

VERSI 2.3
NOVEMBER 2025



PEMROGRAMAN FUNGSIONAL

MODUL 5 - Functional Programming Implementation: Data Visualisation

DISUSUN OLEH:

FERA PUTRI AYU L., S.KOM., M.T.

ALFI AULIA AZZAHRA

RAHMATUN NIKMAH

**TIM LABORATORIUM INFORMATIKA
UNIVERSITAS MUHAMMADIYAH MALANG**

PENDAHULUAN

TUJUAN

Sub-CPMK 7: Mahasiswa mampu mendesain program dengan teknik yang tepat untuk menyelesaikan masalah dengan menggunakan paradigma pemrograman fungsional (P6)

TARGET MODUL

1. Mampu memahami dan mengimplementasikan beberapa python library untuk visualisasi data dengan memanfaatkan teknik pemrograman fungsional
2. Mampu menentukan Function yang tepat dalam prosesing data hingga visualisasi berdasarkan teknik pemrograman fungsional

PERSIAPAN

1. Komputer/Laptop
2. Python, Google Collab/VS Code/Jupyter Notebook
3. [Source Code Modul 5](#)

KEYWORDS

Library, Visualisasi, Matplotlib, Numpy, Pandas.

TABLE OF CONTENTS

PENDAHULUAN.....	2
TUJUAN.....	2
TARGET MODUL.....	2
PERSIAPAN.....	2
KEYWORDS.....	2
TABLE OF CONTENTS.....	2
LIBRARY PADA PYTHON UNTUK VISUALISASI DATA.....	4
1. MATPLOTLIB.....	5
1.1. Pengenalan Pyplot dan Dasar Plotting.....	6
1.2. Jenis-jenis Visualisasi Data dengan Matplotlib.....	8
a. Line Plot.....	8
b. Scatter Plot.....	9
c. Pie Chart.....	10
d. Bar Chart.....	13
e. Histogram.....	13
1.3. Subplot dalam matplotlib.....	14
2. NUMPY.....	16
a. Percobaan 1: Variasi visualisasi data.....	17



b. Percobaan 2: Multiple lines.....	18
Kenapa harus Numpy.....	19
3. PANDAS.....	20
3.1. Pandas Series.....	20
a. Percobaan 1: Membuat Series Menggunakan Array Sederhana.....	20
b. Percobaan 2: Membuat Series dari Dictionary.....	21
3.2. Dataframe Pandas.....	23
a. Percobaan 1: Membuat Dataframe dari Array Sederhana.....	23
b. Percobaan 2: Membuat Dataframe dari Dictionary Pandas.....	24
c. Percobaan 3: Membaca Data dengan Pandas.....	24
d. Percobaan 4: Operasi Dasar pada DataFrame.....	25
e. Percobaan 5: Operasi pada Kolom DataFrame.....	27
f. Numpy & Pandas.....	28
CODELAB.....	29
CODELAB 1.....	29
CODELAB 2.....	30
TUGAS.....	30
TUGAS 1.....	30
TUGAS 2.....	30
TUGAS 3.....	31
NIM GANJIL.....	31
NIM GENAP.....	31
RUBRIK PENILAIAN.....	32

Selamat datang di modul 5!! Tiba-tiba udah sampai modul 5 aja, setelah ini modul 6, lalu UAP, laluga kerasa ya xixixi.

Di modul 1-4 kemarin kita sudah belajar berbagai konsep berfikir secara fungsional mulai dari dasar functional programming, pure function, processing sequence data, hingga konsep first-class function, higher-order function, dll. Dari yang sudah kita pelajari kemarin membantu kita memahami bagaimana memproses data menjadi bersih, sistematis dan tentunya deklaratif sama seperti konsep paradigma fungsional.

Naahh, tanpa disadari sejak modul sebelumnya kita juga sudah menggunakan beberapa library python, seperti functools dan ada iteratools, dll. Artinya, kita dapat memproses data melalui fungsi-fungsi yang sifatnya serupa dengan konsep fungsional. Kalau kemarin kan kita sudah belajar memproses data ya bisa dibilang—mulai dari mengolah, memfilter, sampai memetakan data, sekarang lanjut ke tahapan selanjutnya ya. Data yang sudah diproses dapat diubah menjadi bentuk visual menggunakan library yang ada di python.

Library dalam python tuh banyak loh, diantaranya akan kita bahas pada modul ini, dan pastinya library ini juga bersifat fungsional juga mulai dari operasi yang tidak mengubah data asli, hingga cara pemanggilan yang deklaratif, menghasilkan output baru dan mudah dipahami. Nah supaya makin jelas, kita mulai dulu dengan mengenal beberapa library penting yang digunakan untuk visualisasi data di Python. Cekidot!

LIBRARY PADA PYTHON UNTUK VISUALISASI DATA

Pada modul ini, kita akan mempelajari dan menggunakan berbagai library Python yang berkaitan dengan statistika dan visualisasi data untuk mengolah serta menampilkan informasi secara efektif. Library ini banyak digunakan dalam analisis dan visualisasi data. Tujuannya adalah agar kita dapat mengolah data dan menyajikannya secara lengkap, menarik, dan mudah dipahami.

Python merupakan salah satu bahasa pemrograman yang sangat populer di dunia data karena memiliki banyak library yang kuat untuk keperluan statistik dan visualisasi. Kita tidak perlu menulis ulang kode yang rumit, cukup memanfaatkan fungsi-fungsi yang sudah tersedia di dalam library tersebut.

Secara umum, Python sudah memiliki banyak built-in library, yaitu library yang dapat langsung digunakan tanpa perlu instalasi tambahan setelah Python terpasang di perangkat kalian. Daftar lengkap built-in library dapat kalian baca [disini](#).

Dalam bidang data science, keberadaan berbagai library ini membuat pemrograman Python menjadi lebih sederhana dan nyaman bagi programmer, karena tidak perlu menulis ulang kode yang sama untuk program yang berbeda. Dengan demikian, kita dapat lebih fokus pada bagaimana data diolah dan divisualisasikan. Kehadiran library ini membuat proses analisis data menjadi lebih praktis dan efisien.

