7, 8, 9
]

nilai = [
    55, 58, 60, 65, 68,
    72, 75, 82, 88
]
```

Buat:

```
plt.scatter(
    jam_belajar,
    nilai
)

plt.xlabel("Jam Belajar")
plt.ylabel("Nilai")
plt.title("Jam Belajar vs Nilai")

plt.show()
```

Mahasiswa akan melihat pola titik.

R. Mengajarkan Korelasi Secara Visual

Tanyakan:

Apakah titik-titik tersebut cenderung bergerak naik?

Jika ya:

```
hubungan positif
```

Jika turun:

```
hubungan negatif
```

Jika tidak membentuk pola:

```
hubungan lemah/tidak jelas
```

Namun jangan menyatakan:

“Kalau grafik naik berarti jam belajar menyebabkan nilai naik.”

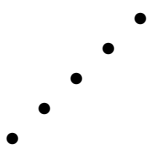
Belum tentu.

Tanamkan sejak semester 1:

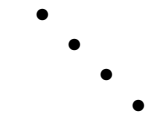
Korelasi bukan otomatis kausalitas.

S. Tiga Pola Scatter Plot

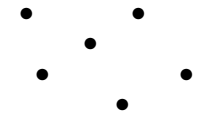
Positif



Negatif



Tidak jelas



Mahasiswa diminta mengidentifikasi ketiganya.

T. Challenge Scatter Plot

Buat:

```
np.random.seed(42)

jam_belajar = np.random.randint(
    1, 10, 50
)

nilai = (
    jam_belajar * 5
    + np.random.randint(-10, 11, 50)
)
```

Kemudian:

```
plt.scatter(
    jam_belajar,
    nilai
)

plt.xlabel("Jam Belajar")
plt.ylabel("Nilai")

plt.title(
    "Hubungan Jam Belajar dan Nilai"
)

plt.show()
```

Tanyakan:

Mengapa titik-titik tidak membentuk garis lurus sempurna?

Karena data memiliki variasi/noise.

Ini menjadi pengantar yang sangat baik menuju data mining.

U. Bagian 5 — Histogram

Sekarang pertanyaannya:

Bagaimana penyebaran nilai?

Gunakan:

```
nilai = [
    55, 60, 62, 65, 65,
    68, 70, 70, 72, 75,
    78, 80, 82, 85, 90
]

plt.hist(nilai)

plt.title("Distribusi Nilai")
plt.xlabel("Nilai")
plt.ylabel("Frekuensi")

plt.show()
```

Mahasiswa mulai mengenal konsep:

```
distribution
```

V. Apa yang Dilihat dari Histogram?

Mahasiswa diminta memperhatikan:

```
Apakah data terkonsentrasi?

Apakah menyebar?

Apakah ada nilai ekstrem?

Di rentang mana data paling banyak?
```

Ini merupakan persiapan langsung menuju EDA pada Modul 7.

W. Mengatur Jumlah Bin

```
plt.hist(
    nilai,
    bins=5
)
```

Bandingkan:

```
plt.hist(
    nilai,
    bins=10
)
```

Tanyakan:

Mengapa bentuk grafik berubah?

Karena `bins` menentukan bagaimana rentang nilai dibagi ke dalam kelompok.

Ini juga memperkenalkan gagasan bahwa:

Cara kita menyajikan data dapat memengaruhi apa yang terlihat.

X. Data Visualization dan Persepsi

Ini bagian penting untuk mahasiswa.

Misalnya data:

```
70
71
72
73
74
```

Jika sumbu Y dimulai dari 69, perubahan terlihat kecil.

Jika sumbu Y dibuat sangat sempit, perubahan dapat terlihat jauh lebih dramatis.

Maka:

Grafik dapat menyesatkan jika skala atau konteksnya tidak tepat.

Untuk mahasiswa semester 1, cukup diberikan contoh sederhana dan tidak perlu masuk ke teori visualisasi yang kompleks.

Y. Bagian 6 — Menggunakan NumPy

Hubungkan dengan Modul 5.

```
import numpy as np
```

Buat data:

```
x = np.arange(1, 11)

y = x ** 2
```

Kemudian:

```
plt.plot(x, y)

plt.title("Fungsi Kuadrat")
plt.xlabel("x")
plt.ylabel("y")

plt.show()
```

Ini memperlihatkan:

```
NumPy
↓
membuat/mengolah array
↓
Matplotlib
↓
memvisualisasikan array
```

Z. Visualisasi Fungsi Matematika

Ini bagus untuk menguatkan hubungan matematika–algoritma–Python.

```
x = np.linspace(
    0,
    10,
    100
)

y = np.sin(x)

plt.plot(x, y)

plt.title("Fungsi Sinus")
plt.xlabel("x")
plt.ylabel("sin(x)")

plt.show()
```

Mahasiswa tidak perlu belajar teori sinus.

Fokusnya:

```
array
```

```
      ↓
  operasi
      ↓
visualisasi
```

AA. Challenge: Bandingkan Dua Fungsi

```
x = np.linspace(
    0,
    10,
    100
)

y1 = x
y2 = x ** 2
```

Plot:

```
plt.plot(
    x,
    y1,
    label="y = x"
)

plt.plot(
    x,
    y2,
    label="y = x²"
)

plt.xlabel("x")
plt.ylabel("y")
plt.title("Perbandingan Fungsi")

plt.legend()

plt.show()
```

Mahasiswa diperkenalkan pada:

```
label
legend
```

AB. Bagian 7 — Multiple Lines

Contoh:

```
bulan = [
```

```
    "Jan",
    "Feb",
    "Mar",
    "Apr",
    "Mei"
]

produk_A = [
    100,
    120,
    130,
    150,
    160
]

produk_B = [
    90,
    110,
    125,
    140,
    180
]
```

Buat:

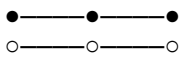
```
plt.plot(
    bulan,
    produk_A,
    marker="o",
    label="Produk A"
)

plt.plot(
    bulan,
    produk_B,
    marker="o",
    label="Produk B"
)

plt.xlabel("Bulan")
plt.ylabel("Penjualan")

plt.title(
    "Perbandingan Penjualan"
)

plt.legend()

plt.show()
```

Pertanyaan:

Produk mana yang mengalami pertumbuhan lebih besar?

AC. Mengapa Legend Penting?

Tanpa legend:



Kita tidak tahu garis mana milik siapa.

Dengan:

```
Produk A
Produk B
```

grafik dapat dibaca.

Prinsip:

Setiap elemen visual harus memiliki fungsi komunikasi.

AD. Bagian 8 — Subplot

Mahasiswa semester 1 boleh diperkenalkan dengan beberapa grafik dalam satu figure.

```
fig, axes = plt.subplots(
    2,
    2,
    figsize=(10, 8)
)
```

Kemudian:

```
axes[0, 0].plot(
    bulan,
    produk_A
)

axes[0, 0].set_title(
    "Produk A"
)

axes[0, 1].plot(
    bulan,
    produk_B
)

axes[0, 1].set_title(
```

```
    "Produk B"
)
```

Untuk dua grafik lainnya:

```
axes[1, 0].bar(
    kategori,
    jumlah
)

axes[1, 0].set_title(
    "Kategori"
)

axes[1, 1].hist(
    nilai,
    bins=5
)

axes[1, 1].set_title(
    "Distribusi"
)
```

Kemudian:

```
plt.tight_layout()
plt.show()
```

Tidak perlu menjadikan `subplot` materi wajib untuk semua mahasiswa. Mahasiswa yang lebih cepat dapat menggunakannya.

AE. Bagian 9 — Visualisasi dengan Pandas

Ini menjadi jembatan menuju Modul 7.

Buat DataFrame:

```
import pandas as pd

df = pd.DataFrame({
    "Bulan": [
        "Jan",
        "Feb",
        "Mar",
        "Apr",
        "Mei"
    ],

    "Penjualan": [
```

```
        100,
        120,
        130,
        150,
        180
    ]
})
```

Kemudian:

```
df.plot(
    x="Bulan",
    y="Penjualan",
    kind="line"
)

plt.title(
    "Penjualan Bulanan"
)

plt.show()
```

Mahasiswa mulai melihat bahwa Pandas menyediakan interface visualisasi sederhana.

AF. Bar Chart dari DataFrame

```
df.plot(
    x="Bulan",
    y="Penjualan",
    kind="bar"
)

plt.title(
    "Penjualan Bulanan"
)

plt.show()
```

Sekarang:

```
Pandas
  ↓
DataFrame
  ↓
plot()
  ↓
Matplotlib
```

Ini akan sangat berguna pada Modul 7.

AG. DataFrame Nilai Mahasiswa

Gunakan dataset:

```
df = pd.DataFrame({
    "Nama": [
        "Andi",
        "Budi",
        "Citra",
        "Dina",
        "Eko"
    ],

    "UTS": [
        80,
        65,
        90,
        75,
        85
    ],

    "UAS": [
        85,
        70,
        92,
        80,
        88
    ]
})
```

Hitung nilai akhir:

```
df["Nilai_Akhir"] = (
    df["UTS"] * 0.4 +
    df["UAS"] * 0.6
)
```

Kemudian:

```
df
```

Visualisasikan:

```
df.plot(
    x="Nama",
    y="Nilai_Akhir",
    kind="bar"
)

plt.title(
    "Nilai Akhir Mahasiswa"
)
```

```
plt.ylabel("Nilai")

plt.show()
```

AH. Challenge: Cari Mahasiswa Tertinggi

Gunakan:

```
tertinggi = df.loc[
    df["Nilai_Akhir"].idxmax()
]

print(tertinggi)
```

Hubungkan dengan algoritma:

```
DATA
                                  ↓
                               SEARCH
                                  ↓
                              MAXIMUM
                                  ↓
                           VISUALIZATION
```

Ini penting.

Mahasiswa tidak hanya belajar membuat grafik.

Mereka belajar menggunakan program untuk menjawab pertanyaan.

AI. Challenge: Buat Grafik yang Tepat

Berikan empat pertanyaan:

Pertanyaan 1

Bagaimana perubahan penjualan dari Januari sampai Desember?

Jawaban:

```
Line chart
```

Pertanyaan 2

Produk mana yang paling banyak terjual?

Jawaban:

```
Bar chart
```

Pertanyaan 3

Apakah harga berkaitan dengan jumlah penjualan?

Jawaban:

```
Scatter plot
```

Pertanyaan 4

Bagaimana distribusi nilai mahasiswa?

Jawaban:

```
Histogram
```

Ini harus menjadi latihan penting.

Mahasiswa belajar **memilih alat berdasarkan masalah**.

AJ. Challenge Logika Visualisasi

Berikan data:

```
kategori = [
    "A",
    "B",
    "C",
    "D"
]

nilai = [
    80,
    81,
    82,
    83
]
```

Tanyakan:

Jika dibuat bar chart, bagaimana grafik tersebut terlihat?

Kemudian ubah:

```
80 → 0
81 → 1
82 → 2
83 → 3
```

Pertanyaan:

Apakah persepsi visual berubah?

Ya.

Lalu diskusikan:

Apakah grafik tersebut berbohong?

Tidak selalu, tetapi cara memilih skala dapat membuat perbedaan terlihat lebih atau kurang dramatis.

Ini menjadi pengantar **visual literacy**.

AK. Challenge Algoritma Tanpa Matplotlib

Untuk menjaga karakter mata kuliah, berikan tugas:

Diberikan list nilai, buat representasi grafik batang sederhana menggunakan karakter `*`.

Contoh:

```
Nilai:
[3, 5, 2, 7]
```

Output:

```
A | ***
B | *****
C | **
D | *******
```

Function:

```
def grafik_batang(data):
    ...
```

Mahasiswa harus memikirkan:

```
list
  ↓
loop
  ↓
repetisi string
  ↓
output
```

Ini sangat bagus untuk melatih logika sebelum mereka bergantung pada Matplotlib.

AL. Challenge Lebih Sulit — Grafik Horizontal

Buat function:

```
def grafik_batang(data, labels):
    ...
```

Contoh:

```
data = [5, 8, 3, 6]

labels = [
    "Python",
    "Java",
    "R",
    "C++"
]
```

Output:

```
Python | *****
Java   | ********
R      | ***
C++    | ******
```

Mahasiswa yang mampu menyelesaikan ini telah menggabungkan:

```
list
loop
function
string
```

yang semuanya berasal dari modul-modul awal.

AM. Mini Project Modul 6

“Menceritakan Data melalui Grafik”

Mahasiswa diberikan dataset:

```
df = pd.DataFrame({
    "Bulan": [
        "Jan", "Feb", "Mar",
        "Apr", "Mei", "Jun",
        "Jul", "Agu", "Sep",
        "Okt", "Nov", "Des"
    ],

    "Penjualan": [
        100, 120, 115,
        140, 160, 170,
        165, 180, 190,
        210, 220, 240
    ],

    "Pengunjung": [
        500, 550, 520,
        600, 650, 700,
        680, 750, 800,
        850, 900, 950
    ]
})
```

Mahasiswa harus membuat minimal:

Grafik 1 — Line Chart

```
Penjualan per bulan
```

Grafik 2 — Bar Chart

```
Penjualan per bulan
```

Grafik 3 — Scatter Plot

```
Pengunjung vs Penjualan
```

Kemudian menjawab:

1. Bulan dengan penjualan tertinggi?
2. Bulan dengan penjualan terendah?

3. Apakah penjualan cenderung meningkat?
4. Apakah jumlah pengunjung tampak berhubungan dengan penjualan?
5. Grafik mana yang paling tepat untuk menjawab masing-masing pertanyaan?
6. Apakah visualisasi cukup untuk membuktikan bahwa pengunjung menyebabkan penjualan meningkat?

AN. Tugas Interpretasi

Ini wajib.

Mahasiswa tidak boleh hanya mengumpulkan:

```
plt.plot(...)
```

Mereka harus menulis:

Temuan: Penjualan menunjukkan kecenderungan meningkat dari awal hingga akhir tahun.

Bukti visual: Grafik line menunjukkan arah umum peningkatan meskipun terdapat beberapa bulan yang mengalami penurunan.

Catatan: Grafik menunjukkan pola pada dataset, tetapi tidak membuktikan hubungan sebab-akibat.

Ini mulai membentuk gaya berpikir data scientist.

AO. Kesalahan Umum

1. Grafik tanpa judul

Pembaca tidak tahu apa yang dilihat.

2. Sumbu tanpa label

Makna data menjadi tidak jelas.

3. Memilih grafik berdasarkan selera

Misalnya:

“Saya suka pie chart.”

Bukan alasan ilmiah.

4. Terlalu banyak elemen

Grafik menjadi sulit dibaca.

5. Grafik tanpa interpretasi

Mahasiswa harus menjelaskan apa yang terlihat.

6. Menganggap grafik membuktikan sebab-akibat

Tidak.

7. Menggunakan grafik yang salah

Misalnya menggunakan line chart untuk membandingkan kategori yang tidak memiliki urutan.

AP. Bagian Pengayaan — Pie Chart

Pie chart boleh diperkenalkan, tetapi **bukan fokus utama**.

Contoh:

```
kategori = [
    "Python",
    "Java",
    "R",
    "C++"
]

jumlah = [
    40,
    30,
    20,
    10
]

plt.pie(
    jumlah,
    labels=kategori,
    autopct="%1.1f%%"
)

plt.title(
    "Preferensi Bahasa Pemrograman"
)
```

```
plt.show()
```

Namun mahasiswa harus memahami:

Pie chart cocok ketika ingin menunjukkan bagian dari keseluruhan dan jumlah kategorinya tidak terlalu banyak.

Jangan menjadikan pie chart sebagai default untuk semua data kategorikal.

AQ. Bagian Pengayaan — Boxplot

Mahasiswa yang lebih cepat dapat diperkenalkan:

```
plt.boxplot(nilai)

plt.title(
    "Distribusi Nilai"
)

plt.ylabel("Nilai")

plt.show()
```

Cukup jelaskan bahwa boxplot membantu melihat:

- median;
- penyebaran;
- kemungkinan outlier.

Tidak perlu masuk ke perhitungan kuartil secara mendalam.

Materi ini akan berguna pada Modul 7 ketika mahasiswa mulai melakukan EDA.

AR. Bagian Pengayaan — Heatmap Korelasi

Karena kita ingin mengarahkan mahasiswa menuju data science, boleh diperlihatkan:

```
korelasi = df[
    [
        "Penjualan",
        "Pengunjung"
    ]
].corr()
```

```
plt.imshow(
    korelasi
)

plt.xticks(
    range(len(korelasi.columns)),
    korelasi.columns
)

plt.yticks(
    range(len(korelasi.columns)),
    korelasi.columns
)

plt.colorbar()

plt.title(
    "Korelasi Variabel"
)

plt.show()
```

Tetapi jangan menjadikan heatmap sebagai kompetensi wajib.

AS. Struktur Notebook Google Colab

Notebook mahasiswa:

```
MODUL 6
VISUALISASI DATA

Nama:
NIM:
Kelas:

1. Tujuan Praktikum

2. Mengapa Visualisasi?

3. Import Library

4. Line Chart

5. Bar Chart

6. Scatter Plot

7. Histogram

8. Visualisasi dengan NumPy

9. Visualisasi dengan Pandas
```

```
10. Challenge Pemilihan Grafik

11. Challenge Algoritma

12. Mini Project

13. Interpretasi

14. Kesalahan Visualisasi

15. Refleksi
```

AT. Rubrik Penilaian

Komponen	Bobot
Konsep visualisasi	10%
Line chart	10%
Bar chart	10%
Scatter plot	10%
Histogram	10%
NumPy + Matplotlib	10%
Pandas + visualisasi	10%
Pemilihan jenis grafik	10%
Interpretasi	10%
Challenge algoritmik	10%
Total	**100%**

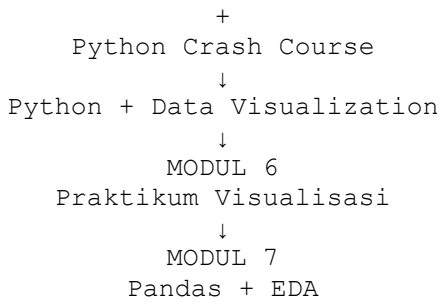
AU. Hubungan dengan Dua Buku Pedoman

Modul 6 sangat erat dengan bagian proyek data visualization dalam **Python Crash Course, 3rd Edition**. Buku tersebut membawa mahasiswa dari fondasi Python menuju penggunaan data dan visualisasi, termasuk penggunaan Matplotlib dan eksplorasi data.

Sementara itu, **Think Python, 3rd Edition** tetap menjadi fondasi kemampuan programming yang diperlukan mahasiswa untuk memahami kode visualisasi: variable, function, sequence, iteration, dan data structures. Buku tersebut tidak menjadi buku khusus visualisasi, tetapi konsep fundamentalnya menjadi prasyarat.

Jadi posisi bacaan mahasiswa:

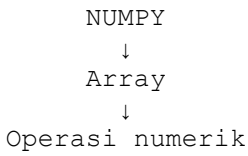
```
Think Python
      ↓
Fundamental Python
```



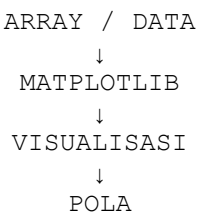
AV. Hubungan Modul 5 → 6 → 7

Ini penting untuk buku modul secara keseluruhan.

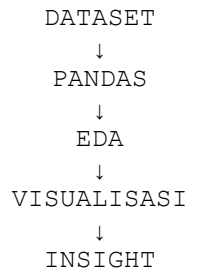
Modul 5



Modul 6

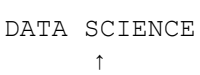


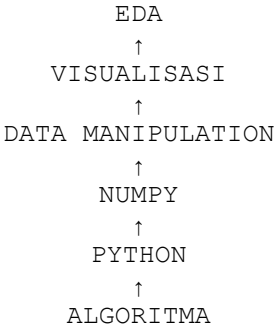
Modul 7



Jadi mahasiswa tidak belajar tiga topik yang terpisah.

Mereka sedang menaiki tangga:





AW. Tantangan Utama Modul 6

Saya menyarankan satu pertanyaan menjadi **signature challenge** modul ini:

“Jika saya hanya boleh membuat satu grafik untuk menjelaskan dataset ini kepada orang lain, grafik apa yang saya pilih dan mengapa?”

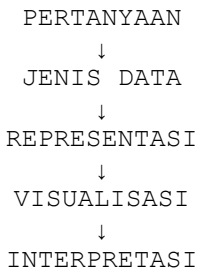
Mahasiswa harus memilih sendiri.

Tidak ada jawaban otomatis.

Karena inti visualisasi bukan kemampuan mengetik:

```
plt.plot()
```

melainkan kemampuan berpikir:



Itulah kompetensi yang nantinya sangat berguna ketika mereka masuk ke **Pandas & EDA pada Modul 7**, kemudian **Data Mining pada Modul 8**, dan akhirnya **Mini Project Data Science pada Modul 10**.

MODUL 7

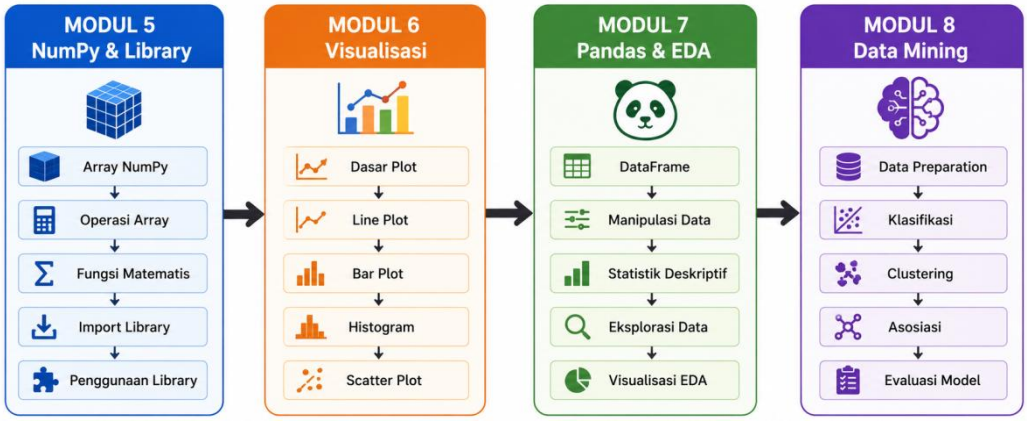
Pandas dan Exploratory Data Analysis (EDA)

Mata Kuliah	Praktikum Algoritma dan Pemrograman
Semester	1
Pertemuan	7 dari 10
Durasi	100 menit
Platform	Google Colab
Bahasa	Python 3
Library utama	Pandas, NumPy, Matplotlib
Level	Pemula → pengantar data analysis

Modul 7 adalah titik ketika mahasiswa mulai benar-benar bekerja seperti seorang pemula di bidang data.

Pada Modul 1–4 mereka belajar membangun algoritma. Modul 5 memperkenalkan NumPy sebagai alat komputasi numerik. Modul 6 memperkenalkan visualisasi. Sekarang keduanya digabungkan dengan **Pandas** untuk mengolah data berbentuk tabel dan melakukan **Exploratory Data Analysis (EDA)**.

Roadmap yang kita gunakan memang menempatkan:



Ini juga sesuai dengan materi pendukung yang Anda miliki: Pandas dijelaskan sebagai library untuk manipulasi dan analisis data tabular melalui `Series` dan `DataFrame`, termasuk filtering, pemilihan, restrukturisasi, pembersihan, dan analisis data.

A. Gagasan Besar Modul

Pada Modul 3 mahasiswa memiliki data seperti:

```
mahasiswa = [
    {"nama": "Andi", "nilai": 80},
    {"nama": "Budi", "nilai": 65},
    {"nama": "Citra", "nilai": 90}
]
```

Kemudian pada Modul 4 data tersebut diproses menggunakan function.

Pada Modul 5 mahasiswa mulai menggunakan NumPy.

Sekarang kita menghadapi data yang lebih realistis:

```
Nama    Prodi       UTS   UAS   IPK
Andi    Informatika 80    85    3.70
Budi    Informatika 70    65    3.20
Citra   Sistem Info 90    95    3.90
...
```

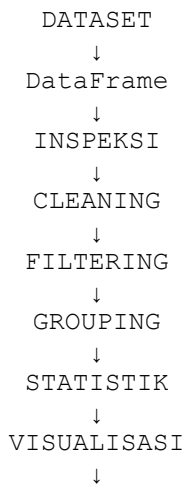
Jika data hanya 5 baris, kita masih dapat mengolahnya secara manual.

Tetapi bagaimana jika:

```
10.000 mahasiswa
500.000 transaksi
1.000.000 data pelanggan
```

Di sinilah Pandas digunakan.

Konsep utama:



Jadi EDA bukan sekadar:

“Memakai Pandas untuk menghasilkan tabel.”

EDA adalah proses **mengenali, memeriksa, membersihkan, mengeksplorasi, dan memahami pola awal suatu dataset sebelum melakukan analisis atau pemodelan lebih lanjut.**

B. Capaian Pembelajaran

Setelah menyelesaikan modul ini, mahasiswa mampu:

1. Menjelaskan fungsi Pandas.
2. Menjelaskan perbedaan `Series` dan `DataFrame`.
3. Membuat DataFrame dari dictionary dan list.
4. Membaca dataset CSV.
5. Menampilkan beberapa baris pertama dan terakhir.
6. Memeriksa struktur dataset.
7. Memilih kolom.
8. Memilih baris berdasarkan kondisi.
9. Melakukan filtering.
10. Mengurutkan data.
11. Menangani data kosong secara sederhana.
12. Menggunakan statistik deskriptif.
13. Mengelompokkan data menggunakan `groupby()`.
14. Menghitung jumlah data berdasarkan kategori.
15. Menggabungkan Pandas dengan NumPy.
16. Menggabungkan Pandas dengan Matplotlib.
17. Melakukan EDA sederhana.
18. Menuliskan insight berdasarkan hasil eksplorasi.
19. Membedakan **hasil komputasi** dengan **interpretasi data**.
20. Menyiapkan dataset sederhana untuk tahap Data Mining pada Modul 8.

C. Mengapa Pandas Setelah NumPy?

Mahasiswa mungkin bertanya:

“Bukankah NumPy sudah bisa mengolah data?”

Bisa.

Tetapi NumPy terutama sangat kuat untuk **komputasi numerik berbasis array**.

Pandas dirancang lebih nyaman untuk data berbentuk tabel.

Secara sederhana:

```
NumPy
  ↓
Array numerik
  ↓
Komputasi
```

sedangkan:

```
Pandas
  ↓
DataFrame
  ↓
Data tabular
  ↓
Manipulasi + Analisis
```

Sumber pendukung yang Anda miliki juga menempatkan NumPy sebagai fondasi library penting seperti Pandas, Scikit-learn, dan Matplotlib, sedangkan Pandas digunakan untuk data tabular dan analisis.

D. Skenario Praktikum 100 Menit

Waktu	Kegiatan
0–10 menit	Review NumPy & visualisasi
10–25 menit	Pandas, Series, DataFrame
25–40 menit	Membaca dan memeriksa dataset
40–55 menit	Selecting & filtering
55–68 menit	Cleaning sederhana
68–80 menit	Statistik & `groupby()`
80–90 menit	EDA + visualisasi
90–97 menit	Data Challenge
97–100 menit	Exit ticket

E. Bagian 1 — Mengenal Pandas

Mulai Google Colab:

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

Pandas biasanya disingkat:

`pd`

sehingga:

```
pd.DataFrame()
```

Pandas merupakan library open-source untuk manipulasi dan analisis data, terutama data tabular. Struktur utamanya adalah `Series` untuk data satu dimensi dan `DataFrame` untuk data dua dimensi.

F. Series

Buat:

```
nilai = pd.Series([80, 75, 90, 65, 85])

print(nilai)
```

Hasilnya kira-kira:

```
0    80
1    75
2    90
3    65
4    85
dtype: int64
```

Bandingkan dengan NumPy:

```
nilai_np = np.array([80, 75, 90, 65, 85])
```

Secara sederhana:

```
NumPy array
→ kumpulan nilai

Pandas Series
→ kumpulan nilai + label/index
```

G. Series dengan Index

Kita dapat memberikan label:

```
nilai = pd.Series(
    [80, 75, 90],
    index=["Andi", "Budi", "Citra"]
)
```

```
print(nilai)
```

Hasil:

```
Andi     80
Budi     75
Citra    90
```

Sekarang:

```
print(nilai["Citra"])
```

hasil:

```
90
```

Mahasiswa mulai melihat bahwa Pandas memberikan **label pada data**.

H. DataFrame

Sekarang masuk ke struktur utama Pandas.

```
data = {
    "Nama": ["Andi", "Budi", "Citra", "Dina"],
    "Nilai": [80, 65, 90, 75],
    "Prodi": [
        "Informatika",
        "Informatika",
        "Sistem Informasi",
        "Informatika"
    ]
}

df = pd.DataFrame(data)

df
```

Google Colab akan menampilkan tabel.

Visual:

```
      Nama              Nilai       Prodi
0     Andi                80        Informatika
1     Budi                65        Informatika
2     Citra               90        Sistem Informasi
3     Dina                75        Informatika
```

Inilah **DataFrame**.

I. Analogi dengan yang Sudah Dipelajari

Mahasiswa sudah belajar:

```
mahasiswa = [
    {"nama": "Andi", "nilai": 80},
    {"nama": "Budi", "nilai": 65},
    {"nama": "Citra", "nilai": 90}
]
```

Pandas dapat mengubah struktur tersebut menjadi:

```
df = pd.DataFrame(mahasiswa)
```

Sehingga:

```
LIST
 +
DICTIONARY
 ↓
PANDAS DATAFRAME
```

Ini merupakan transisi yang sangat natural dari Modul 3.

J. Melihat Dataset

Gunakan:

```
df.head()
```

Untuk lima baris pertama.

Jika dataset besar:

```
df.head(10)
```

Untuk bagian akhir:

```
df.tail()
```

Untuk mengetahui ukuran:

```
df.shape
```

Misalnya:

```
(100, 5)
```

berarti:

```
100 baris
5 kolom
```

K. Memeriksa Struktur Dataset

Gunakan:

```
df.info()
```

Mahasiswa akan melihat:

- jumlah baris;
- nama kolom;
- tipe data;
- jumlah nilai non-null.

Kemudian:

```
df.dtypes
```

Untuk melihat tipe data masing-masing kolom.

Ini merupakan salah satu kebiasaan penting dalam EDA:

Jangan langsung menganalisis data sebelum mengetahui bentuk data yang sedang dianalisis.

L. Statistik Deskriptif

Gunakan:

```
df.describe()
```

Pandas akan menghasilkan statistik numerik seperti:

- count;
- mean;
- std;
- min;
- quartile;
- max.

Mahasiswa tidak perlu menghafalkan seluruh istilah statistik pada pertemuan ini.

Yang penting mereka dapat membaca:

```
count
mean
min
max
```

dan memahami bahwa `describe()` memberikan ringkasan awal dataset numerik.

M. Memilih Kolom

Untuk mengambil satu kolom:

```
df["Nilai"]
```

Atau:

```
df.Nilai
```

Tetapi biasakan:

```
df["Nilai"]
```

karena lebih eksplisit dan lebih aman untuk nama kolom tertentu.

Beberapa kolom:

```
df[["Nama", "Nilai"]]
```

Ini menghasilkan DataFrame baru.

N. Memilih Baris

Gunakan:

```
df.iloc[0]
```

untuk baris pertama.

Kemudian:

```
df.iloc[0:3]
```

untuk tiga baris pertama.

`iloc` memilih berdasarkan posisi integer.

Untuk label index, Pandas menyediakan `loc`.

Contoh:

```
df.loc[0]
```

Pada tahap ini cukup berikan konsep:

```
iloc → berdasarkan posisi
loc  → berdasarkan label
```

Jangan terlalu lama membahas perbedaan keduanya.

O. Filtering

Ini bagian paling penting.

Misalnya:

```
df[df["Nilai"] >= 75]
```

Hasilnya hanya mahasiswa dengan nilai minimal 75.

Secara konsep:

```
DATAFRAME
    ↓
KONDISI
    ↓
TRUE / FALSE
    ↓
FILTER
    ↓
DATA TERPILIH
```

Perhatikan hubungan dengan Modul 2 dan Modul 5.

Modul 2:

```
if nilai >= 60:
```

Modul 5:

```
nilai[nilai >= 60]
```

Modul 7:

```
df[df["Nilai"] >= 60]
```

Jadi mahasiswa sebenarnya mempelajari **konsep yang sama dalam bentuk yang semakin tinggi abstraksinya**.

P. Filtering Dua Kondisi

Misalnya:

Mahasiswa dengan nilai ≥ 70 dan prodi Informatika.

```
hasil = df[
    (df["Nilai"] >= 70) &
    (df["Prodi"] == "Informatika")
]

hasil
```

Untuk `OR`:

```
hasil = df[
    (df["Nilai"] >= 85) |
    (df["Prodi"] == "Sistem Informasi")
]
```

Ini menghubungkan langsung:

```
Modul 2
and / or
        ↓
Modul 5
Boolean array
        ↓
Modul 7
Boolean filtering DataFrame
```

Q. Menambahkan Kolom

Misalnya kita ingin membuat status kelulusan.

```
df["Status"] = df["Nilai"] >= 60

df
```

Hasil:

```
Nama    Nilai    Prodi              Status
Andi    80       Informatika        True
Budi    65       Informatika        True
Citra   90       Sistem Informasi   True
Dina    75       Informatika        True
```

Tetapi kita ingin:

```
Lulus
Tidak Lulus
```

Gunakan:

```
df["Status"] = np.where(
    df["Nilai"] >= 60,
    "Lulus",
    "Tidak Lulus"
)
```

Ini sekaligus menghubungkan Pandas dan NumPy.

R. Menambahkan Kolom Nilai Kategori

Gunakan function dari Modul 4:

```
def kategori_nilai(nilai):
    if nilai >= 80:
        return "Tinggi"
    elif nilai >= 60:
        return "Sedang"
    else:
        return "Rendah"
```

Kemudian:

```
df["Kategori"] = df["Nilai"].apply(kategori_nilai)
```

Sekarang mahasiswa melihat hubungan:

```
MODUL 4
Function
   ↓
MODUL 7
apply()
   ↓
Function diterapkan ke data
```

Ini konsep yang sangat penting.

S. Sorting

Urutkan nilai dari tertinggi:

```
df.sort_values("Nilai", ascending=False)
```

Dari terendah:

```
df.sort_values("Nilai")
```

Mahasiswa sudah pernah belajar:

```
sorted()
```

pada Modul 3.

Sekarang mereka melihat versi yang diterapkan pada DataFrame.

T. Menangani Missing Value

Ini bagian yang harus mulai diperkenalkan karena data nyata hampir tidak pernah sempurna.

Buat dataset:

```
data = {
    "Nama": ["Andi", "Budi", "Citra", "Dina", "Eko"],
    "Nilai": [80, 75, np.nan, 90, 65],
    "IPK": [3.7, 3.2, 3.8, np.nan, 3.1]
}

df = pd.DataFrame(data)

df
```

`NaN` berarti data numerik tidak tersedia.

Periksa:

```
df.isna()
```

Jumlah missing value:

```
df.isna().sum()
```

Hasil misalnya:

```
Nama      0
Nilai     1
IPK       1
```

U. Mengapa Missing Value Penting?

Misalkan rata-rata nilai dihitung:

```
df["Nilai"].mean()
```

Pandas secara default menangani `NaN` dengan cara tertentu dalam operasi statistik.

Tetapi mahasiswa harus memahami:

Data kosong bukan berarti otomatis nol.

Ini konsep yang sangat penting.

Misalnya:

```
Nilai = 0
```

berarti mahasiswa memperoleh nilai nol.

Sedangkan:

```
Nilai = NaN
```

berarti nilainya tidak tersedia atau tidak tercatat.

Keduanya memiliki makna yang berbeda.

V. Menghapus Missing Data

Untuk menghapus baris yang memiliki missing value:

```
df_clean = df.dropna()
```

Kemudian:

```
df_clean
```

Tetapi jangan langsung mengajarkan:

“Kalau ada data kosong, hapus saja.”

Itu justru kebiasaan buruk.

Tanyakan:

- Mengapa data kosong?
- Berapa banyak?
- Apakah penghapusan akan mengurangi data terlalu banyak?
- Apakah nilai kosong dapat diisi?

Ini mulai melatih **data reasoning**.

W. Mengisi Missing Value

Contoh:

```
df["Nilai"] = df["Nilai"].fillna(
    df["Nilai"].mean()
)
```

Artinya nilai kosong diisi menggunakan rata-rata.

Tetapi tekankan:

Mengisi missing value dengan mean bukan selalu solusi yang benar.

Ini hanya contoh teknik.

Mahasiswa belum perlu belajar metode imputasi yang kompleks.

X. Menghitung Data

Jumlah mahasiswa:

```
len(df)
```

atau:

```
df.shape[0]
```

Jumlah mahasiswa berdasarkan status:

```
df["Status"].value_counts()
```

Misalnya:

```
Lulus          8
Tidak Lulus    2
```

Ini merupakan bentuk **frequency counting** yang sudah dipelajari pada Modul 3.

Bedanya:

```
Modul 3
dictionary + loop

Modul 7
value_counts()
```

Y. Grouping

Sekarang masuk ke salah satu kemampuan paling penting Pandas:

```
groupby()
```

Misalnya:

```
df.groupby("Prodi")["Nilai"].mean()
```

Hasil:

```
Informatika          76.4
Sistem Informasi     82.7
```

Pertanyaan:

Prodi mana yang memiliki rata-rata nilai lebih tinggi?

Sekarang kita mulai melakukan analisis berdasarkan kelompok.

Z. Groupby + Multiple Statistics

```
df.groupby("Prodi")["Nilai"].agg(
    ["count", "mean", "min", "max"]
)
```

Hasil berupa tabel ringkasan.

Ini sangat penting karena mahasiswa mulai melihat bagaimana dataset dapat diubah menjadi **informasi**.

```
DATA
  ↓
```

```
GROUP
  ↓
AGGREGATE
  ↓
SUMMARY
  ↓
INFORMATION
```

AA. EDA: Apa yang Sebenarnya Dilakukan?

Sampai titik ini mahasiswa sudah memiliki berbagai teknik.

Sekarang gabungkan.

EDA sederhana dapat berupa:

```
1. Lihat data
2. Periksa struktur
3. Periksa tipe data
4. Periksa missing value
5. Lihat statistik
6. Pilih variabel penting
7. Filter data
8. Kelompokkan
9. Visualisasikan
10. Cari pola
11. Tulis insight
```

Jadi EDA bukan satu fungsi.

EDA adalah **proses eksplorasi**.

AB. Dataset Utama Praktikum

Untuk Modul 7, saya sarankan menggunakan satu dataset yang sama dari awal sampai akhir agar mahasiswa tidak terus-menerus berganti konteks.

Gunakan contoh data mahasiswa:

```
data = {
    "Nama": [
        "Andi", "Budi", "Citra", "Dina", "Eko",
        "Fajar", "Gita", "Hana", "Indra", "Joko",
        "Kiki", "Lina"
    ],
    "Prodi": [
        "Informatika", "Informatika", "Sistem Informasi",
```

```
        "Informatika", "Sistem Informasi", "Informatika",
        "Sistem Informasi", "Informatika", "Sistem Informasi",
        "Informatika", "Sistem Informasi", "Informatika"
    ],
    "UTS": [70, 65, 88, 75, 90, 55, 80, 72, 95, 60, 78, 85],
    "UAS": [75, 70, 92, 80, 85, 60, 88, 76, 90, 65, 82, 89],
    "Kehadiran": [90, 85, 95, 88, 92, 70, 96, 89, 98, 75, 91, 94]
}

df = pd.DataFrame(data)

df
```

Dataset ini sengaja mempunyai beberapa variabel yang menarik:

```
UTS
UAS
Kehadiran
Prodi
```

Sehingga mahasiswa dapat mulai bertanya:

Apakah kehadiran berkaitan dengan nilai?

Pertanyaan tersebut akan menjadi pintu masuk menuju analisis hubungan antarvariabel.

AC. Membuat Nilai Akhir

Gunakan konsep yang sudah dipelajari pada Modul 4:

```
df["Nilai_Akhir"] = (
    df["UTS"] * 0.4 +
    df["UAS"] * 0.6
)
```

Kemudian:

```
df.head()
```

Sekarang dataset memiliki:

```
Nama
Prodi
UTS
UAS
Kehadiran
Nilai_Akhir
```

AD. Menentukan Status

```
df["Status"] = np.where(
    df["Nilai_Akhir"] >= 60,
    "Lulus",
    "Tidak Lulus"
)
```

Sekarang:

```
df["Status"].value_counts()
```

Mahasiswa memperoleh ringkasan kelulusan.

AE. EDA Pertama

Minta mahasiswa menjawab pertanyaan berikut **tanpa langsung membuat grafik**:

1. Berapa jumlah mahasiswa?
2. Berapa mahasiswa setiap prodi?
3. Berapa rata-rata UTS?
4. Berapa rata-rata UAS?
5. Berapa rata-rata nilai akhir?
6. Siapa yang memperoleh nilai tertinggi?
7. Siapa yang memperoleh nilai terendah?
8. Berapa mahasiswa yang lulus?
9. Prodi mana yang memiliki rata-rata nilai akhir tertinggi?
10. Berapa rata-rata kehadiran?

Mahasiswa menggunakan Pandas untuk menemukan jawabannya.

AF. Visualisasi EDA

Sekarang manfaatkan Modul 6.

Distribusi Nilai

```
plt.hist(df["Nilai_Akhir"], bins=5)

plt.xlabel("Nilai Akhir")
plt.ylabel("Jumlah Mahasiswa")
plt.title("Distribusi Nilai Akhir")

plt.show()
```

Pertanyaan:

Apakah nilai mahasiswa cenderung berkumpul pada rentang tertentu?

AG. Bar Chart Rata-rata Prodi

```
rata_prodi = df.groupby("Prodi")["Nilai_Akhir"].mean()

rata_prodi.plot(kind="bar")

plt.ylabel("Rata-rata Nilai")
plt.title("Rata-rata Nilai Akhir per Prodi")

plt.show()
```

Sekarang mahasiswa dapat membandingkan kelompok secara visual.

AH. Scatter Plot Kehadiran vs Nilai

Ini mulai menarik.