Pada modul ini, kita akan membahas beberapa *library* python yang berguna untuk pengolahan dan visualisasi data. Beberapa *library* yang akan kita pelajari antara lain:

- Matplotlib, untuk membuat berbagai jenis grafik dan visualisasi data dasar.



- Numpy, untuk mengolah data numerik dalam bentuk array dan melakukan perhitungan matematis
- Pandas, untuk memanipulasi dan menganalisis data dalam bentuk tabel (dataframe)

Ketiga library ini akan menjadi dasar penting dalam memahami bagaimana data dapat diolah dan divisualisasikan menggunakan paradigma pemrograman fungsional di Python. Lebih lengkapnya lagi kita bahas dibawah yaaa

1. MATPLOTLIB

Matplotlib adalah salah satu library Python yang digunakan untuk membuat berbagai macam grafik dan visualisasi data – seperti grafik garis, diagram batang, scatter plot, dan pie chart. Library ini membantu kita menampilkan data dalam bentuk gambar agar informasi lebih mudah dipahami dibandingkan hanya dalam bentuk angka.

Library ini gratis dan merupakan open source library. Karena fleksibilitasnya yang tinggi, Matplotlib menjadi salah satu library visualisasi data yang paling sering digunakan oleh data analyst dan data scientist. Secara umum, Matplotlib digunakan untuk:

- Membuat grafik dari data numerik (misalnya hasil perhitungan atau data statistik)
- Menyajikan data dengan gaya visual yang dapat disesuaikan (warna, garis, label, dll),
- Membantu memahami hubungan atau pola dari kumpulan data.

Kita dapat langsung mencoba melakukan import untuk mengecek apakah library matplotlib sudah tersedia dalam built-in package kita atau belum dengan cara menjalankan kode berikut:

```
import matplotlib
```

Jika **tidak** terjadi error, maka library siap digunakan. Namun, jika muncul error bahwa modul tidak ditemukan, kalian dapat menginstalnya melalui command prompt (CMD) atau terminal dengan perintah berikut:

```
pip install matplotlib
```

Jika kalian menggunakan Google Colab atau Jupyter Notebook (.ipynb), instalasi dapat dilakukan langsung dari cell menggunakan tanda seru (!) di awal perintah, seperti berikut:

```
!pip install matplotlib
```



Jalankan perintah tersebut di cell lalu klik play / tekan Shift + Enter untuk mengeksekusi. Setelah proses instalasi selesai, kalian bisa langsung mengimpor library Matplotlib tanpa perlu membuka CMD lagi.

1.1. Pengenalan Pyplot dan Dasar Plotting

Sebelum membuat berbagai jenis grafik, penting untuk memahami dulu dua istilah utama:

- Plotting adalah proses menggambar data ke dalam bentuk grafik atau diagram (misalnya garis, titik, atau batang).
- Visualisasi data adalah hasil akhir atau tujuan dari proses plotting – yaitu menyajikan data agar lebih mudah dipahami.

Dalam library Matplotlib, terdapat sebuah package bernama `pyplot` yang digunakan untuk proses plotting dan membuat berbagai jenis grafik – seperti grafik garis, titik, batang, hingga lingkaran (pie) dengan cara yang sederhana dan cepat. Cara mengaksesnya adalah sbb:

```
#1
#cara mengimport library dan package nya
import matplotlib.pyplot
#cara menggunakan
matplotlib.pyplot.show()
```

Untuk memudahkan penulisan kode, pyplot biasanya diimpor dengan nama alias, yaitu plt. Penggunaan alias ini bertujuan agar kita tidak perlu menulis nama package yang panjang setiap kali ingin menggunakan fungsinya.

Berikut adalah cara import package yang lebih pendek:

```
#2
from matplotlib import pyplot
pyplot.show()
```

Namun masih bisa kita persingkat lagi menggunakan alias seperti ini:

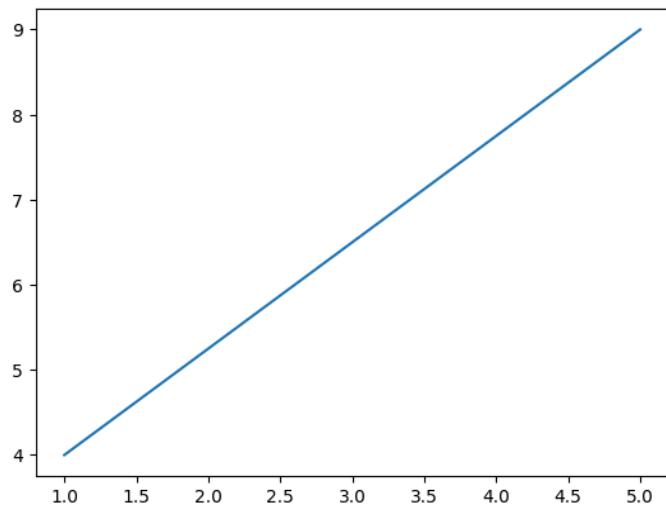
```
#3
from matplotlib import pyplot as plt
plt.show()
```

Bagian **as plt** pada import inilah yang disebut alias, yaitu cara memberi nama pendek pada library atau package saat diimpor agar lebih efisien. Alias ini bisa berupa nama apa pun (misalnya as m atau as grafik), tetapi secara umum – hampir semua programmer Python menggunakan plt untuk matplotlib.pyplot, supaya mudah dibaca dan dikenali.



Sekarang, kita mulai dengan bentuk visualisasi paling sederhana, yaitu grafik garis (line plot).

```
...
desc: menggunakan library matplotlib untuk membuat grafik garis sederhana
pre-cond: impor library 'matplotlib.pyplot' dan pastikan argumen yang
diberikan
berupa list atau array dengan ukuran yang sama, masing-masing berisi
koordinat x dan y.
post-condition: grafik garis yang menghubungkan kedua titik.
...
plt.plot([1, 5], [4, 9])
plt.show()
```



Pada kode di atas, setelah kita mengimport modul `pyplot`, kita menggunakan fungsi `plt.plot()` untuk membuat grafik garis.