```
plt.scatter(
    df["Kehadiran"],
    df["Nilai_Akhir"]
)

plt.xlabel("Kehadiran (%)")
plt.ylabel("Nilai Akhir")
plt.title("Kehadiran vs Nilai Akhir")

plt.show()
```

Pertanyaan:

Apakah mahasiswa dengan kehadiran tinggi cenderung memperoleh nilai lebih tinggi?

Jangan langsung menyatakan:

“Kehadiran menyebabkan nilai tinggi.”

Itu terlalu jauh.

Gunakan bahasa:

“Pada dataset ini terlihat kecenderungan hubungan antara kehadiran dan nilai.”

Ini sekaligus melatih mahasiswa agar **tidak menarik kesimpulan kausal secara sembarangan**.

AI. Correlation

Pandas dapat menghitung korelasi:

```
df[["UTS", "UAS", "Kehadiran", "Nilai_Akhir"]].corr()
```

Mahasiswa akan melihat matriks korelasi.

Jelaskan secara sederhana:

```
mendekati +1
→ hubungan positif kuat

mendekati 0
→ hubungan linear lemah

mendekati -1
→ hubungan negatif kuat
```

Tetapi tekankan:

Korelasi bukan sebab-akibat.

Ini merupakan prinsip yang sangat penting ketika mahasiswa mulai memasuki data science.

AJ. Challenge EDA

Sekarang berikan pertanyaan:

“Apakah kehadiran merupakan indikator yang cukup baik untuk memperkirakan nilai akhir mahasiswa?”

Mahasiswa harus melakukan:

```
1. Menghitung korelasi
2. Membuat scatter plot
3. Mengamati pola
4. Menuliskan kesimpulan sementara
```

Format jawaban:

```
Pertanyaan:
...
```

```
Hasil perhitungan:
...

Visualisasi:
...

Pola yang terlihat:
...

Kesimpulan sementara:
...
```

Dengan format tersebut mahasiswa mulai belajar **menulis analisis**, bukan sekadar menulis kode.

AK. Challenge Lebih Sulit — Mahasiswa Berisiko

Buat filter:

```
risiko = df[
    (df["Kehadiran"] < 80) &
    (df["Nilai_Akhir"] < 60)
]

risiko
```

Pertanyaan:

Siapa saja mahasiswa yang perlu mendapatkan perhatian akademik?

Kemudian:

```
df[
    (df["Kehadiran"] >= 90) &
    (df["Nilai_Akhir"] >= 80)
]
```

Pertanyaan:

Siapa yang menunjukkan performa akademik tinggi?

Di sini mahasiswa mulai belajar konsep **profiling sederhana**.

AL. Challenge Data yang Tidak Terduga

Buat mahasiswa mencari kasus:

Mahasiswa dengan kehadiran tinggi tetapi nilai rendah.

```
df[
    (df["Kehadiran"] >= 90) &
    (df["Nilai_Akhir"] < 60)
]
```

Kemudian:

Mahasiswa dengan kehadiran rendah tetapi nilai tinggi.

```
df[
    (df["Kehadiran"] < 80) &
    (df["Nilai_Akhir"] >= 80)
]
```

Pertanyaan:

Apa yang dapat kita pelajari dari data yang “tidak biasa” tersebut?

Ini sangat baik sebagai pengantar menuju **outlier dan anomaly detection**, yang akan berguna pada Data Mining.

AM. EDA sebagai Jembatan ke Data Mining

Mahasiswa perlu memahami perbedaannya.

EDA

Pertanyaannya:

“Apa yang terdapat dalam data?”

Contoh:

- rata-rata berapa?
- distribusinya bagaimana?

- kelompok mana yang berbeda?
- apakah ada missing value?
- apakah ada pola?
- apakah ada outlier?

Data Mining

Pertanyaannya mulai berubah:

“Apa pola atau pengetahuan yang dapat kita temukan secara sistematis dari data?”

Contoh:

- clustering;
- classification;
- association;
- anomaly detection.

Maka:

```
MODUL 7
  EDA
   ↓
Memahami data
   ↓
MODUL 8
Data Mining
   ↓
Menemukan pola
```

AN. Mini Project Modul 7

“Exploratory Data Analysis Mahasiswa”

Mahasiswa menggunakan dataset yang telah diberikan.

Notebook harus menghasilkan:

1. Informasi Dataset

```
Jumlah baris
Jumlah kolom
Nama kolom
Tipe data
```

2. Statistik

```
Mean
Median
Minimum
Maksimum
```

untuk:

- UTS;
- UAS;
- Kehadiran;
- Nilai Akhir.

3. Analisis Kelompok

Bandingkan rata-rata:

```
Informatika
vs
Sistem Informasi
```

4. Analisis Kelulusan

Tampilkan:

```
Jumlah lulus
Jumlah tidak lulus
Persentase lulus
```

5. Visualisasi

Minimal:

- histogram nilai akhir;
- bar chart rata-rata nilai per prodi;
- scatter plot kehadiran vs nilai akhir.

6. Insight

Minimal lima temuan.

Contoh format:

“Rata-rata nilai akhir mahasiswa Prodi X lebih tinggi dibandingkan Prodi Y.”

atau:

“Pada dataset ini terdapat kecenderungan positif antara kehadiran dan nilai akhir.”

AO. Aturan Penting: Jangan Hanya Menampilkan Grafik

Mahasiswa sering membuat kesalahan:

```
plt.show()
```

selesai.

Padahal grafik harus dibaca.

Setiap visualisasi harus diikuti:

```
Apa yang terlihat?
        ↓
Apa pola utamanya?
        ↓
Apa informasi yang dapat diambil?
```

Misalnya:

“Histogram menunjukkan sebagian besar nilai berada pada rentang 70–90.”

Itu jauh lebih bernilai daripada sekadar:

“Berikut grafik nilai.”

AP. Tugas Tantangan

Gunakan dataset yang sedikit lebih besar.

Mahasiswa membuat data sintetis:

```
np.random.seed(42)

n = 200

data = pd.DataFrame({
    "Jam_Belajar": np.random.randint(1, 10, n),
    "Kehadiran": np.random.randint(60, 101, n),
    "Nilai_UTS": np.random.randint(40, 101, n),
    "Nilai_UAS": np.random.randint(40, 101, n)
})
```

Buat:

```
data["Nilai_Akhir"] = (
    data["Nilai_UTS"] * 0.4 +
    data["Nilai_UAS"] * 0.6
)
```

Kemudian mahasiswa mencari:

1. rata-rata jam belajar;
2. rata-rata kehadiran;
3. rata-rata nilai akhir;
4. korelasi jam belajar dengan nilai;
5. korelasi kehadiran dengan nilai;
6. mahasiswa dengan nilai tertinggi;
7. mahasiswa dengan nilai terendah;
8. distribusi nilai akhir;
9. hubungan jam belajar dan nilai;
10. hubungan kehadiran dan nilai.

AQ. Pertanyaan Kritis

Berikan pertanyaan ini sebagai bagian dari tugas:

Dataset menunjukkan bahwa mahasiswa yang belajar lebih lama cenderung memperoleh nilai lebih tinggi. Apakah kita boleh menyimpulkan bahwa menambah jam belajar pasti menyebabkan nilai mahasiswa meningkat?

Mahasiswa harus menjawab:

Tidak langsung.

Karena hubungan statistik tidak otomatis membuktikan sebab-akibat.

Mereka harus mulai mengenal perbedaan:

```
Korelasi
≠
Kausalitas
```

Ini merupakan bekal berpikir kritis yang sangat penting sebelum masuk ke data mining dan machine learning.

AR. Penggunaan AI pada Modul 7

AI mulai dapat digunakan untuk membantu **membaca dan mempertanyakan data**, tetapi mahasiswa tetap harus menjalankan analisis sendiri.

Contoh prompt:

Berikut hasil EDA saya. Ajukan lima pertanyaan kritis yang seharusnya saya periksa sebelum membuat kesimpulan.

Atau:

Saya menemukan korelasi 0,72 antara kehadiran dan nilai. Berikan kemungkinan interpretasi dan kesalahan interpretasi yang mungkin saya lakukan.

Atau:

Periksa kode Pandas saya. Jangan ubah kodenya. Identifikasi bagian yang berpotensi menghasilkan kesimpulan yang keliru.

Ini lebih cocok daripada:

“Buatkan EDA dataset ini.”

Karena tujuan modul adalah mahasiswa belajar **melakukan eksplorasi**, bukan menyerahkan eksplorasi kepada AI.

AS. Debugging Challenge

Berikan kode:

```
df[df["Nilai"] > 70 and df["Prodi"] == "Informatika"]
```

Program akan bermasalah.

Tanyakan:

Mengapa kita menggunakan `&`, bukan `and`?

Perbaikan:

```
df[
    (df["Nilai"] > 70) &
    (df["Prodi"] == "Informatika")
]
```

Kemudian berikan:

```
df["nilai"]
```

padahal kolomnya:

```
"Nilai"
```

Mahasiswa akan mendapatkan `KeyError`.

Tujuan:

Membaca error Pandas, bukan langsung bertanya kepada AI.

AT. Struktur Notebook Google Colab

```
MODUL 7
PANDAS DAN EXPLORATORY DATA ANALYSIS

Nama:
NIM:
Kelas:

1. Tujuan Praktikum

2. Review NumPy dan Visualisasi

3. Mengenal Pandas
   3.1 Series
   3.2 DataFrame

4. Membuat DataFrame

5. Membaca Dataset

6. Mengenal Dataset
   6.1 head()
   6.2 tail()
   6.3 shape
   6.4 info()
   6.5 dtypes
   6.6 describe()

7. Selecting Data

8. Filtering Data

9. Menambah Kolom

10. Sorting

11. Missing Value
```

12. Groupby

13. Statistik Deskriptif

14. EDA

15. Visualisasi EDA

16. Correlation

17. Data Challenge

18. Mini Project

19. Insight

20. Refleksi

AU. Pola Notebook yang Mulai Berubah

Pada Modul 1–4 pola utamanya:

```
Masalah
↓
Algoritma
↓
Kode
↓
Output
```

Mulai Modul 7 pola berubah menjadi:

```
Pertanyaan
↓
Data
↓
Eksplorasi
↓
Cleaning
↓
Analisis
↓
Visualisasi
↓
Insight
```

Ini perubahan penting.

Mahasiswa mulai berpindah dari **programmer pemula** menjadi **calon data analyst/data scientist**.

AV. Rubrik Penilaian

Komponen	Bobot
Pemahaman DataFrame	10%
Selecting & Filtering	15%
Data Cleaning	10%
Statistik Deskriptif	10%
Groupby & Aggregation	15%
Visualisasi EDA	10%
Data Challenge	10%
Interpretasi / Insight	15%
Kerapian Notebook	5%
Total	**100%**

Saya sengaja memberi **15% pada interpretasi/insight**.

Mulai Modul 7, mahasiswa tidak cukup dinilai dari:

- “Apakah kodenya benar?”
- Mereka juga harus dinilai dari:
- “Apakah dia memahami apa yang dihasilkan kode tersebut?”

AW. Bacaan Lanjutan dari Dua Buku

Di bagian ini ada hal yang perlu kita luruskan agar modul tidak memberikan referensi yang keliru.

Python Crash Course 3rd Edition tidak merupakan buku Pandas khusus. Namun buku ini memiliki bagian data analysis dan visualisasi. Indeksnya menunjukkan pembahasan data analysis serta proyek visualisasi dengan Matplotlib dan Plotly, termasuk pengolahan data dari file CSV.

Untuk memperkuat fondasi yang diperlukan sebelum Pandas, mahasiswa dapat mengulang:

Python Crash Course — Chapter 4: Working with Lists

terutama:

- numerical lists;

- `min()`;
- `max()`;
- `sum()`;
- list comprehension.

Buku tersebut memang membahas fungsi-fungsi statistik sederhana tersebut dalam konteks list.

Kemudian bagian data dari:

Chapter 16–17 — Data Visualization

karena mahasiswa akan menggabungkan Pandas dengan Matplotlib yang sudah diperkenalkan dalam Modul 6. Edisi ketiga menggunakan versi Matplotlib dan Plotly yang lebih baru, dan proyek visualisasinya memang dirancang untuk eksplorasi data secara visual.

Untuk **Think Python 3rd Edition**, buku ini juga bukan manual Pandas. Justru pada bagian akhir buku, Downey secara eksplisit menyebut bahwa minat ke data science dapat dilanjutkan melalui *Think Stats: Exploratory Data Analysis* dan menjelaskan bahwa buku-buku data science tersebut menggunakan NumPy, SciPy, Pandas, dan library Python lainnya.

Jadi pembagian referensinya sebaiknya:

```
Think Python
      +
Python Crash Course
      ↓
Fondasi Python
      ↓
    NumPy
      ↓
  Matplotlib
      ↓
   Pandas
      ↓
    EDA
```

Untuk **materi teknis Pandas**, kita memang membutuhkan sumber tambahan/dokumentasi Pandas. Sumber yang tersedia di File Library juga secara eksplisit menjelaskan `Series`, `DataFrame`, filtering, cleaning, analisis, dan penggunaan `import pandas as pd`.

Dengan demikian, kita tidak memaksakan dua buku utama untuk menjelaskan sesuatu yang memang bukan fokus utamanya.

AX. Hubungan dengan Materi Sebelumnya

Ini bagian yang sebaiknya selalu muncul di modul agar mahasiswa melihat kesinambungan.

Modul 2

```
if nilai >= 60:
```

Modul 3

```
for n in nilai:
    if n >= 60:
        ...
```

Modul 5

```
nilai[nilai >= 60]
```

Modul 7

```
df[df["Nilai"] >= 60]
```

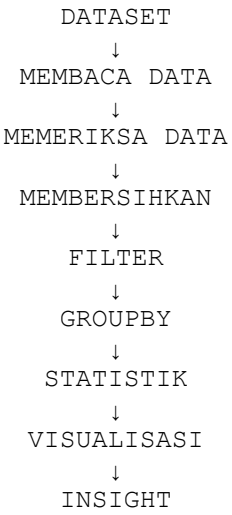
Jadi mahasiswa sebenarnya sedang belajar **satu konsep yang sama dalam level abstraksi yang semakin tinggi**.

```
LOGIKA
 ↓
LOOP
 ↓
LIST
 ↓
NUMPY ARRAY
 ↓
PANDAS DATAFRAME
 ↓
DATA ANALYSIS
```

Ini sangat penting agar mahasiswa tidak merasa setiap modul adalah materi baru yang terputus.

AY. Hubungan dengan Modul 8

Setelah Modul 7 selesai, mahasiswa sudah mampu:



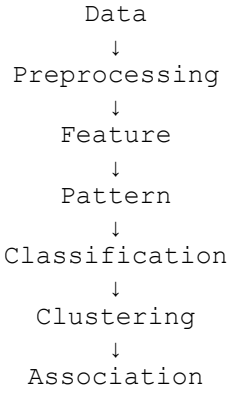
Maka pertanyaan berikutnya adalah:

“Bagaimana kalau kita ingin menemukan pola yang tidak langsung terlihat?”

Itulah Modul 8:

MODUL 8 — DATA MINING

Baru di sana mahasiswa mulai diperkenalkan secara sederhana pada:



Dan saya menyarankan **Modul 8 jangan langsung menggunakan algoritma yang terlalu berat**. Untuk mahasiswa semester 1, lebih baik mereka memahami konsep melalui dataset kecil terlebih dahulu, misalnya:

“Bisakah kita mengelompokkan mahasiswa berdasarkan pola belajar dan kehadiran?”

sebelum masuk ke algoritma clustering yang sebenarnya.

Dengan demikian transisinya menjadi sangat natural:

Modul 7 mengajarkan mahasiswa membaca dan memahami data.

Modul 8 mengajarkan mahasiswa mulai mencari pola tersembunyi dalam data.

MODUL 8

Data Mining dengan Python: Menemukan Pola dari Data

Mata Kuliah	Praktikum Algoritma dan Pemrograman
Semester	1
Pertemuan	8 dari 10
Durasi	100 menit
Platform	Google Colab
Bahasa	Python 3
Library	Pandas, NumPy, Matplotlib, Scikit-learn
Level	Pemula → pengantar Data Mining/Machine Learning

Modul 8 merupakan lompatan penting. Pada Modul 7 mahasiswa belajar bertanya:

“Apa yang ada di dalam data?”

Sekarang pertanyaannya berubah menjadi:

“Pola apa yang dapat ditemukan dari data?”

Peta jalan yang sudah kita susun memang menempatkan **Modul 8 — Data Mining** setelah Pandas & EDA, kemudian dilanjutkan Text Mining dan Mini Project Data Science.

Ada satu hal penting: **Think Python dan Python Crash Course bukan buku data mining.** Keduanya kita gunakan sebagai fondasi kemampuan Python, problem solving, data structures, functions, testing, dan visualisasi. Python Crash Course sendiri menyatakan tujuan utamanya adalah membangun kemampuan programming umum dan Python, kemudian membawa pembaca ke proyek data visualization. Think Python juga menekankan konsep fundamental Python dan penggunaan Jupyter Notebook serta AI coding assistants.

Karena itu, untuk Modul 8 kita perlu memperkenalkan **Scikit-learn** sebagai library tambahan. Ini bukan mengganti dua buku utama, tetapi melanjutkan fondasi yang sudah dibangun.

A. Mengapa Data Mining Sudah Diperkenalkan di Semester 1?

Sekilas mungkin terlalu cepat.

Mahasiswa baru belajar Python beberapa pertemuan, tetapi sekarang sudah diminta melakukan data mining.

Justru di sinilah desain mata kuliah ini berbeda dari praktikum pemrograman konvensional.

Kita tidak bermaksud menjadikan mahasiswa semester 1 sebagai machine learning engineer.

Targetnya jauh lebih sederhana:

```
Mahasiswa memahami:
```

```
                    data
                     ↓
                   fitur
                     ↓
                   pola
                     ↓
                  model
                     ↓
          prediksi / kelompok
                     ↓
               evaluasi
                     ↓
           interpretasi
```

Mereka memperoleh **preview** terhadap mata kuliah berikutnya.

Jadi ketika nanti mengambil:

- Statistik;
- Data Science;
- Basis Data;
- Machine Learning;
- Data Mining;
- Artificial Intelligence;

mereka tidak lagi melihat istilah tersebut sebagai sesuatu yang asing.

B. Dari EDA ke Data Mining

Modul 7:

```
                    DATA
                     ↓
                    EDA
                     ↓
          Apa yang terjadi?
```

```
        ↓
    Insight
```

Modul 8:

```
          DATA
           ↓
       FEATURES
           ↓
      ALGORITHM
           ↓
  PATTERN / MODEL
           ↓
      PREDICTION
```

Perbedaannya sangat penting.

EDA lebih bersifat eksploratif.

Data mining mulai menggunakan algoritma untuk menemukan pola secara sistematis.

C. Dua Dunia Data Mining

Untuk mahasiswa semester 1, cukup diperkenalkan dua pendekatan utama.

1. Supervised Learning

Data sudah memiliki jawaban/label.

Contoh:

Jam Belajar	Kehadiran	Nilai	Status
5	90	85	Lulus
2	70	55	Tidak
7	95	90	Lulus

Kita ingin mempelajari pola:

```
Features
   ↓
Model
   ↓
Target
```

Contoh target:

```
Lulus
Tidak Lulus
```

Ini disebut **classification**.

2. Unsupervised Learning

Data tidak memiliki label.

Misalnya:

```
Mahasiswa
 ↓
Jam belajar
Kehadiran
Nilai
 ↓
Algoritma
 ↓
Kelompok 1
Kelompok 2
Kelompok 3
```

Kita tidak memberi tahu komputer kelompoknya.

Algoritma mencoba menemukan kelompok berdasarkan kemiripan.

Ini disebut **clustering**.

Scikit-learn sendiri membedakan supervised learning seperti classification/regression dan unsupervised learning seperti clustering. (Scikit-Learn)

D. Mengapa Saya Memilih KNN dan K-Means?

Untuk mahasiswa semester 1, jangan langsung memasukkan:

- Decision Tree;
- Random Forest;
- SVM;
- Neural Network;
- XGBoost;
- Deep Learning.

Bukan karena algoritma tersebut tidak penting.

Justru karena terlalu banyak abstraksi.

Saya memilih dua algoritma yang mudah divisualisasikan:

```
KNN
→ classification
→ "data baru paling mirip dengan siapa?"

K-Means
→ clustering
→ "data ini secara alami terbagi menjadi kelompok apa?"
```

KNN sangat cocok untuk pengenalan karena prinsipnya sederhana: mencari sejumlah tetangga terdekat dan menentukan kelas berdasarkan suara mayoritas. Scikit-learn mendokumentasikan `KNeighborsClassifier` dengan prinsip tersebut. (Scikit-Learn)

K-Means juga cocok untuk pemula karena mahasiswa dapat melihat titik-titik data, centroid, dan kelompok secara visual. Scikit-learn mendefinisikan K-Means sebagai metode clustering yang membagi data ke dalam sejumlah kelompok dengan meminimalkan jarak kuadrat terhadap centroid. (Scikit-Learn)

E. Tujuan Pembelajaran

Setelah menyelesaikan modul ini, mahasiswa mampu:

1. Menjelaskan pengertian sederhana data mining.
2. Membedakan supervised dan unsupervised learning.
3. Menjelaskan feature dan target.
4. Menjelaskan classification dan clustering.
5. Mengenal Scikit-learn.
6. Membagi dataset menjadi training dan testing.
7. Membuat model K-Nearest Neighbors.
8. Melakukan prediksi.
9. Mengukur akurasi sederhana.
10. Memahami konsep scaling.
11. Membuat model K-Means.
12. Menentukan jumlah cluster secara sederhana.
13. Memvisualisasikan hasil clustering.
14. Membandingkan hasil model dengan data asli.
15. Memahami bahwa model dapat menghasilkan kesalahan.
16. Menghindari kesimpulan berlebihan dari hasil model.
17. Menjelaskan keterbatasan dataset dan model.

F. Skenario Praktikum 100 Menit

Waktu	Kegiatan
0–10 menit	Review EDA
10–20 menit	Konsep Data Mining
20–30 menit	Feature, target, train/test
30–50 menit	Classification dengan KNN
50–60 menit	Evaluasi model
60–65 menit	Scaling
65–85 menit	Clustering dengan K-Means
85–93 menit	Visualisasi cluster
93–98 menit	Data Challenge
98–100 menit	Exit Ticket

G. Bagian 1 — Apa Itu Data Mining?

Mulai dengan contoh sederhana.

Kita mempunyai:

```
Mahasiswa

Jam Belajar
Kehadiran
Nilai UTS
Nilai UAS
```

Dari data tersebut kita mungkin ingin mengetahui:

Mahasiswa mana yang berpotensi lulus?

atau:

Apakah mahasiswa dapat dikelompokkan berdasarkan pola belajarnya?

atau:

Apakah terdapat kelompok mahasiswa dengan karakteristik yang berbeda?

Di sinilah data mining mulai bekerja.

Secara sederhana:

Data mining adalah proses menggunakan metode komputasional untuk menemukan pola atau informasi yang berguna dari kumpulan data.

Untuk mahasiswa semester 1, definisi ini sudah cukup.

H. Jangan Samakan Data Mining dengan “Mencari Data”

Ini harus ditegaskan.

Data mining bukan:

```
Google
  ↓
mencari data
```

Data mining:

```
Dataset
  ↓
Algoritma
  ↓
Pola
  ↓
Informasi
```

I. Konsep Feature dan Target

Misalnya:

Jam_Belajar	Kehadiran	UTS	UAS	Status
5	90	80	85	Lulus
2	70	55	60	Tidak
7	95	90	92	Lulus

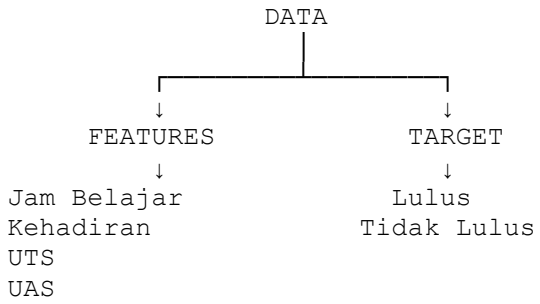
Maka:

```
Feature:
Jam_Belajar
Kehadiran
UTS
UAS
```

Target:

```
Status
```

Visual:



Istilah ini harus dikuasai karena akan terus muncul dalam machine learning.

J. X dan y

Dalam Scikit-learn kita biasanya menggunakan:

`X`

untuk features.

Sedangkan:

`y`

untuk target.

Contoh:

```
X = df[
    ["Jam_Belajar", "Kehadiran", "UTS", "UAS"]
]

y = df["Status"]
```

Jangan meminta mahasiswa menghafalkan huruf X dan y tanpa konteks.

Jelaskan:

```
X = informasi yang digunakan model
y = jawaban yang ingin dipelajari model
```

K. Dataset Praktikum

Gunakan dataset mahasiswa yang konsisten dengan Modul 7.

Kali ini tambahkan variabel `Jam_Belajar`.

```
import pandas as pd
import numpy as np

data = {
    "Nama": [
        "Andi", "Budi", "Citra", "Dina", "Eko",
        "Fajar", "Gita", "Hana", "Indra", "Joko",
        "Kiki", "Lina"
    ],

    "Jam_Belajar": [
        6, 3, 8, 5, 7, 2,
        7, 5, 9, 3, 6, 8
    ],

    "Kehadiran": [
        90, 75, 95, 88, 92, 65,
        96, 89, 98, 70, 91, 94
    ],

    "UTS": [
        80, 65, 88, 75, 90, 55,
        80, 72, 95, 60, 78, 85
    ],

    "UAS": [
        85, 70, 92, 80, 85, 60,
        88, 76, 90, 65, 82, 89
    ]
}

df = pd.DataFrame(data)
```

Kemudian:

```
df["Nilai_Akhir"] = (
    df["UTS"] * 0.4 +
    df["UAS"] * 0.6
)
```

Buat target:

```
df["Status"] = np.where(
    df["Nilai_Akhir"] >= 60,
    "Lulus",
    "Tidak Lulus"
)
```

L. Pertanyaan Sebelum Machine Learning

Sebelum menggunakan library, mahasiswa harus menjawab:

Apakah kita sebenarnya membutuhkan machine learning untuk dataset 12 mahasiswa?

Jawabannya:

Tidak.

Kita bisa melihat datanya secara langsung.

Ini sengaja dilakukan.

Tujuannya agar mahasiswa memahami:

Machine learning bukan solusi untuk semua masalah.

Pada dataset sangat kecil dan sederhana, algoritma machine learning bahkan bisa menjadi berlebihan.

Tetapi dataset ini cocok untuk **belajar konsep**.

M. Bagian 2 — Scikit-learn

Import:

```
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import accuracy_score
```

Scikit-learn menyediakan berbagai algoritma machine learning dan utilitas preprocessing serta evaluasi. Dokumentasi resminya saat ini menunjukkan KNN, clustering, train/test split, scaling, dan berbagai metode evaluasi sebagai bagian dari ekosistemnya. (Scikit-Learn)

N. Mengapa Dataset Harus Dibagi?

Jangan lakukan:

```
DATA
↓
Model
↓
uji dengan DATA YANG SAMA
```

Karena model sudah melihat data tersebut.

Lebih baik:

```
DATA
 ↓
┌──────────────┐
↓              ↓
TRAIN          TEST
↓              ↓
belajar        menguji
```

Scikit-learn menyediakan `train_test_split()` untuk membagi data menjadi subset training dan testing. (Scikit-Learn)

O. Train-Test Split

Misalnya:

```
X = df[
    ["Jam_Belajar", "Kehadiran", "UTS", "UAS"]
]

y = df["Status"]
```

Kemudian:

```
X_train, X_test, y_train, y_test = train_test_split(
    X,
    y,
    test_size=0.25,
    random_state=42
)
```

Sekarang:

```
X_train → fitur untuk belajar
y_train → jawaban training

X_test → fitur untuk pengujian
y_test → jawaban sebenarnya
```

P. Mengapa `random_state=42`?

Mahasiswa akan sering melihat:

```
random_state=42
```

Jelaskan sederhana:

Agar pembagian acak dapat direproduksi.

Tanpa itu, hasil pembagian dapat berbeda ketika kode dijalankan kembali.

Angka `42` bukan angka sakral.

Bisa:

```
random_state=10
```

atau:

```
random_state=123
```

Yang penting nilainya dibuat tetap ketika kita ingin memperoleh pembagian yang sama.

Q. Bagian 3 — K-Nearest Neighbors

Sekarang konsep utama.

Bayangkan ada mahasiswa baru:

```
Jam belajar = 6
Kehadiran = 90
UTS = 78
UAS = 82
```

Kita ingin memperkirakan:

```
Lulus?
```

KNN menggunakan gagasan:

Data baru kemungkinan memiliki kelas yang sama dengan data yang paling mirip di sekitarnya.

Scikit-learn menjelaskan bahwa KNN classification menentukan label berdasarkan tetangga terdekat dan voting mayoritas. (Scikit-Learn)

R. Analogi KNN

Bayangkan peta:

```
        ● Lulus
    ● Lulus

        ? ← mahasiswa baru

      ● Tidak
        ● Tidak
```

Jika tiga titik terdekat:

```
Lulus
Lulus
Tidak Lulus
```

maka:

```
2 vs 1
↓
Lulus
```

Itulah gagasan dasar KNN.

S. Membuat Model KNN

```
model = KNeighborsClassifier(
    n_neighbors=3
)
```

Artinya:

Gunakan 3 tetangga terdekat.

Kemudian:

```
model.fit(
    X_train,
    y_train
)
```

Model sekarang telah "belajar" dari data training.

T. Prediksi

```
y_pred = model.predict(X_test)

print(y_pred)
```

Bandingkan:

```
print(y_test.values)
print(y_pred)
```

Mahasiswa dapat melihat:

```
Data sebenarnya
vs
Prediksi model
```

U. Evaluasi

Gunakan:

```
akurasi = accuracy_score(
    y_test,
    y_pred
)

print(akurasi)
```

Misalnya:

```
0.75
```

berarti:

```
75%
```

Secara sederhana:

Dari data test, sekitar 75% prediksi model benar.

Tetapi karena dataset kita sangat kecil, angka tersebut **jangan dianggap sebagai bukti bahwa model akan memiliki akurasi 75% pada populasi mahasiswa yang sebenarnya**.

Ini harus menjadi bagian dari pendidikan statistik dan machine learning sejak awal.

V. Accuracy Bukan Segalanya

Misalkan:

```
100 mahasiswa

95 Lulus
5 Tidak Lulus
```

Model yang selalu menjawab:

```
"Lulus"
```

akan mendapatkan:

```
95% accuracy
```

Padahal model tersebut gagal menemukan seluruh mahasiswa yang tidak lulus.

Karena itu mahasiswa perlu belajar:

Metrik evaluasi harus disesuaikan dengan tujuan.

Untuk Modul 8 semester 1, cukup fokus pada accuracy.

Confusion matrix dapat diperkenalkan sebagai bonus.

W. Confusion Matrix Sederhana

```
from sklearn.metrics import confusion_matrix

cm = confusion_matrix(
    y_test,
    y_pred
)

print(cm)
```

Jelaskan konsep:

```
             Prediksi
             Lulus   Tidak
Aktual
Lulus
Tidak
```

Tidak perlu masuk terlalu dalam ke precision dan recall.

Itu dapat menjadi materi mata kuliah machine learning berikutnya.

X. Bagian 4 — Mengapa Scaling Penting?

Ini bagian yang sangat bagus untuk mahasiswa.

Misalnya fitur:

```
Jam_Belajar   1–10
Kehadiran     60–100
UTS           40–100
UAS           40–100
```

Rentangnya berbeda.

KNN menggunakan jarak. Scikit-learn secara eksplisit menunjukkan bahwa scaling penting untuk KNN karena algoritmanya menggunakan jarak Euclidean. (Scikit-Learn)

Y. StandardScaler

Gunakan:

```
scaler = StandardScaler()

X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

Perhatikan:

```
TRAIN
↓
fit_transform

TEST
↓
transform
```

Jangan:

```
scaler.fit_transform(X_test)
```

Karena preprocessing sebaiknya dipelajari dari training data, kemudian diterapkan ke test data.

Untuk mahasiswa semester 1, cukup pahami prinsip tersebut.

Z. KNN dengan Scaling

```
model = KNeighborsClassifier(
    n_neighbors=3
)

model.fit(
    X_train_scaled,
    y_train
)

y_pred = model.predict(
    X_test_scaled
)
```

Kemudian:

```
accuracy_score(
    y_test,
    y_pred
)
```

Bandingkan hasil dengan model sebelum scaling.

Pertanyaan:

Apakah hasil selalu menjadi lebih baik setelah scaling?

Jawaban:

Tidak selalu.

Scaling adalah teknik yang tepat untuk algoritma tertentu dan jenis data tertentu; bukan tombol “buat model lebih akurat”.

AA. Pipeline

Untuk mahasiswa yang lebih cepat, perkenalkan:

```
from sklearn.pipeline import Pipeline

model = Pipeline([
    ("scaler", StandardScaler()),
    ("knn", KNeighborsClassifier(n_neighbors=3))
])
```

Kemudian:

```
model.fit(X_train, y_train)

y_pred = model.predict(X_test)
```

Ini mulai memperkenalkan ide:

```
DATA
 ↓
SCALING
 ↓
MODEL
 ↓
PREDICTION
```

Scikit-learn sendiri menggunakan pola pipeline seperti ini dalam contoh resmi KNN. (Scikit-Learn)

Untuk mahasiswa semester 1, pipeline cukup dikenalkan sebagai **cara menggabungkan beberapa tahap pemrosesan**.

AB. Challenge KNN

Ubah:

```
n_neighbors=3
```

menjadi:

```
n_neighbors=1
```

kemudian:

```
n_neighbors=5
```

dan:

```
n_neighbors=7
```

Catat:

```
k = 1 → accuracy ...
k = 3 → accuracy ...
k = 5 → accuracy ...
k = 7 → accuracy ...
```

Pertanyaan:

Apakah semakin besar `k`, semakin baik?

Tidak.

Pemilihan `k` bergantung pada dataset. Dokumentasi Scikit-learn juga menekankan bahwa pilihan jumlah tetangga merupakan hal yang bergantung pada data. (Scikit-Learn)

AC. Bagian 5 — Unsupervised Learning

Sekarang **hapus target**.

Tidak ada:

```
Lulus
Tidak Lulus
```

Kita hanya memiliki:

```
Jam_Belajar
Kehadiran
Nilai
```

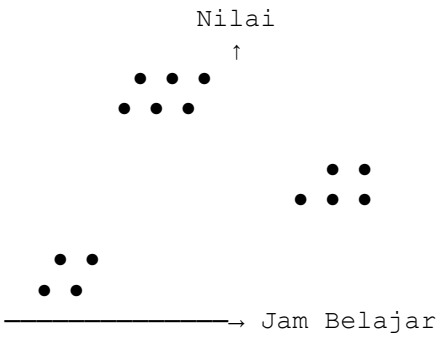
Pertanyaan:

Bisakah mahasiswa dikelompokkan berdasarkan kemiripan karakteristiknya?

Ini adalah **clustering**.

AD. Contoh Konseptual

Bayangkan:



Secara visual mungkin terdapat tiga kelompok.

Tetapi kita belum memberi nama:

```
Kelompok 1
Kelompok 2
Kelompok 3
```

Algoritma yang akan mencoba menemukan kelompok tersebut adalah K-Means.

AE. K-Means

Import:

```
from sklearn.cluster import KMeans
```

Siapkan data:

```
X_cluster = df[
    ["Jam_Belajar", "Nilai_Akhir"]
]
```

Kemudian:

```
kmeans = KMeans(
    n_clusters=3,
    random_state=42,
    n_init="auto"
)
```

Jalankan:

```
df["Cluster"] = kmeans.fit_predict(
    X_cluster
)
```

Sekarang dataset memiliki:

```
Nama
Jam_Belajar
Kehadiran
UTS
UAS
Nilai_Akhir
Status
Cluster
```

Scikit-learn menyediakan `KMeans` sebagai salah satu algoritma clustering utama. (Scikit-Learn)

AF. Melihat Hasil Cluster

```
df[
    [
        "Nama",
        "Jam_Belajar",
        "Nilai_Akhir",
        "Cluster"
    ]
]
```

Mahasiswa mungkin mendapatkan:

```
Andi     6    83    1
Budi     3    68    2
Citra    8    90    0
...
```

Jangan mengatakan:

```
Cluster 0 = mahasiswa pintar
Cluster 1 = mahasiswa sedang
Cluster 2 = mahasiswa malas
```

Belum tentu.

Nomor cluster hanyalah label algoritmik.

AG. Kesalahan yang Harus Ditekankan

Ini sangat penting.

Jika hasil:

```
Cluster 0
Cluster 1
Cluster 2
```

maka:

Nomor cluster tidak mempunyai makna intrinsik.

Cluster 0 tidak otomatis “terbaik”.

Kita harus melihat karakteristik masing-masing cluster.

AH. Menganalisis Karakteristik Cluster

Gunakan:

```
df.groupby("Cluster")[
    ["Jam_Belajar", "Kehadiran", "Nilai_Akhir"]
].mean()
```

Sekarang kita mungkin memperoleh:

```
Cluster 0
Jam belajar      7.8
Kehadiran       93
Nilai akhir     87

Cluster 1
Jam belajar      3.2
Kehadiran       73
Nilai akhir     66

Cluster 2
Jam belajar      5.4
Kehadiran       87
Nilai akhir     78
```

Baru kita dapat memberikan interpretasi:

```
Cluster 0
→ performa akademik tinggi

Cluster 1
→ performa relatif rendah

Cluster 2
→ performa menengah
```

Tetapi ini adalah **interpretasi kita**, bukan nama bawaan dari algoritma.

AI. Visualisasi Cluster

Gunakan:

```
plt.scatter(
    df["Jam_Belajar"],
    df["Nilai_Akhir"],
    c=df["Cluster"]
)
```

```
plt.xlabel("Jam Belajar")
plt.ylabel("Nilai Akhir")
plt.title("Clustering Mahasiswa")

plt.show()
```

Sekarang mahasiswa melihat hasil data mining secara visual.

Ini menggabungkan:

```
Modul 5
NumPy

Modul 6
Visualisasi

Modul 7
Pandas

Modul 8
Data Mining
```

AJ. Centroid

K-Means menghasilkan pusat cluster.

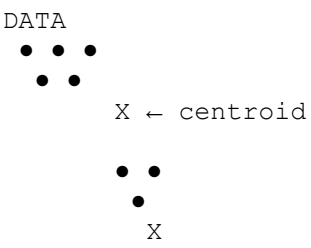
Lihat:

```
print(kmeans.cluster_centers_)
```

Visualisasikan:

```
centers = kmeans.cluster_centers_

plt.scatter(
    df["Jam_Belajar"],
    df["Nilai_Akhir"],
    c=df["Cluster"]
)

plt.scatter(
    centers[:, 0],
    centers[:, 1],
    marker="X",
    s=200
)

plt.xlabel("Jam Belajar")
plt.ylabel("Nilai Akhir")
plt.title("Cluster dan Centroid")

plt.show()
```

Sekarang mahasiswa dapat melihat:



Secara intuitif:

Centroid adalah titik pusat yang mewakili suatu cluster.

AK. Berapa Jumlah Cluster?

Ini pertanyaan penting.

Kita memilih:

```
n_clusters=3
```

Tetapi:

Mengapa 3?

Jangan menjawab:

Karena contohnya punya tiga kelompok.

Itu tidak cukup.

Kita dapat mencoba:

```
k_values = range(2, 7)

inertia = []

for k in k_values:
    model = KMeans(
        n_clusters=k,
        random_state=42,
        n_init="auto"
    )

    model.fit(X_cluster)

    inertia.append(model.inertia_)
```

Visualisasikan:

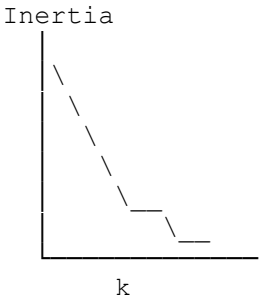
```
plt.plot(k_values, inertia, marker="o")

plt.xlabel("Jumlah Cluster")
plt.ylabel("Inertia")
plt.title("Elbow Method")

plt.show()
```

AL. Elbow Method

Jelaskan secara intuitif:

Kita mencoba beberapa jumlah cluster dan melihat bagaimana ukuran kesalahan internal berubah.

Biasanya kita mencari titik seperti:



Titik perubahan tajam disebut **elbow**.

Namun jangan mengatakan:

“Elbow selalu memberikan jumlah cluster yang benar.”

Tidak.