Nilai `[1, 5]` merepresentasikan koordinat pada sumbu x, sedangkan `[4, 9]` adalah koordinat pada sumbu y. Kedua nilai tersebut kemudian dihubungkan menjadi sebuah garis lurus. Terakhir, fungsi `plt.show()` digunakan untuk menampilkan hasil grafik di layar. Jadi, hanya dengan beberapa baris kode ini, kita sudah bisa membuat visualisasi data sederhana menggunakan Matplotlib.

❤️ Latihan 1

Dari ketiga cara import yang sudah dicontohkan di materi, gunakan salah satunya untuk melengkapi kode berikut dan berikan alasan mengapa itu yang kalian pilih?

Modifikasi angka-angka yang ada dan amati hasilnya bagaimana hubungan antara angka tersebut terhadap output grafik dan pola koordinatnya!

```
import ....
....plot([1, 5], [4, 9]) #ganti dengan 4 NIM terakhir kalian
....show()
```

#percobaan1

1.2. Jenis-jenis Visualisasi Data dengan Matplotlib

Sekarang setelah kita tahu cara dasar menggunakan pyplot untuk membuat grafik sederhana, saatnya kita mengenal berbagai bentuk visualisasi data yang bisa dibuat menggunakan Matplotlib.

Setiap jenis grafik punya fungsi dan tujuan yang berbeda, tergantung pada data yang ingin kita tampilkan. Ada grafik yang cocok untuk menunjukkan perubahan nilai (grafik garis), ada yang lebih pas untuk melihat hubungan antar variabel (scatter plot), dan ada juga yang digunakan untuk membandingkan jumlah antar kategori (bar chart / pie chart) atau melihat distribusi data (histogram).

Nah, semua grafik ini bisa dibuat dengan mudah menggunakan fungsi-fungsi yang sudah disediakan oleh pyplot, seperti:

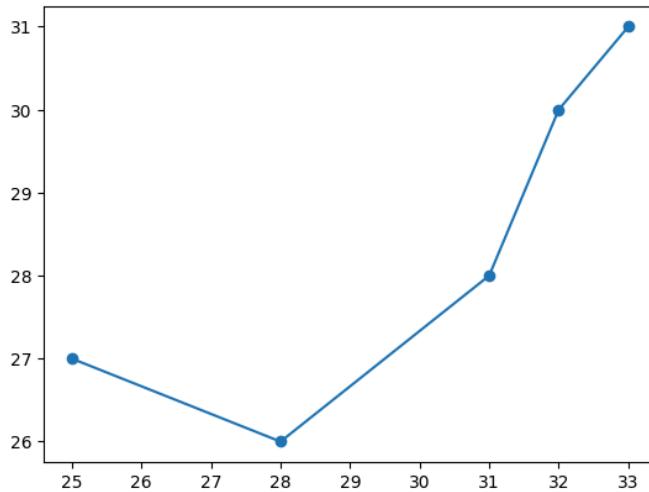
Fungsi Pyplot	Jenis Grafik	Kegunaan Utama
<code>plt.plot()</code>	Grafik Garis	Menampilkan tren atau perubahan nilai dari waktu ke waktu
<code>plt.scatter()</code>	Scatter Plot	Menunjukkan hubungan antar dua variabel
<code>plt.bar()</code>	Diagram Batang	Membandingkan nilai antar kategori
<code>plt.pie()</code>	Diagram Lingkaran	Menampilkan proporsi dari setiap kategori terhadap soal
<code>plt.hist()</code>	Histogram	Melihat sebaran (distribusi) data numerik

Yuk kita bahas satu-satu

a. Line Plot

Sebelum lanjut ke jenis grafik lainnya, mari kita ingat kembali bentuk visualisasi paling dasar yang sudah dibahas sebelumnya, yaitu grafik garis (line plot). Grafik ini biasanya digunakan untuk menampilkan perubahan nilai dari waktu ke waktu atau tren data secara berurutan.

```
from matplotlib import markers
x = [25, 28, 31, 32, 33]
y = [27, 26, 28, 30, 31]
plt.plot(x, y, marker = 'o')
plt.show()
```



Nah, kalau line plot menampilkan perubahan nilai, maka berikutnya kita akan belajar grafik yang menampilkan hubungan antar dua variabel, yaitu Scatter Plot.

b. Scatter Plot

Setelah kita berhasil membuat grafik garis sederhana, sekarang saatnya mencoba bentuk visualisasi lain yang sering digunakan untuk menganalisis hubungan antar data, yaitu scatter plot atau diagram pencar.

Apasih scatter plot itu? Scatter plot adalah jenis visualisasi data yang digunakan untuk menampilkan hubungan antara dua variabel dalam bentuk titik-titik pada bidang koordinat. Setiap titik mewakili satu pasang nilai dari dua variabel (biasanya disebut variabel x dan y).

Visualisasi jenis ini sangat berguna untuk melihat bagaimana grafik ini bisa menunjukkan hubungan antar dua variabel. Dalam analisis data, scatter plot sering digunakan untuk melihat arah korelasi. Misalnya, kita bisa mengetahui apakah dua variabel memiliki korelasi positif, negatif, atau tidak memiliki hubungan sama sekali. Scatter plot juga membantu menemukan anomali (outlier) atau data yang tampak menyimpang / berbeda jauh dari pola umum.