Pemilihan jumlah cluster tetap membutuhkan pertimbangan data dan tujuan analisis. Dokumentasi Scikit-learn bahkan menunjukkan bahwa tidak ada satu jumlah cluster yang selalu “benar” untuk semua kondisi. (Scikit-Learn)

AM. Silhouette Score — Bonus

Untuk mahasiswa yang lebih cepat:

```
from sklearn.metrics import silhouette_score
```

Kemudian:

```
score = silhouette_score(
    X_cluster,
    df["Cluster"]
)

print(score)
```

Secara sederhana:

```
mendekati +1
→ cluster relatif terpisah dengan baik

sekitar 0
→ titik berada dekat batas cluster

negatif
→ kemungkinan penempatan cluster kurang baik
```

Dokumentasi Scikit-learn menjelaskan silhouette coefficient berada pada rentang -1 sampai +1 dan digunakan untuk melihat seberapa baik pemisahan antarcluster. (Scikit-Learn)

Tetapi **silhouette score jangan menjadi materi wajib** untuk semua mahasiswa. Jadikan pengayaan.

AN. Supervised vs Unsupervised

Sekarang mahasiswa membuat tabel perbandingan:

Aspek	Classification	Clustering
Jenis	Supervised	Unsupervised
Target	Ada	Tidak ada
Tujuan	Memprediksi kelas	Menemukan kelompok
Contoh	Lulus/Tidak	Kelompok mahasiswa
Algoritma	KNN	K-Means
Output	Label	Cluster

Ini harus menjadi salah satu hasil belajar utama Modul 8.

AO. Mini Challenge 1 — Klasifikasi

Gunakan dataset mahasiswa.

Tugas:

Buat model yang memprediksi apakah mahasiswa akan lulus atau tidak berdasarkan:

```
Jam_Belajar
Kehadiran
UTS
UAS
```

Langkah:

```
1. Tentukan X
2. Tentukan y
3. Split train/test
4. Scaling
5. Buat KNN
6. Fit
7. Predict
8. Hitung accuracy
9. Analisis hasil
```

Mahasiswa wajib menjelaskan setiap langkah.

Bukan hanya menyalin kode.

AP. Mini Challenge 2 — Clustering

Gunakan:

```
Jam_Belajar
Kehadiran
Nilai_Akhir
```

Tugas:

Temukan tiga kelompok mahasiswa menggunakan K-Means.

Kemudian:

1. tampilkan hasil cluster;
2. hitung rata-rata setiap cluster;
3. visualisasikan;
4. berikan nama interpretatif pada cluster;
5. jelaskan alasan pemberian nama.

Contoh:

```
Cluster 0
→ Performa Tinggi
```

```
Cluster 1
→ Performa Menengah

Cluster 2
→ Perlu Perhatian
```

Tetapi mahasiswa harus mendasarkan penamaan tersebut pada statistik cluster.

AQ. Challenge Logika — Tanpa Library

Ini penting untuk menjaga identitas mata kuliah sebagai **Algoritma dan Pemrograman**.

Tanyakan:

Bisakah Anda menjelaskan secara manual bagaimana KNN menentukan kelas?

Mahasiswa tidak perlu membuat algoritma lengkap.

Cukup:

```
Data baru
 ↓
Hitung jarak ke data training
 ↓
Urutkan berdasarkan jarak
 ↓
Ambil k terdekat
 ↓
Hitung suara
 ↓
Kelas mayoritas
```

Dengan demikian mereka tetap memahami algoritma di balik library.

AR. Tantangan Jarak Euclidean

Berikan dua titik:

```
A = (2, 3)
B = (5, 7)
```

Minta mahasiswa menghitung jaraknya secara manual:

```
√((5-2)² + (7-3)²)
```

Kemudian gunakan Python:

```
import numpy as np

A = np.array([2, 3])
B = np.array([5, 7])

jarak = np.sqrt(
    np.sum((A - B) ** 2)
)

print(jarak)
```

Mahasiswa melihat hubungan:

```
Matematika
    ↓
Algoritma
    ↓
Python
    ↓
Library
    ↓
Machine Learning
```

Ini sangat sesuai dengan filosofi mata kuliah.

AS. Challenge Lebih Sulit — Prediksi Mahasiswa Baru

Buat mahasiswa baru:

```
mahasiswa_baru = pd.DataFrame({
    "Jam_Belajar": [6],
    "Kehadiran": [92],
    "UTS": [82],
    "UAS": [86]
})
```

Prediksi:

```
prediksi = model.predict(
    mahasiswa_baru
)

print(prediksi)
```

Pertanyaan:

Apa status mahasiswa tersebut menurut model?

Kemudian pertanyaan yang lebih penting:

Apakah prediksi model berarti mahasiswa tersebut **pasti** lulus?

Jawabannya:

Tidak.

Model hanya memberikan prediksi berdasarkan pola data training.

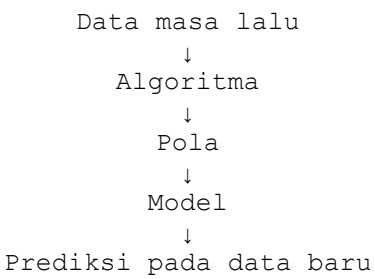
AT. Ini Harus Ditekankan: Model Bukan Peramal

Mahasiswa sering mendapatkan kesan:

```
Machine Learning
=
komputer tahu masa depan
```

Koreksi sejak dini.

Lebih tepat:



Prediksi selalu memiliki ketidakpastian.

AU. Overfitting — Pengenalan Saja

Mahasiswa tidak perlu belajar rumus.

Cukup gunakan analogi:

Mahasiswa yang menghafal seluruh soal latihan belum tentu memahami konsep.

Dalam machine learning:

```
Model terlalu cocok dengan training data
↓
Training sangat bagus
↓
Data baru buruk
```

Itulah gambaran sederhana **overfitting**.

Dokumentasi Scikit-learn menekankan bahwa mengevaluasi model pada data yang sama dengan data yang digunakan untuk belajar merupakan kesalahan metodologis dan dapat memberikan gambaran performa yang terlalu optimistis. (Scikit-Learn)

AV. Data Leakage — Pengenalan Awal

Berikan contoh sederhana:

```
Data test
↓
ikut digunakan untuk menentukan preprocessing/model
↓
kemudian dipakai lagi untuk menguji model
```

Ini bermasalah.

Prinsip yang perlu diingat:

Data pengujian harus tetap menjadi data yang tidak digunakan untuk belajar.

Untuk semester 1 cukup sebatas konsep.

AW. Mini Project Modul 8

“Profiling dan Prediksi Mahasiswa”

Mahasiswa membuat satu notebook dengan dua bagian.

Bagian A — Classification

Pertanyaan:

Dapatkah kita memprediksi status mahasiswa berdasarkan karakteristik akademiknya?

Features:

```
Jam_Belajar
Kehadiran
UTS
UAS
```

Target:

```
Status
```

Gunakan:

```
KNN
```

Output minimal:

```
Training data
Test data
Prediksi
Accuracy
```

Bagian B — Clustering

Pertanyaan:

Apakah mahasiswa dapat dikelompokkan berdasarkan karakteristik akademiknya?

Features:

```
Jam_Belajar
Kehadiran
Nilai_Akhir
```

Gunakan:

```
K-Means
```

Output:

```
Cluster
Rata-rata setiap cluster
Visualisasi
Interpretasi
```

AX. Struktur Notebook

```
MODUL 8
DATA MINING

Nama:
NIM:
Kelas:

1. Tujuan Praktikum

2. Review Pandas & EDA

3. Konsep Data Mining

4. Supervised Learning
   4.1 Feature
   4.2 Target
   4.3 Classification

5. Train-Test Split

6. KNN
   6.1 Konsep
   6.2 Membuat Model
   6.3 Training
   6.4 Prediction
   6.5 Evaluation

7. Feature Scaling

8. Unsupervised Learning
   8.1 Clustering
   8.2 K-Means
   8.3 Centroid
   8.4 Cluster Interpretation

9. Visualisasi Cluster

10. Elbow Method

11. Data Challenge

12. Mini Project

13. Interpretasi

14. Keterbatasan Model

15. Refleksi
```

AY. Rubrik Penilaian

Komponen	Bobot
Konsep Data Mining	10%
Feature & Target	10%
Train-Test Split	10%
KNN	15%
Evaluasi	10%
Scaling	10%
K-Means	15%
Visualisasi	10%
Interpretasi	5%
Refleksi/keterbatasan	5%
Total	**100%**

Saya sengaja memberikan **10% untuk keterbatasan model**.

Mahasiswa semester 1 harus mulai dibiasakan mengatakan:

“Model saya menghasilkan akurasi 80% pada dataset pengujian.”

bukan:

“Model saya benar 80% dan pasti bisa memprediksi mahasiswa.”

Perbedaan kalimat itu kecil, tetapi secara ilmiah sangat besar.

AZ. Penggunaan AI pada Modul 8

Modul ini justru menjadi tempat yang bagus untuk mengajarkan **cara menggunakan AI secara benar dalam machine learning**.

Prompt yang baik:

Jelaskan konsep KNN menggunakan analogi mahasiswa di kelas. Jangan berikan kode.

Kemudian:

Berikut kode KNN saya. Jangan perbaiki. Jelaskan setiap bagian kode agar saya dapat menjelaskannya saat praktikum.

Atau:

Saya mendapatkan accuracy 0.67. Berikan kemungkinan penyebabnya, tetapi jangan menyimpulkan bahwa model saya buruk sebelum melihat ukuran dataset dan pembagian train-test.

Lebih lanjut:

Buatkan lima pertanyaan kritis yang harus saya jawab sebelum menyimpulkan bahwa model KNN saya bagus.

Ini sejalan dengan Think Python edisi ketiga yang memang memasukkan penggunaan LLM sebagai bagian dari pembelajaran programming, termasuk prompting, testing, dan debugging.

Aturan praktis mahasiswa:

```
AI boleh membantu:
✓ menjelaskan
✓ memberi pertanyaan
✓ mencari kesalahan
✓ membantu debugging
✓ membandingkan metode

AI tidak boleh menggantikan:
✗ memahami algoritma
✗ menjalankan eksperimen
✗ membaca output
✗ menulis interpretasi tanpa verifikasi
```

BA. Bacaan dari Dua Buku

Di sini perlu sangat jelas.

Think Python, 3rd Edition dan **Python Crash Course, 3rd Edition** tetap menjadi buku fondasi, tetapi **tidak cukup sebagai buku utama Data Mining**.

Python Crash Course terutama memberikan fondasi programming dan kemudian membawa mahasiswa ke data visualization. Buku tersebut memang menyatakan bahwa setelah fondasi Python, pembaca diarahkan ke proyek data visualization dengan dataset dan library modern.

Think Python juga berfokus pada fundamental programming: syntax, semantics, variables, functions, data structures, files, databases, debugging, testing, serta penggunaan AI dalam proses belajar programming.

Jadi untuk Modul 8:

```
Think Python
      ↓
Algoritma + function + data structures

Python Crash Course
      ↓
Programming + data + visualization

      ↓

Scikit-learn
      ↓
Data Mining / Machine Learning
```

Untuk materi Data Mining, dokumentasi Scikit-learn menjadi sumber teknis tambahan. Dokumentasi resminya mencakup K-Means dan berbagai metode clustering, termasuk evaluasi clustering. (Scikit-Learn)

BB. Konsep yang Harus Dibawa ke Modul Berikutnya

Setelah Modul 8, mahasiswa harus sudah mengenal:

```
Dataset
   ↓
Feature
   ↓
Target
   ↓
Training
   ↓
Testing
   ↓
Model
   ↓
Prediction
   ↓
Evaluation
```

dan:

```
Dataset
   ↓
Feature
   ↓
Clustering
   ↓
Cluster
   ↓
Interpretation
```

Ini menjadi fondasi untuk modul berikutnya.

BC. Hubungan dengan Modul 9

Modul 9 adalah:

TEXT MINING

Mahasiswa akan menemukan bahwa data tidak selalu berupa angka.

Selama ini:

```
Jam_Belajar
Kehadiran
Nilai
Harga
Jumlah
```

Pada Modul 9:

```
"mahasiswa sangat puas dengan pembelajaran"
```

menjadi data.

Kemudian:

```
TEKS
  ↓
CLEANING
  ↓
TOKENIZATION
  ↓
REPRESENTATION
  ↓
FEATURE
  ↓
ANALYSIS
```

Bahkan classification yang baru dipelajari di Modul 8 dapat digunakan lagi:

```
Teks
  ↓
Feature
  ↓
Model
  ↓
Sentimen
```

Jadi Modul 8 bukan materi yang berdiri sendiri.

Ia menjadi jembatan:

```
Pandas & EDA
      ↓
DATA MINING
      ↓
CLASSIFICATION
      ↓
TEXT MINING
      ↓
MINI PROJECT
```

BD. Posisi Modul 8 dalam Keseluruhan Mata Kuliah

Sekarang roadmap mahasiswa sudah menjadi sangat jelas:

```
          MODUL 1
Algoritma & Problem Solving
          ↓
          MODUL 2
  Percabangan & Perulangan
          ↓
          MODUL 3
     List & Dictionary
          ↓
          MODUL 4
         Function
          ↓
          MODUL 5
      NumPy & Library
          ↓
          MODUL 6
        Visualisasi
          ↓
          MODUL 7
      Pandas & EDA
          ↓
          MODUL 8
      DATA MINING
          ↓
          MODUL 9
      TEXT MINING
          ↓
         MODUL 10
MINI PROJECT DATA SCIENCE
```

Dengan struktur ini, mahasiswa semester 1 tidak hanya selesai dengan kemampuan:

“Saya bisa membuat program Python.”

Mereka sudah memiliki gambaran yang jauh lebih besar:

“Saya bisa menggunakan Python untuk memahami data, mengolahnya, memvisualisasikannya, menemukan pola, dan mulai membuat prediksi.”

Dan itu merupakan fondasi yang sangat bagus untuk mata kuliah lanjutan.

MODUL 9

Text Mining dengan Python: Mengolah Teks Menjadi Data

Mata Kuliah	Praktikum Algoritma dan Pemrograman
Semester	1
Pertemuan	9 dari 10
Durasi	100 menit
Platform	Google Colab
Bahasa	Python 3
Library	Pandas, NumPy, Matplotlib, Scikit-learn
Level	Pemula → pengantar Text Mining/Natural Language Processing

Modul 9 memperluas pemahaman mahasiswa bahwa **data tidak selalu berupa angka**.

Pada Modul 5–8 mahasiswa banyak bekerja dengan:

```
70
85
90
3.75
95%
```

Sekarang mereka berhadapan dengan:

```
"Pembelajaran praktikum sangat menarik."

"Materi sulit tetapi dosennya membantu."

"Aplikasi sering mengalami error."
```

Pertanyaannya:

Bagaimana teks seperti itu dapat diproses oleh komputer?

Jawaban sederhananya:

```
TEKS
↓
CLEANING
↓
TOKENIZATION
↓
REPRESENTASI ANGKA
↓
```

```
FEATURE
   ↓
ANALISIS / MODEL
```

Di sinilah mahasiswa mulai melihat hubungan antara Python, data mining, dan kecerdasan buatan.

A. Posisi Modul 9

Roadmap praktikum sekarang:

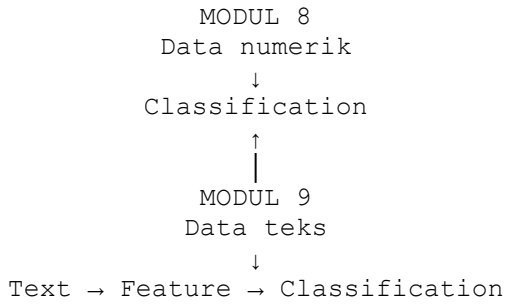
```
MODUL 1
Algoritma & Problem Solving
   ↓
MODUL 2
Percabangan & Perulangan
   ↓
MODUL 3
List & Dictionary
   ↓
MODUL 4
Function
   ↓
MODUL 5
NumPy & Library
   ↓
MODUL 6
Visualisasi
   ↓
MODUL 7
Pandas & EDA
   ↓
MODUL 8
Data Mining
   ↓
MODUL 9
TEXT MINING
   ↓
MODUL 10
Mini Project Data Science
```

Modul 8 memperkenalkan mahasiswa pada:

```
data numerik
   ↓
feature
   ↓
model
   ↓
classification
```

Modul 9 menunjukkan bahwa **teks juga dapat diubah menjadi feature**, kemudian digunakan dalam analisis atau classification.

Jadi hubungan kedua modul sangat kuat:



B. Tujuan Pembelajaran

Setelah menyelesaikan modul ini, mahasiswa mampu:

1. Menjelaskan pengertian sederhana text mining.
2. Menjelaskan mengapa komputer perlu mengubah teks menjadi angka.
3. Melakukan preprocessing teks sederhana.
4. Melakukan lowercase.
5. Menghapus tanda baca.
6. Melakukan tokenisasi sederhana.
7. Menghitung frekuensi kata.
8. Mengidentifikasi kata yang sering muncul.
9. Mengenal stopwords.
10. Mengenal representasi Bag of Words.
11. Menggunakan `CountVectorizer`.
12. Mengenal TF-IDF.
13. Menggunakan `TfidfVectorizer`.
14. Menggunakan hasil representasi teks sebagai feature.
15. Melakukan classification sederhana terhadap teks.
16. Menghubungkan Text Mining dengan konsep Data Mining pada Modul 8.
17. Memvisualisasikan frekuensi kata.
18. Menginterpretasikan hasil text mining secara kritis.

C. Apa Itu Text Mining?

Secara sederhana:

Text mining adalah proses memperoleh informasi atau pola dari kumpulan data berbentuk teks dengan bantuan metode komputasional.

Contoh data:

```
"Pelayanan sangat cepat."

"Pelayanan lambat dan mengecewakan."

"Produknya bagus."

"Produk rusak dan pelayanan buruk."
```

Manusia dapat memahami kalimat tersebut.

Komputer tidak memahami maknanya seperti manusia.

Komputer membutuhkan representasi numerik.

Maka:

```
Teks
  ↓
Angka
  ↓
Algoritma
  ↓
Pola
```

Ini adalah ide fundamental yang harus dipahami mahasiswa.

D. Mengapa Teks Harus Diubah Menjadi Angka?

Misalnya:

```
"produk bagus"
```

KNN atau algoritma machine learning tidak dapat langsung menghitung jarak dari string tersebut.

Kita perlu mengubahnya menjadi bentuk numerik:

```
produk → 1
bagus  → 1
```

atau dalam representasi tertentu:

```
[1, 1, 0, 0, ...]
```

Kemudian algoritma dapat memprosesnya.

Jadi:

```
Bahasa manusia
      ↓
Representasi numerik
      ↓
Machine Learning
```

E. Dataset Praktikum

Agar mahasiswa tidak sekadar menghafal konsep, gunakan contoh yang dekat dengan kehidupan mahasiswa.

Kita buat dataset ulasan mahasiswa terhadap pembelajaran:

```
import pandas as pd

data = {
    "teks": [
        "Pembelajaran sangat menarik",
        "Materi mudah dipahami",
        "Dosen menjelaskan dengan jelas",
        "Praktikum sangat menyenangkan",
        "Materi terlalu sulit",
        "Penjelasan kurang jelas",
        "Praktikum membosankan",
        "Tugas terlalu banyak",
        "Pembelajaran sangat bagus",
        "Materi sangat membantu"
    ],

    "sentimen": [
        "positif",
        "positif",
        "positif",
        "positif",
        "negatif",
        "negatif",
        "negatif",
        "negatif",
        "positif",
        "positif"
    ]
}

df = pd.DataFrame(data)

df
```

Dataset ini kecil.

Sengaja.

Tujuannya bukan membuat model produksi.

Tujuannya agar mahasiswa memahami seluruh proses dari awal sampai akhir.

F. Mulai dari Algoritma Manual

Sebelum menggunakan library, ambil:

```
teks = "Pembelajaran sangat menarik"
```

Ubah menjadi huruf kecil:

```
teks.lower()
```

Hasil:

```
pembelajaran sangat menarik
```

Kemudian:

```
teks.lower().split()
```

Hasil:

```
[
    "pembelajaran",
    "sangat",
    "menarik"
]
```

Ini penting.

Mahasiswa melihat bahwa tokenisasi sederhana sebenarnya bisa dilakukan menggunakan Python dasar.

Hubungkan dengan Modul 3:

```
string
 ↓
split()
 ↓
list
```

Jadi Text Mining bukan dunia yang sepenuhnya baru.

G. Tokenisasi

Tokenisasi adalah proses memecah teks menjadi bagian-bagian kecil yang disebut token.

Contoh:

```
"Python sangat menarik"
```

menjadi:

```
Python
sangat
menarik
```

Dalam Python:

```
teks = "Python sangat menarik"

token = teks.lower().split()

print(token)
```

Hasil:

```
['python', 'sangat', 'menarik']
```

Mahasiswa perlu memahami bahwa metode `split()` adalah tokenisasi **sederhana**, bukan solusi NLP yang sempurna untuk semua bahasa.