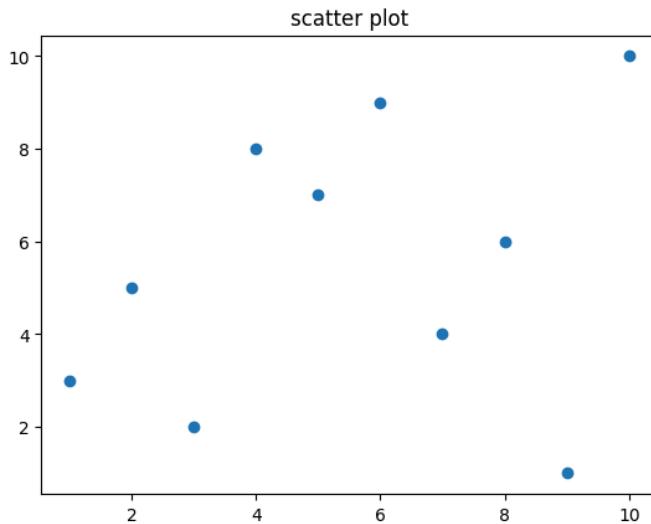
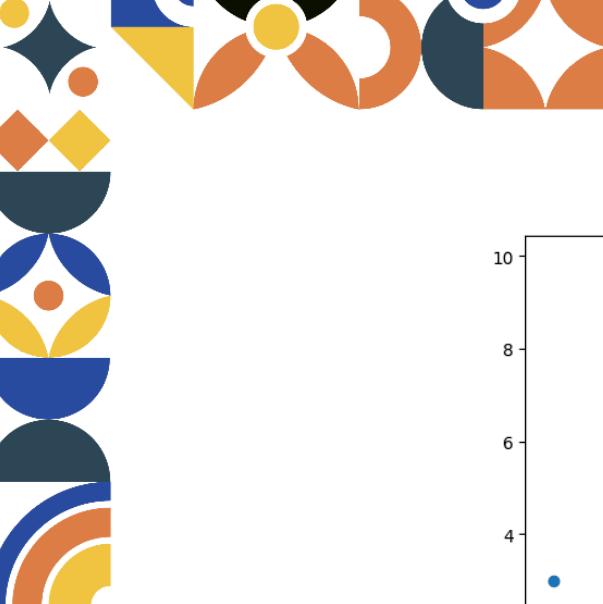
Berikut contoh program sederhana untuk membuat scatter plot menggunakan Matplotlib:

```

x = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
y = [3, 5, 2, 8, 7, 9, 4, 6, 1, 10]
plt.scatter(x, y)
plt.title('scatter plot')
plt.show()
#percobaan2

```





Pada kode di atas, kita membuat dua list bernama `x` dan `y` yang masing-masing berisi lima nilai.

List pertama `(x)` berisi data untuk sumbu x, dan list kedua `(y)` berisi data untuk sumbu y. Fungsi `plt.scatter()` kemudian digunakan untuk menggambar titik-titik pada koordinat yang sesuai antara nilai x dan y. Terakhir, perintah `plt.show()` digunakan untuk menampilkan hasilnya dalam bentuk grafik. Hasil yang muncul adalah sebuah scatter plot sederhana dengan titik-titik yang menggambarkan pasangan nilai dari kedua variabel.

Grafik ini bisa kita modifikasi agar tampil lebih informatif dan menarik – misalnya dengan mengubah warna titik, menentukan ukuran titik, atau menambahkan label dan judul grafik.

Emang bisa ya?? bisa dong

Matplotlib menyediakan berbagai parameter dalam fungsi `plt.scatter()` yang bisa digunakan untuk menyesuaikan tampilan plot sesuai kebutuhan kita. Misalnya, kita bisa menambahkan argumen `color`, `s`, dan `label` untuk mengatur warna, ukuran, serta keterangan titik. kalian bisa baca referensi dari [sini](#)

📍 Latihan 2

Sekarang coba kalian modifikasi tampilan pada scatter plot **#percobaan2** dengan menambahkan label dan mengganti warna atau sesuai kreativitas kalian

c. Pie Chart

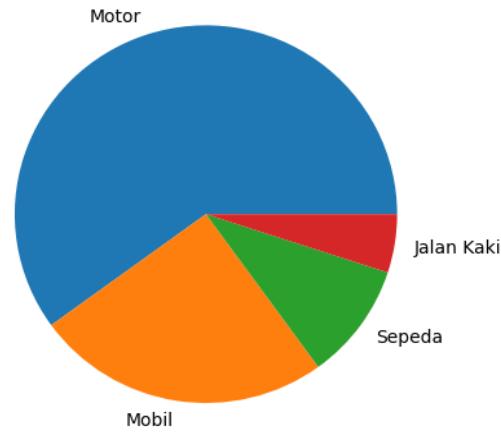
Selain grafik garis dan scatter plot yang sudah kita pelajari sebelumnya, terdapat juga jenis visualisasi lain yang sering digunakan untuk menampilkan data kategori, yaitu Pie Chart atau diagram lingkaran.



Pie chart adalah jenis diagram yang menggambarkan data dalam bentuk lingkaran, dimana lingkaran tersebut dibagi menjadi irisan sesuai proporsi masing-masing kategori. Setiap bagian dari lingkaran mewakili kontribusi suatu kategori terhadap total keseluruhan.

```
# Data jenis transportasi yang digunakan mahasiswa
transportasi = {
    'Motor': 60,
    'Mobil': 25,
    'Sepeda': 10,
    'Jalan Kaki': 5
}
# Memisahkan data menjadi label dan nilai
labels = list(transportasi.keys())
values = list(transportasi.values())
# Membuat pie chart dasar
plt.pie(values, labels=labels)
plt.title("Persentase Jenis Transportasi Mahasiswa")
plt.show()
```

Persentase Jenis Transportasi Mahasiswa



Pada kode di atas, kita memiliki data dictionary bernama `transportasi` yang berisi data jenis kendaraan beserta jumlah penggunanya. Setiap key menunjukkan jenis kendaraan, dan value menunjukkan jumlah mahasiswa yang menggunakan kendaraan tersebut.

Data itu kemudian dipecah menjadi 2 list. `labels` berisi nama kategori (label kendaraan). dan `values` berisi nilai tiap masing-masing kategori.

Fungsi `plt.pie()` digunakan untuk membuat diagram lingkaran berdasarkan data tersebut, sedangkan `plt.show()` digunakan untuk menampilkan hasil grafik.



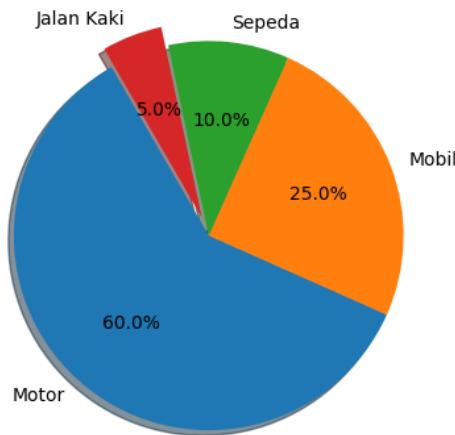
Hasilnya adalah sebuah pie chart sederhana yang menampilkan proporsi jenis transportasi mahasiswa.

Setelah memahami dasar dari pie chart, kita bisa memodifikasinya agar tampil lebih menarik dan informatif. Kita dapat menambahkan label persentase, efek bayangan, serta menonjolkan satu irisan tertentu (menggunakan `explode`) agar terlihat menonjol.

```
# Menambahkan efek explode pada pie chart
explode = [0, 0, 0, 0.1] # Menyoroti kategori 'Jalan Kaki' (di index 3)
yang memiliki jumlah paling sedikit

plt.pie(
    values,
    labels=labels,
    explode=explode,
    autopct='%.1f%%',
    shadow=True,
    startangle=120
)
plt.title("Persentase Jenis Transportasi Mahasiswa")
plt.show()
```

Persentase Jenis Transportasi Mahasiswa



Pada contoh ini, kita menggunakan data transportasi yang sama, tetapi menambahkan beberapa parameter tambahan untuk mempercantik visualisasi:

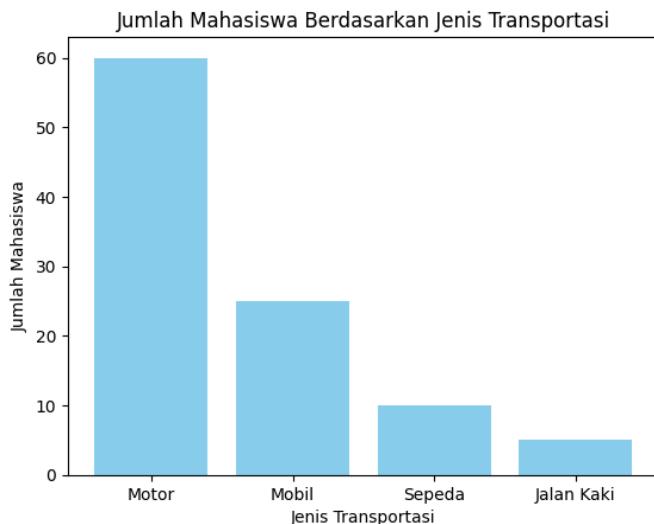
- `explode` → digunakan untuk menyoroti kategori tertentu dengan menarik irisan keluar dari lingkaran, nilai 0 berarti irisan tetap berada di lingkaran, nilai positif (misalnya 0.1 atau 0.2) akan menarik irisan keluar.
- `autopct` → menampilkan nilai persentase pada setiap irisan.
- `shadow` → memberikan efek bayangan pada pie chart agar terlihat lebih menonjol.

- startangle → mengatur sudut awal tampilan diagram agar hasilnya lebih proporsional.

d. Bar Chart

Setelah mempelajari diagram lingkaran, sekarang kita akan membuat Bar Chart (diagram batang). Bar chart berguna untuk membandingkan jumlah atau nilai antar kategori dengan lebih jelas. Setiap batang mewakili satu kategori dan panjangnya menunjukkan besar nilai dari kategori tersebut.

```
# kita akan menggunakan Data jenis transportasi yang digunakan mahasiswa
plt.bar(labels, values, color='skyblue')
plt.title("Jumlah Mahasiswa Berdasarkan Jenis Transportasi")
plt.xlabel("Jenis Transportasi")
plt.ylabel("Jumlah Mahasiswa")
plt.show()
```



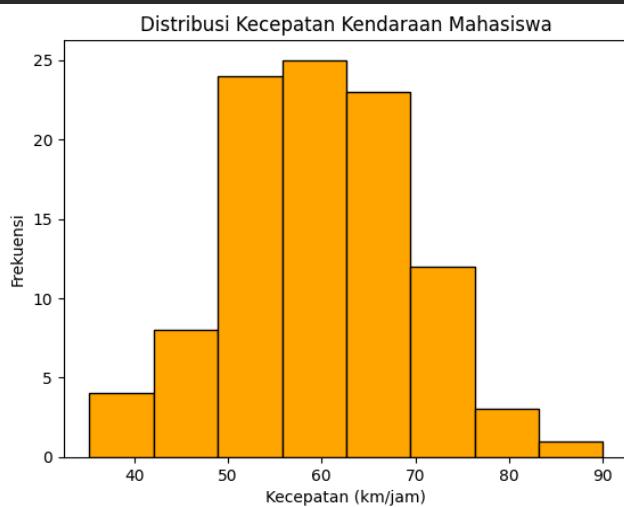
Pada kode ini, fungsi `plt.bar()` digunakan untuk membuat diagram batang dari data yang sama seperti pada pie chart sebelumnya. Diagram ini memudahkan kita membandingkan jumlah dengan lebih jelas secara visual

e. Histogram

Berbeda dengan Bar Chart yang digunakan untuk data kategori, Histogram digunakan untuk data numerik. Histogram membantu kita melihat bagaimana data tersebar – misalnya seberapa sering nilai tertentu muncul dalam sebuah kumpulan data.

```
import numpy as np
# Data simulasi kecepatan kendaraan (dalam km/jam)
kecepatan = np.random.normal(60, 10, 100) # 100 data acak dengan
rata-rata 60 dan deviasi 10
```

```
plt.hist(kecepatan, bins=8, color='orange', edgecolor='black')
plt.title("Distribusi Kecepatan Kendaraan Mahasiswa")
plt.xlabel("Kecepatan (km/jam)")
plt.ylabel("Frekuensi")
plt.show()
```



Kode di atas menggunakan fungsi `plt.hist()` untuk membuat histogram dari data kecepatan kendaraan. Setiap batang (bar) pada grafik menunjukkan berapa banyak data yang masuk ke dalam rentang nilai tertentu (yang disebut bin). Misalnya, jika `bins=8`, maka seluruh data akan dibagi ke dalam 8 kelompok nilai kecepatan (ada 8 bar).