H. Masalah Tanda Baca

Misalnya:

```
teks = "Python sangat menarik!"
```

Jika:

```
print(teks.lower().split())
```

hasilnya:

```
['python', 'sangat', 'menarik!']
```

Perhatikan:

```
menarik
```

dan:

```
menarik!
```

dianggap berbeda.

Padahal manusia memahami keduanya sebagai kata yang sama.

Maka diperlukan preprocessing.

I. Menghapus Tanda Baca

Gunakan:

```
import string

teks = "Python sangat menarik!"

teks_bersih = teks.lower().translate(
    str.maketrans("", "", string.punctuation)
)

print(teks_bersih)
```

Hasil:

```
python sangat menarik
```

Kemudian:

```
print(teks_bersih.split())
```

J. Function Cleaning

Hubungkan dengan Modul 4.

```
import string

def bersihkan_teks(teks):
    teks = teks.lower()

    teks = teks.translate(
        str.maketrans("", "", string.punctuation)
    )

    return teks
```

Uji:

```
teks = "Pembelajaran Sangat Menarik!"

print(bersihkan_teks(teks))
```

Hasil:

```
pembelajaran sangat menarik
```

Ini penting secara pedagogis:

```
MODUL 4
Function
   ↓
MODUL 9
Text preprocessing
```

K. Memproses Seluruh Dataset

Sekarang:

```
df["teks_bersih"] = df["teks"].apply(
    bersihkan_teks
)

df
```

Perhatikan lagi:

```
Function
↓
apply()
↓
seluruh kolom teks
```

Konsep ini sudah dipelajari di Modul 7.

L. Menghitung Frekuensi Kata

Sekarang kita ingin mengetahui:

Kata apa yang paling sering muncul?

Gunakan Python dasar.

```
semua_kata = []
```

```
for teks in df["teks_bersih"]:
    semua_kata.extend(teks.split())

print(semua_kata)
```

Sekarang gunakan `Counter`:

```
from collections import Counter

frekuensi = Counter(semua_kata)

print(frekuensi)
```

Kata paling sering:

```
frekuensi.most_common(10)
```

Mahasiswa sekarang melihat hubungan:

```
                Modul 3
Dictionary / struktur data
                  ↓
                Modul 9
                Counter
                  ↓
          Frekuensi kata
```

M. Apa yang Kita Temukan?

Misalnya hasil:

```
pembelajaran    2
sangat          3
materi          3
praktikum       2
...
```

Kita sudah melakukan bentuk paling sederhana dari text mining:

```
Teks
 ↓
Token
 ↓
Frekuensi
 ↓
Informasi
```

Belum menggunakan machine learning.

Dan ini penting.

Text mining tidak selalu berarti machine learning.

Menghitung frekuensi kata saja sudah merupakan bentuk eksplorasi teks.

N. Stopwords

Dalam bahasa alami terdapat kata-kata yang sering muncul tetapi sering kali kurang informatif untuk tugas tertentu.

Contoh:

```
yang
dan
di
ke
dari
ini
itu
```

Kata-kata seperti ini sering disebut **stopwords**.

Tetapi jangan mengajarkan:

“Stopword selalu harus dihapus.”

Tidak.

Penghapusan stopword tergantung tujuan analisis.

Misalnya untuk analisis sentimen, kata tertentu yang tampak umum dapat tetap memiliki fungsi penting dalam konteks tertentu.

Jadi prinsipnya:

Preprocessing harus mengikuti tujuan analisis.

O. Mengenal Stopwords Bahasa Indonesia

Untuk praktikum sederhana, kita dapat membuat daftar kecil sendiri:

```
stopwords = {
    "yang",
    "dan",
    "di",
```

```
    "ke",
    "dari",
    "ini",
    "itu"
}
```

Function:

```
def hapus_stopword(tokens):
    return [
        kata
        for kata in tokens
        if kata not in stopwords
    ]
```

Contoh:

```
tokens = [
    "pembelajaran",
    "sangat",
    "menarik",
    "dan",
    "mudah"
]

print(hapus_stopword(tokens))
```

Hasil:

```
['pembelajaran', 'sangat', 'menarik', 'mudah']
```

P. Bag of Words

Sekarang kita masuk ke konsep penting.

Misalkan ada tiga dokumen:

```
D1 = "python menarik"
D2 = "python mudah"
D3 = "data menarik"
```

Vocabulary:

```
python
menarik
mudah
data
```

Kemudian representasi:

```
        python menarik mudah data

D1         1       1      0     0
D2         1       0      1     0
D3         0       1      0     1
```

Inilah gagasan sederhana **Bag of Words**.

Yang diperhatikan adalah kemunculan kata, bukan urutan kata.

Q. Mengapa Disebut “Bag”?

Analogi sederhana:

Bayangkan kita memasukkan kata-kata ke dalam kantong.

```
"python sangat menarik"
```

menjadi:

```
python
sangat
menarik
```

Urutan tidak menjadi fokus utama.

Yang diperhatikan:

kata apa yang ada dan berapa kali muncul?

Ini adalah salah satu keterbatasan Bag of Words.

R. Menggunakan CountVectorizer

Sekarang kita tidak perlu membuat tabel tersebut secara manual.

Gunakan Scikit-learn:

```
from sklearn.feature_extraction.text import CountVectorizer
```

Buat:

```
vectorizer = CountVectorizer()

X = vectorizer.fit_transform(
    df["teks_bersih"]
)
```

Lihat vocabulary:

```
print(vectorizer.get_feature_names_out())
```

Kemudian:

```
print(X.toarray())
```

Sekarang mahasiswa dapat melihat teks berubah menjadi matriks angka.

S. Inilah Titik Penting Modul

Perhatikan:

```
Teks
↓
CountVectorizer
↓
Matriks angka
↓
Machine Learning
```

Mahasiswa harus benar-benar memahami ini.

Karena pada Modul 8:

```
Data numerik
↓
X
↓
KNN
```

Sekarang:

```
Data teks
↓
Vectorizer
↓
X
↓
KNN / classifier
```

Jadi sebenarnya kita hanya mengganti cara membuat `X`.

T. Melihat Bentuk Matriks

```
print(X.shape)
```

Misalnya:

```
(10, 18)
```

Artinya:

```
10 dokumen
18 kata unik
```

Jelaskan:

```
baris
↓
dokumen

kolom
↓
feature/kata
```

Ini menghubungkan langsung dengan:

```
NumPy shape
Pandas DataFrame
Machine Learning X
```

U. Mengubah ke DataFrame

Agar mudah dibaca:

```
bow = pd.DataFrame(
    X.toarray(),
    columns=vectorizer.get_feature_names_out()
)

bow
```

Sekarang mahasiswa dapat melihat representasi Bag of Words sebagai tabel.

Ini adalah momen yang sangat bagus untuk memperlihatkan:

```
Teks
 ↓
Pandas
 ↓
Vectorization
 ↓
Numerical features
```

V. Keterbatasan Bag of Words

Misalnya:

```
"Saya tidak suka Python"

"Saya suka Python"
```

Bag of Words bisa saja melihat kata:

```
saya
tidak
suka
python
```

Masalahnya:

Kata “tidak” memiliki peran penting dalam makna kalimat, tetapi representasi sederhana Bag of Words tidak memahami konteks seperti manusia.

Jadi:

```
Bag of Words
≠
Pemahaman bahasa
```

Ini penting agar mahasiswa tidak menganggap text mining sederhana sebagai “komputer sudah memahami bahasa”.

W. TF-IDF

Sekarang diperkenalkan metode yang lebih baik untuk mengukur pentingnya kata dalam dokumen.

TF-IDF adalah singkatan dari:

Term Frequency–Inverse Document Frequency.

Tidak perlu mengajarkan rumus panjang pada semester 1.

Gunakan intuisi:

Kata yang sering muncul dalam satu dokumen tetapi tidak muncul di banyak dokumen lain cenderung lebih informatif.

Misalnya:

```
"python"
```

muncul hampir di semua dokumen.

Maka kata tersebut mungkin kurang membedakan dokumen.

Sebaliknya:

```
"algoritma"
```

hanya muncul pada beberapa dokumen.

Kata tersebut bisa lebih informatif.

X. Menggunakan TfidfVectorizer

```
from sklearn.feature_extraction.text import TfidfVectorizer
```

Kemudian:

```
tfidf = TfidfVectorizer()

X_tfidf = tfidf.fit_transform(
    df["teks_bersih"]
)
```

Lihat vocabulary:

```
print(
    tfidf.get_feature_names_out()
)
```

Dan:

```
print(X_tfidf.toarray())
```

Mahasiswa tidak perlu menghitung TF-IDF secara manual.

Tetapi mereka harus memahami:

```
TF-IDF
  ↓
memberi bobot
  ↓
kata penting
  ↓
feature numerik
```

Y. Bag of Words vs TF-IDF

Aspek	Bag of Words	TF-IDF
Representasi	Frekuensi	Bobot
Memperhatikan frekuensi	Ya	Ya
Mempertimbangkan frekuensi antar-dokumen	Tidak	Ya
Mudah dipahami	Sangat	Sedang
Cocok untuk pengantar	Ya	Ya

Mahasiswa tidak perlu menghafal formula TF-IDF.

Yang penting memahami alasan menggunakannya.

Z. Text Classification

Sekarang kita menghubungkan Modul 9 dengan Modul 8.

Dataset memiliki:

```
teks
sentimen
```

Maka:

```
X = df["teks_bersih"]
y = df["sentimen"]
```

Ubah teks menjadi angka:

```
vectorizer = TfidfVectorizer()

X_tfidf = vectorizer.fit_transform(X)
```

Sekarang:

```
TEKS
  ↓
TF-IDF
  ↓
  X
```

Target:

```
  y
  ↓
positif / negatif
```

Ini sudah sama persis dengan pola Modul 8.

AA. Train-Test Split

```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(
    X_tfidf,
    y,
    test_size=0.3,
    random_state=42
)
```

Kemudian gunakan KNN:

```
from sklearn.neighbors import KNeighborsClassifier

model = KNeighborsClassifier(
    n_neighbors=3
)
```

Training:

```
model.fit(
    X_train,
    y_train
)
```

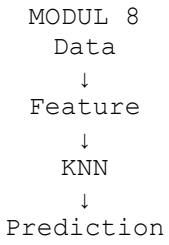
Prediction:

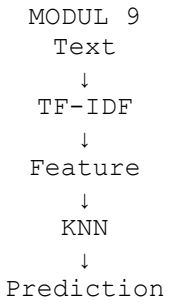
```
y_pred = model.predict(X_test)
```

Evaluasi:

```
from sklearn.metrics import accuracy_score

print(
    accuracy_score(y_test, y_pred)
)
```

Mahasiswa sekarang melihat:





Konsepnya sama.

AB. Mengapa Dataset Ini Tidak Cocok untuk Kesimpulan Serius?

Ini harus dijelaskan.

Dataset kita hanya:

```
10 dokumen
```

Itu sangat kecil.

Maka jika model menghasilkan:

```
100% accuracy
```

jangan mengatakan:

“Model sangat akurat.”

Lebih tepat:

“Pada dataset kecil ini dan pembagian data pengujian tertentu, model menghasilkan akurasi 100%.”

Perbedaan ini penting.

Mahasiswa harus mulai memahami bahwa **angka evaluasi selalu mempunyai konteks**.

AC. Mini Dataset Lebih Besar

Untuk latihan berikutnya, gunakan dataset sintetis.

```
data = {
    "teks": [
        "pelayanan sangat baik",
        "produk sangat bagus",
        "saya sangat puas",
        "pengiriman cepat dan bagus",
        "pelayanan ramah",
        "produk mengecewakan",
        "pelayanan sangat buruk",
        "pengiriman sangat lambat",
        "saya tidak puas",
        "produk rusak",
        "pelayanan mengecewakan",
        "produk sangat buruk",
        "sangat membantu",
        "sangat menyenangkan",
        "kualitas sangat baik",
        "kualitas buruk sekali",
        "sangat puas dengan produk",
        "produk tidak bagus",
        "pelayanan memuaskan",
        "pengiriman mengecewakan"
    ],

    "sentimen": [
        "positif", "positif", "positif",
        "positif", "positif", "negatif",
        "negatif", "negatif", "negatif",
        "negatif", "negatif", "negatif",
        "positif", "positif", "positif",
        "negatif", "positif", "negatif",
        "positif", "negatif"
    ]
}

df = pd.DataFrame(data)
```

Mahasiswa melakukan seluruh proses dari awal.

AD. Challenge 1 — Frekuensi Kata

Mahasiswa harus membuat program yang menghasilkan:

```
10 kata paling sering
```

Gunakan:

```
from collections import Counter

semua_kata = []

for teks in df["teks"]:
    teks = bersihkan_teks(teks)
```

```
    semua_kata.extend(teks.split())

frekuensi = Counter(semua_kata)

frekuensi.most_common(10)
```

Kemudian pertanyaan:

Apakah kata yang paling sering muncul otomatis merupakan kata yang paling penting?

Jawaban:

Tidak.

Ini menjadi alasan mengapa kita mengenal TF-IDF.

AE. Challenge 2 — Kata Positif dan Negatif

Pisahkan:

```
positif = df[
    df["sentimen"] == "positif"
]

negatif = df[
    df["sentimen"] == "negatif"
]
```

Kemudian hitung frekuensi kata masing-masing.

Mahasiswa mencari:

Kata apa yang paling sering muncul dalam ulasan positif?

dan:

Kata apa yang paling sering muncul dalam ulasan negatif?

AF. Visualisasi Frekuensi Kata

Gunakan Pandas dan Matplotlib.

```
frekuensi_df = pd.DataFrame(
```

```
    frekuensi.most_common(10),
    columns=["Kata", "Frekuensi"]
)
```

Kemudian:

```
frekuensi_df.plot(
    kind="bar",
    x="Kata",
    y="Frekuensi",
    legend=False
)

plt.title("10 Kata Paling Sering")
plt.ylabel("Frekuensi")

plt.show()
```

Ini menggabungkan:

```
Modul 3
struktur data
     ↓
Modul 5
 NumPy
     ↓
Modul 6
visualisasi
     ↓
Modul 7
 Pandas
     ↓
Modul 9
Text Mining
```

AG. Challenge 3 — Klasifikasi Ulasan Baru

Buat:

```
teks_baru = [
    "pelayanan sangat bagus",
    "produk sangat buruk",
    "saya puas",
    "pengiriman lambat"
]
```

Model dapat digunakan untuk memprediksi:

```
X_baru = tfidf.transform(teks_baru)
```

```
prediksi = model.predict(X_baru)

print(prediksi)
```

Mahasiswa mendapatkan:

```
positif
negatif
positif
negatif
```

atau hasil lain tergantung dataset dan model.

Pertanyaan:

Apakah prediksi tersebut selalu benar?

Tidak.

AH. Challenge Logika

Jangan langsung menggunakan `TfidfVectorizer`.

Tanyakan:

Jika Anda hanya boleh menggunakan Python dasar, bagaimana Anda akan menentukan apakah sebuah kalimat memiliki lebih banyak kata positif atau negatif?

Misalnya:

```
kata_positif = {
    "bagus",
    "baik",
    "puas",
    "menyenangkan",
    "memuaskan"
}

kata_negatif = {
    "buruk",
    "lambat",
    "rusak",
    "kecewa",
    "mengecewakan"
}
```

Buat function:

```
def klasifikasi_manual(teks):
    ...
```

Konsep algoritmanya:

```
                TEKS
                 ↓
               TOKEN
                 ↓
        HITUNG KATA POSITIF
                 ↓
        HITUNG KATA NEGATIF
                 ↓
            BANDINGKAN
                 ↓
               HASIL
```

Ini sangat bagus untuk mempertahankan fokus mata kuliah pada **algoritma**.

AI. Mengapa Pendekatan Manual Penting?

Karena mahasiswa akan melihat:

```
Manual
↓
aturan sederhana
```

kemudian:

```
Machine Learning
↓
model belajar dari data
```

Perbedaannya menjadi jelas.

Contoh manual:

```
"bagus" → positif
"buruk" → negatif
```

Machine learning:

```
banyak contoh
      ↓
   feature
      ↓
   model
      ↓
```

```
pola statistik
      ↓
  prediksi
```

AJ. Challenge 4 — Apakah Urutan Kata Penting?

Berikan:

```
"tidak bagus"
```

```
"bagus"
```

Tanyakan:

Apakah kedua kalimat memiliki makna yang sama?

Tidak.

Kemudian jelaskan bahwa representasi sederhana seperti Bag of Words dapat mengalami kesulitan memahami hubungan antar kata.

Ini membuka pintu menuju konsep NLP yang lebih maju.

Tidak perlu mengajarkan transformer atau LLM pada modul ini.

Cukup tanamkan kesadaran:

Representasi data menentukan apa yang dapat dipelajari oleh algoritma.

AK. Mini Project Modul 9

“Analisis Sentimen Ulasan Mahasiswa”

Mahasiswa menggunakan dataset ulasan:

```
teks
sentimen
```

Tahap 1 — Cleaning

Buat:

```
teks_bersih
```

Tahap 2 — Eksplorasi

Cari:

```
jumlah dokumen
jumlah positif
jumlah negatif
10 kata paling sering
```

Tahap 3 — Representasi

Gunakan:

```
CountVectorizer
```

dan:

```
TfidfVectorizer
```

Tahap 4 — Classification

Gunakan:

```
KNN
```

Tahap 5 — Evaluasi

Hitung:

```
accuracy
```

Tahap 6 — Uji Kalimat Baru

Masukkan minimal lima kalimat baru.

Tahap 7 — Interpretasi

Mahasiswa menjawab:

1. Kata apa yang paling sering muncul?
2. Apakah kata yang paling sering selalu paling informatif?
3. Apakah model dapat membedakan positif dan negatif?
4. Apa kelemahan dataset?
5. Mengapa hasil model tidak boleh dianggap sempurna?

AL. Bonus — Membandingkan Bag of Words dan TF-IDF

Mahasiswa membuat dua model:

```
Model A
CountVectorizer
+
KNN
```

dan:

```
Model B
TfidfVectorizer
+
KNN
```

Kemudian:

```
Accuracy A = ...
Accuracy B = ...
```

Pertanyaan:

Apakah TF-IDF selalu menghasilkan akurasi lebih tinggi?

Jawaban:

Tidak.

Ini merupakan kesempatan bagus untuk membangun sikap eksperimental.

Tidak ada algoritma yang otomatis paling baik untuk semua dataset.

AM. Bonus — Naive Bayes

Untuk mahasiswa yang lebih cepat, kita dapat memperkenalkan satu classifier yang memang sering digunakan sebagai baseline untuk text classification:

```
from sklearn.naive_bayes import MultinomialNB
```

Kemudian:

```
model_nb = MultinomialNB()
```

```
model_nb.fit(
    X_train,
    y_train
)

prediksi = model_nb.predict(
    X_test
)
```

Evaluasi:

```
accuracy_score(
    y_test,
    prediksi
)
```

Tetapi ini **materi pengayaan**, bukan materi wajib.

Mahasiswa cukup memahami:

```
TF-IDF
↓
Classifier
↓
Prediction
```

Tidak perlu mempelajari teori probabilitas Naive Bayes secara mendalam pada semester 1.

AN. Debugging Challenge

Berikan:

```
vectorizer = TfidfVectorizer()

X_train = vectorizer.fit_transform(
    df_train["teks"]
)

X_test = vectorizer.fit_transform(
    df_test["teks"]
)
```

Tanyakan:

Apakah ini cara yang benar?

Tidak.

Seharusnya:

```
X_train = vectorizer.fit_transform(
    df_train["teks"]
)

X_test = vectorizer.transform(
    df_test["teks"]
)
```

Karena vocabulary dan parameter representasi dipelajari dari training data.

Ini menghubungkan kembali konsep:

```
TRAIN
  ↓
BELAJAR

TEST
  ↓
TRANSFORM
```

Sama seperti prinsip scaling pada Modul 8.

AO. Kesalahan Umum Mahasiswa

1. Menganggap teks langsung dapat digunakan KNN

Salah.

Harus:

```
Teks
  ↓
Vectorization
  ↓
Numerical features
  ↓
Model
```

2. Menganggap 100% accuracy berarti model sempurna

Salah.

3. Menghapus semua kata pendek

Tidak selalu benar.

4. Menghapus stopword tanpa mempertimbangkan tujuan

Tidak selalu benar.

5. Menggunakan seluruh dataset untuk training lalu menguji pada dataset yang sama

Ini membuat evaluasi tidak dapat dipercaya.

6. Menganggap model memahami bahasa seperti manusia

Tidak.

Model memproses representasi numerik berdasarkan pola data.

AP. Penggunaan AI pada Modul 9

Modul ini sangat cocok untuk memperkenalkan penggunaan AI secara kritis.