Semakin tinggi batang pada histogram, berarti semakin banyak data yang berada di rentang kecepatan tersebut. Dari hasil grafik, kita bisa melihat bagaimana data tersebar — apakah kebanyakan mahasiswa berkendara di sekitar kecepatan tertentu (misalnya 60 km/jam), apakah ada yang jauh lebih cepat atau lebih lambat, dan seberapa lebar variasi kecepatannya.

Dengan kata lain, histogram membantu kita memahami pola distribusi data numerik, misalnya apakah datanya normal (menumpuk di tengah), miring ke kiri/kanan, atau memiliki banyak nilai ekstrem (outlier).

1.3. Subplot dalam matplotlib

Subplot adalah fitur dalam Matplotlib yang memungkinkan kita menampilkan beberapa grafik dalam satu jendela (figure/gambar). Fitur ini sangat berguna ketika kita ingin:

- Membandingkan beberapa dataset secara langsung, atau
- Menampilkan berbagai jenis grafik yang berasal dari data yang sama dalam satu tampilan.
- Misalnya, kita bisa menampilkan grafik garis dan pie chart dari data yang sama dalam satu figure agar mudah dibandingkan.

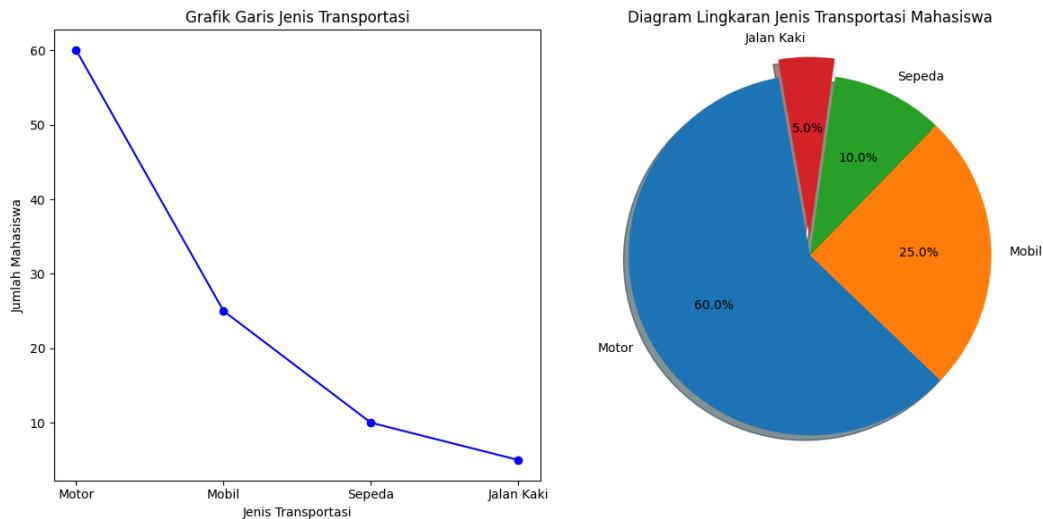
pastikan kalian sudah run semua cell ya

```
# Membuat figure dan menentukan ukuran tampilan
plt.figure(figsize=(12, 6))

# Subplot pertama: grafik garis
plt.subplot(1, 2, 1)
plt.plot(labels, values, marker='o', color='blue')
plt.title("Grafik Garis Jenis Transportasi")
plt.xlabel("Jenis Transportasi")
plt.ylabel("Jumlah Mahasiswa")

# Subplot kedua: pie chart
plt.subplot(1, 2, 2)
plt.pie(values, labels=labels, explode=explode, autopct='%.1f%%',
shadow=True, startangle=100)
plt.title("Diagram Lingkaran Jenis Transportasi Mahasiswa")

# Menampilkan seluruh subplot
plt.tight_layout()
plt.show()
#percobaan3
```



Pada kode di atas, kita menggunakan fitur subplot untuk menampilkan dua grafik sekaligus dalam satu tampilan.

Langkah pertama, kita membuat figure baru menggunakan `plt.figure(figsize=(12, 6))`. Parameter figsize di sini berfungsi untuk mengatur ukuran tampilan agar grafik tidak terlihat terlalu kecil atau saling berhimpitan – pada contoh ini lebarnya 12 dan tingginya 6 satuan.

Setelah itu, kita membuat grafik pertama menggunakan `plt.subplot(1, 2, 1)`. Angka (1, 2, 1) menandakan bahwa kita ingin membuat satu baris dengan dua kolom grafik, dan yang sedang digambar saat ini berada di posisi pertama (bagian kiri).

Selanjutnya, kita membuat subplot kedua dengan `plt.subplot(1, 2, 2)`, yang akan muncul di sisi kanan.

📌 Latihan 3

Sekarang coba tambahkan subplot ketiga di bawah dua grafik sebelumnya (#percobaan3) untuk menampilkan diagram batang (bar chart) dari data yang sama. Gunakan fungsi `plt.subplot()` dengan parameter yang sesuai agar bar chart muncul di baris kedua.

- Ubah konfigurasi subplot dari (1, 2, 1) menjadi (2, 2, 1) agar figure memiliki 2 baris dan 2 kolom grafik.
- Tempatkan bar chart di subplot ke-3 dengan memanggil `plt.subplot(2, 2, 3)`.

2. NUMPY

NumPy (singkatan dari Numerical Python) adalah salah satu library Python yang digunakan untuk mengolah data angka. Dengan NumPy, kita bisa bekerja dengan data berbentuk array, yaitu kumpulan angka yang bisa tersusun dalam baris dan kolom – mirip seperti tabel atau lembar kerja di Excel.

NumPy membantu kita melakukan berbagai perhitungan matematika dengan cepat dan mudah, seperti penjumlahan, pengurangan, rata-rata, atau operasi lainnya tanpa perlu menulis kode yang panjang. NumPy juga sering digunakan bersama library lain di Python, seperti Matplotlib untuk membuat grafik, Pandas untuk analisis data, dan Seaborn untuk visualisasi statistik.

Jadi, bisa dibilang NumPy adalah dasar / pondasi utama dari berbagai library yang sering dipakai di dunia data science karena semua library tersebut memakai data dalam bentuk array dari NumPy yang artinya bisa langsung dipakai di Matplotlib tanpa perlu diubah bentuknya dulu. Untuk penjelasan lebih lanjut bisa kalian baca-baca [disini](#) dan dokumentasi resmi tentang NumPy, kalian dapat mengunjungi situs web resmi NumPy [disini](#)

Untuk menggunakan NumPy, pastikan library ini sudah terinstal di perangkat kalian. Jika belum, kalian bisa melakukan instalasi melalui terminal, command prompt dengan perintah berikut

```
pip install numpy
```

Jika kalian menggunakan Google Colab, cukup jalankan perintah ini di cell baru dengan menambahkan tanda seru di depannya:

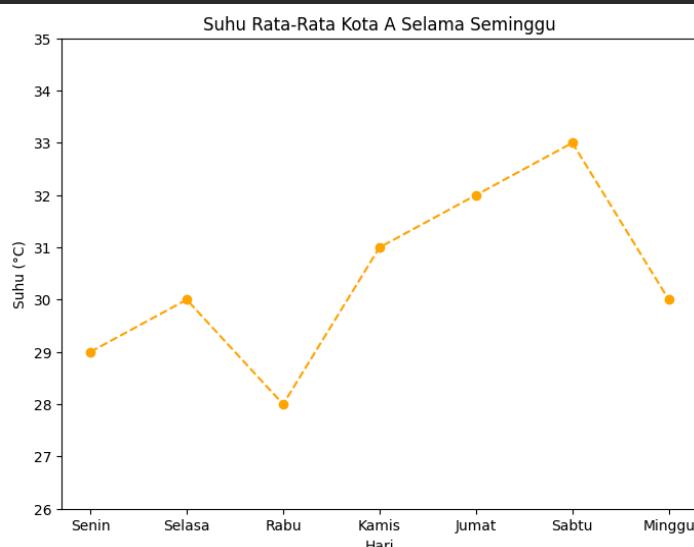
```
!pip install numpy
```

Berikut adalah contoh penggunaan NumPy dalam visualisasi data menggunakan Matplotlib:

a. Percobaan 1: Variasi visualisasi data

Pada percobaan ini, kita akan menggunakan NumPy untuk membuat data suhu rata-rata di suatu kota selama satu minggu, lalu menampilkannya dalam bentuk grafik garis menggunakan Matplotlib.

```
import matplotlib.pyplot as plt
import numpy as np #jangan lupa import library ya
# Data suhu rata-rata (dalam °C) selama 7 hari
hari = np.array(["Senin", "Selasa", "Rabu", "Kamis", "Jumat", "Sabtu",
"Musim"])
suhu_kotaA = np.array([29, 30, 28, 31, 32, 33, 30])
# Membuat figure dan diagram plot / garis
plt.figure(figsize=(8, 6))
plt.plot(hari, suhu_kotaA, color='orange', linestyle='--', marker='o')
# Memberi judul dan label
plt.title("Suhu Rata-Rata Kota A Selama Seminggu")
plt.xlabel("Hari")
plt.ylabel("Suhu (°C)")
# Mengatur batas sumbu Y agar lebih rapi
plt.ylim([26, 35]) # y dimulai dari 26-35
plt.show()
```



Pada kode di atas, kita membuat dua array dengan `np.array()`, yaitu `hari` sebagai sumbu X dan `suhu_kotaA` sebagai sumbu Y. menambahkan parameter seperti:

- `color='orange'` untuk memberi warna garis,

- `linestyle='--'` agar tampil dengan gaya garis putus-putus, dan
- `marker='o'` untuk menampilkan titik di setiap data.

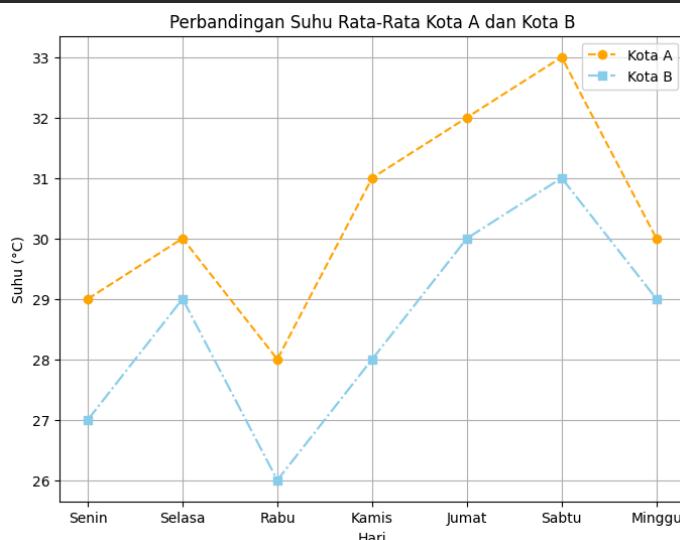
Selengkapnya bisa kalian pelajari melalui link [disini](#).