Mahasiswa boleh meminta AI:

Jelaskan mengapa teks harus diubah menjadi angka sebelum digunakan dalam machine learning.

atau:

Jelaskan perbedaan Bag of Words dan TF-IDF dengan contoh sederhana.

atau:

Berikan lima contoh kalimat yang dapat digunakan untuk menguji model klasifikasi sentimen saya.

Namun mahasiswa harus melakukan sendiri:

```
cleaning
   ↓
vectorization
   ↓
training
   ↓
testing
   ↓
evaluation
```

Kemudian AI dapat digunakan untuk:

“Saya memperoleh accuracy 70%. Berikan kemungkinan alasan mengapa hasil tersebut rendah, tetapi jangan menyatakan penyebab pasti.”

Ini melatih mahasiswa menggunakan AI sebagai **asisten berpikir**, bukan mesin pembuat jawaban.

AQ. Struktur Notebook Google Colab

```
MODUL 9
TEXT MINING

Nama:
NIM:
Kelas:

1. Tujuan Praktikum

2. Review Pandas dan Data Mining

3. Apa Itu Text Mining?

4. Teks sebagai Data

5. Text Cleaning
   5.1 Lowercase
   5.2 Punctuation
   5.3 Tokenization

6. Function untuk Cleaning

7. Word Frequency

8. Stopwords

9. Bag of Words

10. CountVectorizer

11. TF-IDF

12. TfidfVectorizer

13. Text Classification

14. Train-Test Split

15. KNN

16. Evaluasi

17. Visualisasi Frekuensi Kata

18. Data Challenge
```

19. Mini Project Sentiment Analysis

20. Interpretasi

21. Keterbatasan Model

22. Refleksi

AR. Rubrik Penilaian

Komponen	Bobot
Konsep Text Mining	10%
Text Cleaning	15%
Tokenisasi & Frekuensi	10%
Bag of Words	10%
TF-IDF	15%
Classification	15%
Evaluasi	10%
Visualisasi	5%
Interpretasi	5%
Tantangan algoritmik	5%
Total	**100%**

AS. Bacaan dari Dua Buku Utama

Di sini juga harus dibedakan antara **fondasi** dan **materi khusus**.

Python Crash Course menyediakan fondasi yang relevan untuk Modul 9 melalui:

- string;
- list;
- dictionary;
- function;
- data processing;
- visualisasi;
- pengenalan library.

Indeks buku menunjukkan pembahasan string, list, dictionary, functions, files, testing, serta proyek data visualization.

Think Python juga relevan untuk:

- strings;
- lists;

- dictionaries;
- functions;
- data structures;
- debugging;
- testing.

Buku tersebut secara keseluruhan memang diarahkan untuk membangun kemampuan programming fundamental sebelum masuk ke bidang yang lebih khusus.

Untuk **Text Mining**, kedua buku tersebut tidak cukup sebagai referensi teknis utama. Maka mahasiswa diarahkan pada dokumentasi Scikit-learn untuk:

```
CountVectorizer
TfidfVectorizer
classification
```

Ini konsisten dengan prinsip yang kita gunakan sejak Modul 5: **dua buku utama menjadi fondasi Python, sedangkan library-specific documentation digunakan ketika memasuki ekosistem data science.**

AT. Hubungan dengan Modul 8

Ini perlu ditampilkan di akhir modul agar mahasiswa melihat kesinambungannya.

Modul 8:

```
DATA NUMERIK
      ↓
   FEATURE
      ↓
     KNN
      ↓
CLASSIFICATION
```

Modul 9:

```
   DATA TEKS
      ↓
   CLEANING
      ↓
    TF-IDF
      ↓
   FEATURE
      ↓
     KNN
      ↓
CLASSIFICATION
```

Jadi sebenarnya mahasiswa tidak mempelajari algoritma baru dari nol.

Mereka belajar:

Bagaimana mengubah jenis data yang berbeda menjadi feature yang dapat diproses oleh algoritma.

Ini adalah salah satu konsep paling penting dalam data science.

AU. Jembatan ke Modul 10

Setelah Modul 9, mahasiswa sudah menyentuh empat jenis pekerjaan:

```
PROGRAMMING
      ↓
DATA PROCESSING
      ↓
DATA ANALYSIS
      ↓
DATA MINING
      ↓
TEXT MINING
```

Maka Modul 10 sebaiknya **tidak lagi memperkenalkan banyak konsep baru**.

Modul 10 harus menjadi integrasi:

MODUL 10 — MINI PROJECT DATA SCIENCE

Mahasiswa memilih satu masalah sederhana.

Misalnya:

```
DATA
  ↓
Pandas
  ↓
EDA
  ↓
Visualisasi
  ↓
Feature Engineering
  ↓
Model sederhana
  ↓
Evaluasi
```

↓
Kesimpulan

Atau untuk mahasiswa yang memilih teks:

DATA TEKS
↓
Cleaning
↓
TF-IDF
↓
Classification
↓
Evaluation
↓
Interpretation

Dengan demikian seluruh semester akhirnya menyatu:

MODUL 1
LOGIKA
↓
MODUL 2
KONTROL PROGRAM
↓
MODUL 3
STRUKTUR DATA
↓
MODUL 4
FUNCTION
↓
MODUL 5
NUMPY
↓
MODUL 6
VISUALISASI
↓
MODUL 7
PANDAS + EDA
↓
MODUL 8
DATA MINING
↓
MODUL 9
TEXT MINING
↓
MODUL 10
MINI PROJECT

Dengan desain seperti ini, **praktikum Algoritma dan Pemrograman semester 1 tidak berhenti pada kemampuan membuat program sederhana**, tetapi secara bertahap membawa mahasiswa dari logika pemrograman menuju cara berpikir komputasional dan data-driven. Itu yang menurut saya menjadi kekuatan utama struktur 10 pertemuan ini.

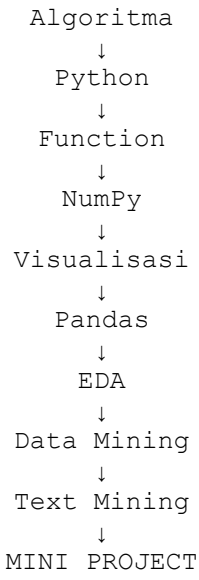
MODUL 10

Mini Project Data Science dengan Python

Mata Kuliah	Praktikum Algoritma dan Pemrograman
Semester	1
Pertemuan	10 dari 10
Durasi	100 menit
Platform	Google Colab
Bahasa	Python 3
Library	Pandas, NumPy, Matplotlib, Scikit-learn
Level	Pemula → integrasi Data Science

Modul 10 adalah **puncak dari seluruh rangkaian praktikum**.

Pada pertemuan ini mahasiswa tidak lagi terutama belajar sintaks baru. Mereka diminta menggabungkan kemampuan yang sudah dipelajari:



Tujuan akhirnya bukan menghasilkan model yang canggih.

Tujuannya adalah mahasiswa mampu menjawab satu pertanyaan menggunakan **proses komputasional yang lengkap**.

A. Filosofi Mini Project

Selama Modul 1–9 mahasiswa banyak mengerjakan instruksi seperti:

- Hitung rata-rata.

- Buat function.
- Filter data.
- Buat grafik.
- Buat model KNN.

Pada Modul 10 pendekatannya dibalik.

Mahasiswa diberikan:

sebuah masalah.

Kemudian mereka harus menentukan sendiri:

```
                    Apa datanya?
                         ↓
              Apa yang ingin diketahui?
                         ↓
          Bagaimana membersihkan data?
                         ↓
            Apa yang perlu dihitung?
                         ↓
       Apa yang perlu divisualisasikan?
                         ↓
              Apakah perlu model?
                         ↓
          Bagaimana mengevaluasinya?
                         ↓
Apa kesimpulannya?
```

Inilah pergeseran dari **belajar coding** menjadi **menggunakan coding untuk memecahkan masalah**.

B. Capaian Pembelajaran

Setelah menyelesaikan modul ini, mahasiswa mampu:

1. Merumuskan masalah sederhana berbasis data.
2. Menentukan dataset yang sesuai.
3. Membaca dataset menggunakan Pandas.
4. Melakukan pemeriksaan awal dataset.
5. Membersihkan data sederhana.
6. Melakukan EDA.
7. Membuat visualisasi yang relevan.
8. Membuat feature sederhana.
9. Menggunakan algoritma data mining sederhana bila diperlukan.
10. Mengevaluasi hasil model.
11. Menginterpretasikan hasil analisis.

12. Menjelaskan keterbatasan analisis.
13. Menyusun notebook Google Colab yang sistematis.
14. Menjelaskan hubungan antara algoritma dan hasil analisis.
15. Mempresentasikan hasil mini project secara singkat.

C. Skenario Praktikum 100 Menit

Karena ini pertemuan terakhir, waktunya tidak lagi digunakan untuk menjelaskan banyak teori baru.

Waktu	Kegiatan
0–10 menit	Penjelasan proyek dan kriteria
10–20 menit	Menentukan masalah dan dataset
20–35 menit	Data loading & inspection
35–50 menit	Cleaning & EDA
50–65 menit	Visualisasi & insight
65–80 menit	Analisis/model sederhana
80–90 menit	Evaluasi
90–97 menit	Penyusunan kesimpulan
97–100 menit	Presentasi singkat/exit ticket

Untuk proyek yang lebih besar, pengerjaan dapat dilanjutkan sebagai tugas di luar 100 menit.

D. Pilihan Mini Project

Saya menyarankan mahasiswa **tidak semuanya mengerjakan dataset yang sama**.

Berikan beberapa pilihan agar kelas lebih hidup.

Pilihan 1 — Analisis Nilai Mahasiswa

Pertanyaan:

Faktor apa yang tampak berhubungan dengan performa akademik mahasiswa?

Variabel:

```
Jam belajar
Kehadiran
UTS
UAS
Nilai akhir
```

Output:

```
EDA
Visualisasi
Korelasi
Clustering sederhana
```

Pilihan 2 — Analisis Penjualan

Pertanyaan:

Produk atau kategori apa yang paling berkontribusi terhadap penjualan?

Variabel:

```
Produk
Kategori
Harga
Jumlah
Tanggal
Total
```

Output:

```
Total penjualan
Produk terlaris
Tren penjualan
Visualisasi
Segmentasi sederhana
```

Pilihan 3 — Analisis Film

Pertanyaan:

Karakteristik film seperti apa yang memiliki rating tinggi?

Variabel:

```
Judul
Genre
Rating
Tahun
Durasi
```

Output:

```
EDA
Distribusi rating
Rating berdasarkan genre
Tren berdasarkan tahun
```

Pilihan 4 — Analisis Penggunaan Media Sosial

Pertanyaan:

Apakah terdapat pola hubungan antara penggunaan media sosial dan produktivitas?

Variabel:

```
Jam penggunaan
Platform
Jam belajar
Jam tidur
Produktivitas
```

Catatan penting:

Data harus diperlakukan sebagai dataset pembelajaran, bukan dasar untuk membuat klaim ilmiah tentang populasi.

Pilihan 5 — Analisis Ulasan

Pertanyaan:

Bagaimana kecenderungan sentimen pengguna terhadap suatu produk atau layanan?

Variabel:

```
ulasan
sentimen
```

Output:

```
Cleaning
Frekuensi kata
TF-IDF
Classification sederhana
```

Ini merupakan kelanjutan langsung Modul 9.

E. Struktur Mini Project

Semua proyek wajib memiliki struktur:

```
1. Problem
2. Dataset
```

```
3. Data Understanding
4. Data Cleaning
5. Exploratory Data Analysis
6. Visualization
7. Analysis / Modeling
8. Evaluation
9. Insight
10. Conclusion
11. Limitation
```

Dengan struktur ini mahasiswa tidak bisa hanya mengumpulkan kode.

F. Bagian 1 — Problem

Mahasiswa harus menulis:

Apa masalah yang ingin diselesaikan?

Contoh:

“Kami ingin mengetahui apakah terdapat hubungan antara jam belajar, kehadiran, dan nilai akhir mahasiswa.”

Kemudian:

“Kami ingin mengelompokkan mahasiswa berdasarkan pola belajar dan performa akademik.”

Perhatikan bahwa pertanyaan harus spesifik.

Jangan:

“Kami ingin menganalisis data mahasiswa.”

Itu terlalu umum.

G. Membuat Research Question Sederhana

Setiap kelompok wajib membuat minimal **dua pertanyaan**.

Contoh:

Pertanyaan 1

Apakah mahasiswa dengan kehadiran lebih tinggi cenderung memperoleh nilai lebih tinggi?

Pertanyaan 2

Apakah mahasiswa dapat dikelompokkan berdasarkan jam belajar, kehadiran, dan nilai?

Maka metode yang digunakan:

```
Pertanyaan 1
      ↓
EDA + correlation + visualization

Pertanyaan 2
      ↓
   K-Means
```

Ini mengajarkan mahasiswa bahwa:

Metode dipilih berdasarkan pertanyaan, bukan karena ingin menggunakan algoritma tertentu.

Ini prinsip yang sangat penting.

H. Bagian 2 — Data Understanding

Pertama:

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

Kemudian membaca data:

```
df = pd.read_csv("dataset.csv")
```

Periksa:

```
df.head()
```

Kemudian:

```
df.shape
df.info()
df.describe()
```

Dan:

```
df.isna().sum()
```

Mahasiswa harus menjelaskan **apa arti output**, bukan hanya menjalankan lima fungsi tersebut.

I. Data Dictionary

Mahasiswa membuat tabel sederhana:

Variabel	Tipe	Keterangan
Jam_Belajar	numerik	jam belajar per hari
Kehadiran	numerik	persentase kehadiran
UTS	numerik	nilai UTS
UAS	numerik	nilai UAS
Status	kategorikal	lulus/tidak

Ini latihan kecil tetapi penting.

Mahasiswa belajar bahwa sebelum menganalisis data, mereka harus mengetahui **arti setiap variabel**.

J. Bagian 3 — Data Cleaning

Mahasiswa memeriksa:

```
df.isna().sum()
```

Kemudian memeriksa duplikasi:

```
df.duplicated().sum()
```

Jika diperlukan:

```
df = df.drop_duplicates()
```

Untuk missing value:

```
df = df.dropna()
```

Tetapi mahasiswa harus menjelaskan:

Mengapa data tersebut dihapus?

Jangan membiasakan mahasiswa menggunakan:

```
dropna()
```

tanpa berpikir.

K. Validasi Data

Ini dapat menjadi bagian penting dari project.

Misalnya variabel:

```
Kehadiran
```

seharusnya berada pada:

```
0–100
```

Mahasiswa memeriksa:

```
df[
    (df["Kehadiran"] < 0) |
    (df["Kehadiran"] > 100)
]
```

Jika tidak ada hasil:

Tidak ditemukan nilai kehadiran di luar rentang yang diharapkan.

Jika ada:

Perlu diperiksa kembali data tersebut.

Ini merupakan pengenalan awal **data validation**.

L. Bagian 4 — Exploratory Data Analysis

Mahasiswa melakukan:

```
df.describe()
```

Kemudian minimal:

```
Mean
Median
Minimum
Maximum
```

untuk variabel numerik yang relevan.

Kemudian:

```
df["Kategori"].value_counts()
```

untuk variabel kategorikal.

M. Visualisasi

Setiap project minimal membuat **tiga visualisasi**, dan setiap visualisasi harus memiliki alasan.

Contoh:

Histogram

Untuk melihat distribusi:

```
plt.hist(df["Nilai_Akhir"], bins=10)

plt.xlabel("Nilai Akhir")
plt.ylabel("Jumlah")

plt.title("Distribusi Nilai Akhir")

plt.show()
```

Bar Chart

Untuk membandingkan kelompok:

```
df.groupby("Prodi")["Nilai_Akhir"].mean().plot(
    kind="bar"
)

plt.ylabel("Rata-rata Nilai")
plt.title("Rata-rata Nilai Berdasarkan Prodi")

plt.show()
```

Scatter Plot

Untuk melihat hubungan:

```
plt.scatter(
    df["Jam_Belajar"],
    df["Nilai_Akhir"]
)

plt.xlabel("Jam Belajar")
plt.ylabel("Nilai Akhir")

plt.title(
    "Jam Belajar vs Nilai Akhir"
)

plt.show()
```

N. Jangan Membuat Grafik Secara Asal

Mahasiswa harus menuliskan:

```
Tujuan visualisasi:
...

Pola yang terlihat:
...

Interpretasi:
...
```

Misalnya:

Visualisasi scatter plot digunakan untuk melihat apakah terdapat kecenderungan hubungan antara jam belajar dan nilai akhir.

Kemudian:

Data menunjukkan kecenderungan positif, tetapi hubungan tersebut tidak dapat langsung dianggap sebagai hubungan sebab-akibat.

Ini harus menjadi standar penulisan.

O. Feature Engineering

Sekarang mahasiswa mulai membuat variabel baru dari data yang sudah ada.

Misalnya:

```
df["Nilai_Akhir"] = (
    df["UTS"] * 0.4 +
    df["UAS"] * 0.6
)
```

Kemudian:

```
df["Status"] = np.where(
    df["Nilai_Akhir"] >= 60,
    "Lulus",
    "Tidak Lulus"
)
```

Ini disebut secara sederhana:

feature engineering

Mahasiswa tidak perlu memahami seluruh teori feature engineering.

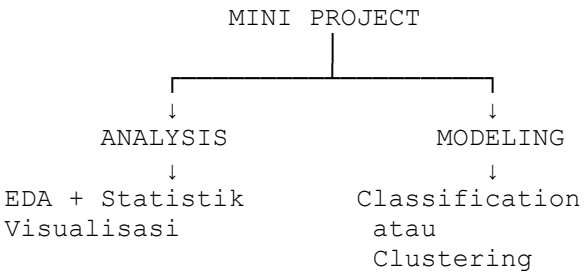
Cukup memahami:

Kita dapat membuat variabel baru dari variabel yang sudah ada agar data lebih berguna untuk analisis.

P. Bagian 5 — Pilih Jalur Analisis

Di sini project mulai bercabang.

Mahasiswa memilih salah satu:



Dengan demikian tidak semua mahasiswa dipaksa menggunakan machine learning.

Ini penting.

Q. Jalur A — Analisis Data

Untuk mahasiswa yang masih pemula, cukup melakukan:

```
Pandas
  +
NumPy
  +
Matplotlib
```

Contoh:

```
rata_rata = df.groupby(
    "Kategori"
)["Nilai_Akhir"].mean()

print(rata_rata)
```

Kemudian visualisasikan.

Mahasiswa memberikan kesimpulan.

R. Jalur B — Classification

Jika dataset memiliki target:

```
Lulus
Tidak Lulus
```

mahasiswa dapat menggunakan KNN.

Struktur:

```
          X
          ↓
train_test_split
          ↓
      scaling
          ↓
        KNN
          ↓
    prediction
          ↓
    accuracy
```

Kode dasar:

```
from sklearn.model_selection import train_test_split
```

```
from sklearn.preprocessing import StandardScaler
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score
```

Kemudian:

```
X = df[
    [
        "Jam_Belajar",
        "Kehadiran",
        "UTS",
        "UAS"
    ]
]

y = df["Status"]
```

Split:

```
X_train, X_test, y_train, y_test = train_test_split(
    X,
    y,
    test_size=0.2,
    random_state=42
)
```

Scaling:

```
scaler = StandardScaler()

X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
```

Model:

```
model = KNeighborsClassifier(
    n_neighbors=3
)

model.fit(
    X_train,
    y_train
)
```

Prediksi:

```
y_pred = model.predict(X_test)
```

Evaluasi:

```
accuracy = accuracy_score(
    y_test,
```

```
    y_pred
)

print("Accuracy:", accuracy)
```

S. Jalur C — Clustering

Jika dataset tidak memiliki target:

```
Jam_Belajar
Kehadiran
Nilai
```

mahasiswa dapat menggunakan K-Means.

```
from sklearn.cluster import KMeans

X = df[
    [
        "Jam_Belajar",
        "Kehadiran",
        "Nilai_Akhir"
    ]
]
```

Model:

```
model = KMeans(
    n_clusters=3,
    random_state=42,
    n_init="auto"
)

df["Cluster"] = model.fit_predict(X)
```

Analisis:

```
df.groupby("Cluster")[
    [
        "Jam_Belajar",
        "Kehadiran",
        "Nilai_Akhir"
    ]
].mean()
```

Kemudian visualisasi.

T. Tidak Semua Project Harus Menggunakan Machine Learning

Ini aturan yang saya sarankan dimasukkan secara eksplisit ke modul:

Project tidak dinilai berdasarkan seberapa canggih algoritma yang digunakan.

Project sederhana dengan:

```
Pandas
  +
 EDA
  +
Visualisasi
  +
Insight
```

dapat memperoleh nilai tinggi jika:

- pertanyaannya jelas;
- analisisnya benar;
- visualisasinya tepat;
- interpretasinya baik.

Sebaliknya, project yang menggunakan KNN tetapi mahasiswa tidak memahami hasilnya tidak seharusnya memperoleh nilai tinggi.

Ini menjaga agar mata kuliah tetap menjadi **Algoritma dan Pemrograman**, bukan sekadar lomba menggunakan library.

U. Bagian 6 — Evaluation

Jika menggunakan model, mahasiswa wajib menampilkan evaluasi.

Minimal:

```
Accuracy
```

Jika classification.

Jika clustering:

```
Cluster summary
```

dan untuk mahasiswa yang lebih maju:

```
Silhouette Score
```

Namun mahasiswa juga harus menjawab:

Apakah metrik tersebut cukup untuk menyatakan model bagus?

Jawaban:

Belum tentu.

V. Bagian 7 — Insight

Ini adalah bagian terpenting project.

Mahasiswa harus menghasilkan minimal:

5 insight.

Contoh:

1. Rata-rata nilai akhir kelompok A lebih tinggi daripada kelompok B.
2. Sebagian besar data berada pada rentang nilai 70–85.
3. Terdapat kecenderungan positif antara jam belajar dan nilai akhir.
4. Terdapat beberapa data dengan pola yang berbeda dari mayoritas.
5. Cluster tertentu memiliki rata-rata kehadiran dan nilai yang lebih tinggi.

Setiap insight harus dapat ditelusuri ke:

```
Data
↓
Perhitungan / grafik
↓
Interpretasi
```

W. Jangan Menulis Insight yang Tidak Didukung Data

Misalnya:

“Mahasiswa yang rajin pasti mendapat nilai tinggi.”

Tidak boleh.

Jika hanya terlihat korelasi:

“Pada dataset ini terdapat kecenderungan bahwa mahasiswa dengan jam belajar lebih tinggi memiliki nilai akhir lebih tinggi.”

Lebih tepat.

Kita ingin mahasiswa mulai terbiasa dengan bahasa ilmiah yang proporsional terhadap bukti.

X. Bagian 8 — Limitations

Setiap project wajib memiliki bagian:

Keterbatasan

Minimal tiga poin.

Contoh:

Dataset hanya terdiri dari 200 observasi.

Data merupakan data sintetis.

Variabel yang digunakan belum mencakup faktor lain yang mungkin memengaruhi nilai.

Model hanya diuji pada satu pembagian train-test.

Accuracy tidak cukup untuk menggambarkan seluruh performa model.

Ini akan membentuk kebiasaan ilmiah yang sangat baik.

Y. Bagian 9 — Kesimpulan

Kesimpulan harus menjawab pertanyaan awal.

Misalnya:

Pertanyaan:

Apakah jam belajar berhubungan dengan nilai?

Kesimpulan:

Berdasarkan eksplorasi terhadap dataset, terdapat kecenderungan hubungan positif antara jam belajar dan nilai akhir. Namun hasil tersebut tidak dapat digunakan untuk menyimpulkan hubungan sebab-akibat karena dataset terbatas dan penelitian ini tidak mengontrol faktor lain.

Ini jauh lebih baik daripada:

“Kesimpulannya jam belajar memengaruhi nilai.”

Z. Presentasi 3 Menit

Pada akhir praktikum, setiap kelompok mempresentasikan:

```
30 detik
Masalah

30 detik
Dataset

60 detik
Hasil utama

30 detik
Visualisasi

30 detik
Kesimpulan
```

Total:

3 menit per kelompok.

Jika kelas besar, presentasi dapat dilakukan dengan sistem *gallery walk* atau hanya beberapa kelompok dipilih secara acak.

AA. Struktur Notebook Final

Notebook Google Colab harus mempunyai struktur:

```
# MINI PROJECT DATA SCIENCE

## 1. Identitas

## 2. Problem Statement

## 3. Research Questions
```

```
## 4. Dataset

## 5. Data Dictionary

## 6. Import Library

## 7. Data Loading

## 8. Data Understanding

## 9. Data Cleaning

## 10. Exploratory Data Analysis

## 11. Visualization

## 12. Feature Engineering

## 13. Modeling
   jika diperlukan

## 14. Evaluation
   jika menggunakan model

## 15. Findings / Insight

## 16. Limitations

## 17. Conclusion

## 18. Reflection
```

AB. Contoh Project Lengkap

Agar mahasiswa memiliki contoh konkret, gunakan project:

“Apakah Kehadiran dan Jam Belajar Berkaitan dengan Nilai Mahasiswa?”

Dataset:

```
import pandas as pd
import numpy as np

np.random.seed(42)

n = 200

df = pd.DataFrame({
    "Jam_Belajar": np.random.randint(1, 10, n),
    "Kehadiran": np.random.randint(60, 101, n),
```

```
    "UTS": np.random.randint(40, 101, n),
    "UAS": np.random.randint(40, 101, n)
})
```

Nilai akhir:

```
df["Nilai_Akhir"] = (
    df["UTS"] * 0.4 +
    df["UAS"] * 0.6
)
```

Status:

```
df["Status"] = np.where(
    df["Nilai_Akhir"] >= 60,
    "Lulus",
    "Tidak Lulus"
)
```

AC. EDA Contoh

```
df.describe()
```

Korelasi:

```
df[
    [
        "Jam_Belajar",
        "Kehadiran",
        "UTS",
        "UAS",
        "Nilai_Akhir"
    ]
].corr()
```

AD. Visualisasi Contoh

```
plt.scatter(
    df["Jam_Belajar"],
    df["Nilai_Akhir"]
)

plt.xlabel("Jam Belajar")
plt.ylabel("Nilai Akhir")
plt.title(
    "Jam Belajar vs Nilai Akhir"
)

plt.show()
```

Kemudian:

```
plt.scatter(
    df["Kehadiran"],
    df["Nilai_Akhir"]
)

plt.xlabel("Kehadiran")
plt.ylabel("Nilai Akhir")
plt.title(
    "Kehadiran vs Nilai Akhir"
)

plt.show()
```

AE. Clustering Contoh

```
from sklearn.cluster import KMeans

X = df[
    [
        "Jam_Belajar",
        "Kehadiran",
        "Nilai_Akhir"
    ]
]

model = KMeans(
    n_clusters=3,
    random_state=42,
    n_init="auto"
)

df["Cluster"] = model.fit_predict(X)
```

Kemudian:

```
df.groupby("Cluster")[
    [
        "Jam_Belajar",
        "Kehadiran",
        "Nilai_Akhir"
    ]
].mean()
```

Mahasiswa kemudian memberikan interpretasi.

AF. Visualisasi Cluster

```
plt.scatter(
    df["Jam_Belajar"],
```

```
    df["Nilai_Akhir"],
    c=df["Cluster"]
)

plt.xlabel("Jam Belajar")
plt.ylabel("Nilai Akhir")
plt.title("Cluster Mahasiswa")

plt.show()
```

Pertanyaan:

Apakah cluster yang dihasilkan masuk akal?

Ini pertanyaan yang lebih penting daripada sekadar:

“Apakah kode berhasil?”

AG. Tantangan Tingkat Lanjut

Mahasiswa yang selesai lebih cepat dapat membandingkan:

Model 1

KNN dengan:

```
Jam_Belajar
Kehadiran
```

Model 2

KNN dengan:

```
Jam_Belajar
Kehadiran
UTS
UAS
```

Kemudian membandingkan hasil.

Pertanyaan:

Apakah menambah feature selalu meningkatkan performa model?

Jawaban:

Tidak selalu.

Ini menjadi pengantar awal tentang pemilihan feature.

AH. Tantangan Algoritma

Karena mata kuliah ini tetap **Algoritma dan Pemrograman**, mahasiswa harus menyelesaikan satu tugas tanpa library data science.

Misalnya:

Buat function untuk mencari tiga nilai tertinggi dari sebuah list tanpa menggunakan `sorted()`.

Contoh:

```
def tiga_tertinggi(data):
    ...
```

Atau:

Buat function untuk menghitung frekuensi setiap kata dari sebuah kalimat tanpa `Counter`.

Atau:

Buat function untuk menghitung rata-rata setiap kelompok berdasarkan dictionary.

Tujuannya:

```
Library
tidak boleh menghilangkan
kemampuan berpikir algoritmik.
```

AI. Challenge Terakhir — “Jangan Gunakan AI Dulu”

Saya menyarankan satu bagian khusus:

10 Menit Tanpa AI

Mahasiswa diberikan masalah:

Diberikan sebuah list nilai. Cari nilai tertinggi kedua tanpa menggunakan `sort()` atau `sorted()`.

Mereka harus:

```
1. Memahami masalah
2. Menyusun algoritma
3. Menulis kode
4. Menguji
```

Baru setelah itu mereka boleh bertanya kepada AI:

“Berikan kritik terhadap algoritma saya tanpa memberikan solusi.”

Ini sangat bagus untuk mengukur kemampuan berpikir mahasiswa sebenarnya.

AJ. Penggunaan AI dalam Mini Project

AI boleh digunakan untuk:

```
✓ menjelaskan error
✓ membantu memahami library
✓ mengusulkan pertanyaan analisis
✓ membantu membuat data sintetis
✓ memberikan kritik terhadap kode
✓ memberikan alternatif visualisasi
✓ menguji logika
```

Tetapi mahasiswa wajib dapat menjelaskan:

```
Kode saya melakukan apa?
↓
Mengapa saya menggunakan metode itu?
↓
Apa arti output?
↓
Apa keterbatasannya?
```

Jika mahasiswa tidak mampu menjelaskan kode yang dikumpulkan, nilai project dapat dikurangi.

AK. Rubrik Penilaian Mini Project

Komponen	Bobot
Problem formulation	10%
Dataset & data understanding	10%
Data cleaning	10%
EDA	15%
Visualisasi	10%
Algoritma/model	15%
Evaluasi	10%
Insight	10%
Conclusion & limitation	5%
Presentasi	5%
Total	**100%**

Yang penting:

Algoritma/model hanya 15%.

Ini disengaja.

Mahasiswa semester 1 tidak boleh berpikir:

“Semakin canggih algoritmanya, semakin bagus project.”

Yang dinilai adalah keseluruhan proses pemecahan masalah.

AL. Rubrik Kemampuan Pemrograman

Selain nilai project, dosen dapat memberikan indikator khusus:

Level	Kemampuan
1	Mampu menjalankan kode
2	Mampu memodifikasi kode
3	Mampu menulis kode sederhana
4	Mampu membuat function
5	Mampu memilih metode yang sesuai
6	Mampu menjelaskan hasil
7	Mampu menemukan dan memperbaiki kesalahan

Target akhir semester:

Minimal Level 4–5 untuk mayoritas mahasiswa.

Mahasiswa yang lebih kuat dapat mencapai Level 6–7.

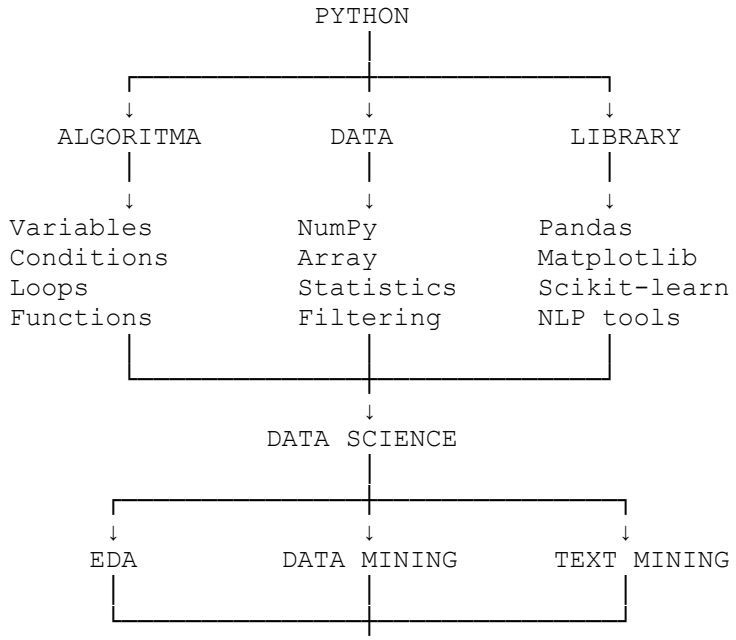
AM. Refleksi Akhir Semester

Mahasiswa menjawab secara tertulis:

1. Apa perbedaan antara algoritma dan library?
2. Apa perbedaan NumPy dan Pandas?
3. Apa fungsi visualisasi dalam analisis data?
4. Apa perbedaan EDA dan Data Mining?
5. Apa perbedaan classification dan clustering?
6. Mengapa data training dan testing harus dipisahkan?
7. Mengapa teks perlu diubah menjadi feature numerik?
8. Apa yang dimaksud dengan korelasi?
9. Mengapa korelasi tidak otomatis berarti sebab-akibat?
10. Bagian mana dari Python yang menurut Anda paling berguna untuk mata kuliah berikutnya?

AN. Peta Kompetensi Setelah 10 Pertemuan

Setelah seluruh praktikum selesai, mahasiswa seharusnya memiliki peta kemampuan:



```
          ↓
MINI PROJECT
```

AO. Hubungan dengan Dua Buku Pedoman

Pada tahap akhir ini semakin jelas fungsi dua buku utama.

Think Python, 3rd Edition menjadi fondasi berpikir pemrograman:

```
syntax
semantics
variables
functions
data structures
files
debugging
testing
```

Buku tersebut memang menempatkan konsep-konsep tersebut sebagai bagian fundamental Python dan juga membahas bagaimana AI coding assistants dapat digunakan dalam proses pemrograman.

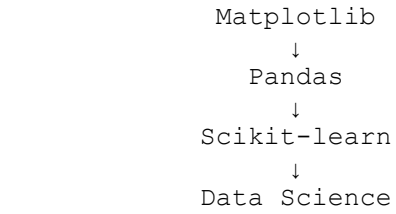
Python Crash Course, 3rd Edition menjadi fondasi praktis:

```
variables
lists
dictionaries
if
loops
functions
classes
files
testing
data visualization
```

Struktur buku tersebut memang bergerak dari Python fundamentals menuju proyek yang lebih aplikatif, termasuk data visualization.

Kemudian modul praktikum kita menambahkan lapisan:

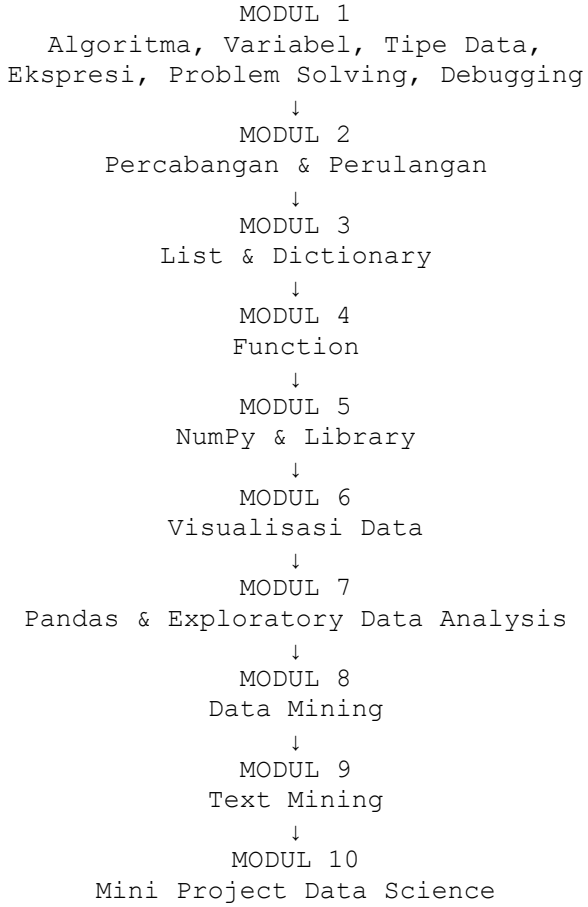
```
Think Python
     +
Python Crash Course
     ↓
Python Programming
     ↓
   NumPy
     ↓
```



Dengan demikian dua buku tersebut **tidak perlu dipaksa menjadi buku Data Science**. Mereka berfungsi sebagai fondasi programming yang kemudian diperluas melalui library dan dokumentasi terkait.

AP. Peta Besar Buku Modul Praktikum

Sekarang seluruh buku modul sudah dapat disusun sebagai:



Ada sebuah prinsip yang menurut saya perlu dipertahankan ketika nanti seluruh 10 modul ini kita rapikan menjadi buku:

Setiap kali library baru diperkenalkan, mahasiswa harus terlebih dahulu memahami masalah algoritmik yang diselesaikan library tersebut.

Dengan prinsip itu, mahasiswa tidak berubah menjadi “penghafal sintaks Pandas”, “pengguna KNN”, atau “penyalin kode ChatGPT”. Mereka tetap belajar **algoritma dan pemrograman**, tetapi diarahkan sejak semester pertama untuk melihat ke mana kemampuan Python tersebut digunakan dalam dunia data science.

Modul Praktikum

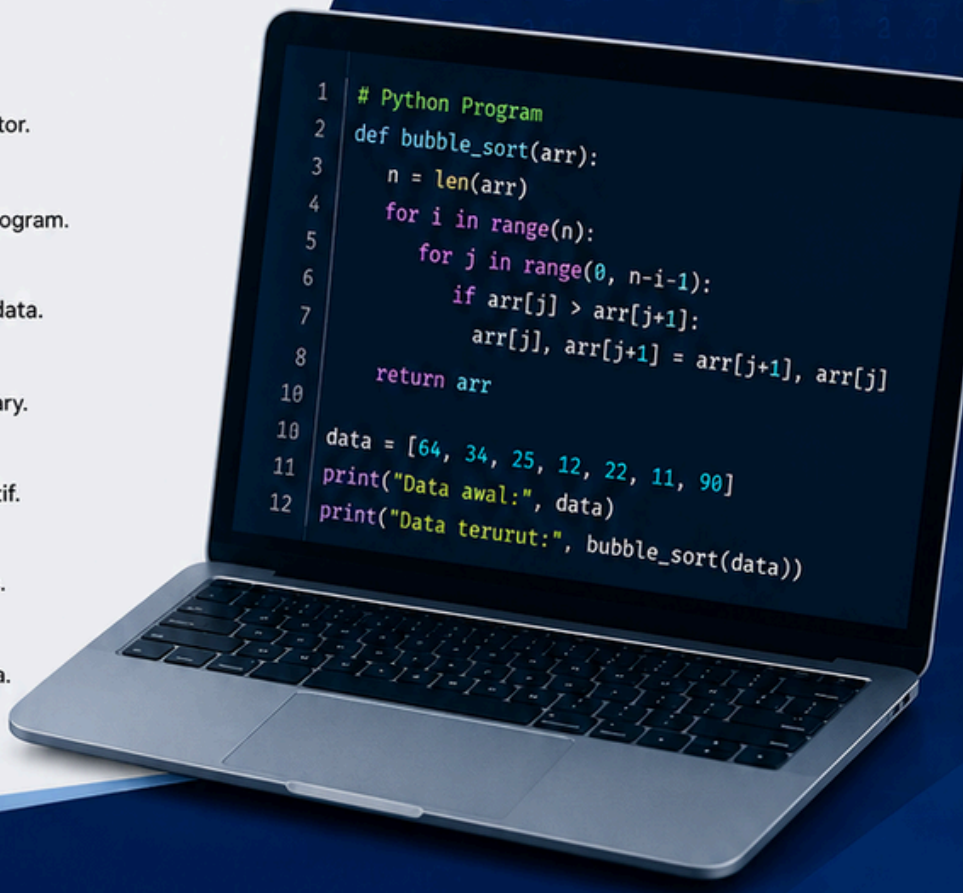
Algoritma dan Pemrograman

Modul praktikum ini disusun untuk memandu mahasiswa dalam memahami dasar-dasar algoritma dan pemrograman dengan **bahasa Python** menggunakan platform **Google Colab.** Materi disajikan secara bertahap, mulai dari konsep dasar hingga penerapan dalam pengolahan data, analisis, dan visualisasi. Setiap modul dilengkapi dengan penjelasan, contoh program, latihan, serta tantangan untuk melatih kemampuan **problem solving** secara sistematis.

Modul ini dirancang untuk mendukung pembelajaran praktis dan mandiri serta mempersiapkan mahasiswa menjadi calon **data scientist** yang kompeten.

YANG AKAN ANDA PELAJARI

- **Dasar pemrograman Python**
  Variabel, tipe data, ekspresi, dan operator.
- **Algoritma dan struktur kontrol**
  Percabangan, perulangan, dan logika program.
- **Struktur data**
  List, tuple, dictionary, dan manipulasi data.
- **Fungsi dan modul**
  Membuat fungsi dan menggunakan library.
- **Visualisasi data**
  Membuat grafik dan visualisasi informatif.
- **Pandas & EDA**
  Analisis data eksploratif dengan Pandas.
- **Data Mining**
  Penerapan teknik data mining sederhana.



Teknik Informatika
Fakultas Sains dan Teknologi
UIN Maulana Malik Ibrahim Malang

2026

ISBN 978-623-10-1234-5