b. Percobaan 2: Multiple lines

Sekarang, kita lanjutkan percobaan sebelumnya. Misalnya, kita ingin membandingkan suhu rata-rata antara dua kota (Kota A dan Kota B) selama seminggu dalam satu grafik.

pastiin cell sebelumnya sudah di run ya

```
# Data suhu kotaB rata-rata (°C) selama 7 hari
suhu_kotaB = np.array([27, 29, 26, 28, 30, 31, 29])
# Membuat grafik dengan dua garis
plt.figure(figsize=(8, 6))
plt.plot(hari, suhu_kotaA, label="Kota A", color='orange', marker='o',
linestyle='--')
plt.plot(hari, suhu_kotaB, label="Kota B", color='skyblue', marker='s',
linestyle='-.')
# Menambahkan elemen pendukung grafik
plt.title("Perbandingan Suhu Rata-Rata Kota A dan Kota B")
plt.xlabel("Hari")
plt.ylabel("Suhu (°C)")
plt.legend()
plt.grid(True)
plt.show()
#percobaan4
```



Pada kode ini, kita masih menggunakan data `hari` yang sama, tetapi menambahkan satu array baru `suhu_kotaB` untuk kota lain. Kedua data kemudian divisualisasikan bersama menggunakan dua fungsi `plt.plot()`.

Parameter `label` digunakan untuk memberi keterangan pada masing-masing garis, dan `plt.legend()` berfungsi menampilkan kotak legenda agar pembaca bisa membedakan garis untuk tiap kota. Tambahan `plt.grid(True)` memberikan garis bantu pada grafik supaya lebih mudah dibaca.

📌 Latihan 4

Coba kalian tambahkan satu kota lagi "Kota C" dengan suhu yang berbeda (bebas), lalu ditampilkan dalam grafik yang sama. Dan ubah gaya garis (linestyle) dan warna (color) agar tiap kota terlihat berbeda. Contoh output bisa dilihat di [colab](#)

Kenapa harus Numpy

```
# Data tanpa np.array (numpy)
hari = ["Senin", "Selasa", "Rabu", "Kamis", "Jumat", "Sabtu", "Minggu"]
# Data suhu kotaB rata-rata (°C) selama 7 hari
suhu_kotaB = [27, 29, 26, 28, 30, 31, 29]
suhu_kotaA = [29, 30, 28, 31, 32, 33, 30]
# Membuat grafik dengan dua garis
plt.figure(figsize=(8, 6))
plt.plot(hari, suhu_kotaA, label="Kota A", color='orange', marker='o',
         linestyle='--')
plt.plot(hari, suhu_kotaB, label="Kota B", color='skyblue', marker='s',
         linestyle='-.')
# Menambahkan elemen pendukung grafik
plt.title("Perbandingan Suhu Rata-Rata Kota A dan Kota B")
plt.xlabel("Hari")
plt.ylabel("Suhu (°C)")
plt.legend()
plt.grid(True)
plt.show()
```

Apa bedanya cell diatas dengan Percobaan 2? bukankah outputnya sama aja? cari jawabannya dengan mengerjakan [TUGAS 2](#)

Jadi, inti dari Numpy ini mengolah / bermain hanya dengan angka atau array. naahh gimana jika ada data lain seperti text atau list atau dictionary?? kenalan yuk sama Pandas

3. PANDAS

Pandas adalah salah satu library Python yang sangat populer dan banyak digunakan untuk mengolah serta menganalisis data. Library ini memudahkan kita untuk membaca, memanipulasi, membersihkan, dan menampilkan data dengan cara yang efisien dan praktis.

Kalau diibaratkan, Pandas ini seperti “Excel-nya Python”, tapi jauh lebih fleksibel dan bisa diatur lewat kode program. Dengan Pandas, kita bisa membaca, mengubah, menghitung, dan menampilkan data dengan sangat mudah. Misalnya membaca file CSV atau Excel, menghapus data kosong, menghitung rata-rata, sampai menampilkan data tertentu sesuai kebutuhan kita. Biasanya Pandas digunakan bersama library lain seperti NumPy, Matplotlib, atau Seaborn untuk analisis dan visualisasi data. Pandas memiliki dua struktur data utama:

- Series: Struktur satu dimensi yang bisa memuat data dari berbagai tipe (integer, string, float, dll.). Dapat diakses menggunakan indeks.
- DataFrame: Struktur data dua dimensi yang berbentuk tabel, dengan baris dan kolom. DataFrame merupakan gabungan dari beberapa Series.

Kedua struktur ini sangat berguna untuk menyimpan dan mengolah data dalam berbagai bentuk – mulai dari angka, teks, hingga hasil perhitungan statistik.

Untuk menggunakan Pandas, pastikan library ini sudah terinstal. Jika belum, kalian bisa menjalankan perintah berikut di terminal / command prompt

```
pip install pandas
```

Jika kalian menggunakan Google Colab, jalankan dengan tanda seru di depannya:

```
!pip install pandas
```

3.1. Pandas Series

Pandas Series adalah struktur data satu dimensi yang mirip seperti satu kolom di tabel Excel. Setiap elemen dalam Series punya label atau indeks di sebelah kirinya, sehingga kita bisa dengan mudah mencari atau mengambil data berdasarkan indeks tersebut. Series bisa berisi berbagai jenis data seperti angka, teks, bahkan campuran keduanya. Biasanya, Series digunakan untuk menyimpan data sederhana sebelum akhirnya digabungkan ke dalam bentuk tabel yang lebih besar, yaitu DataFrame.

a. Percobaan 1: Membuat Series Menggunakan Array Sederhana

Cara paling sederhana untuk membuat Series adalah menggunakan array (misalnya dari NumPy / list biasa). Pandas Series memiliki indeks di sebelah kiri dan data di sebelah kanan, dimana setiap elemen otomatis diberi nomor indeks mulai dari 0, kecuali kita tentukan sendiri.



```
import pandas as pd # jangan lupa import library ya
# Membuat series kosong
series = pd.Series()
print("Pandas Series: ", series)
#output:
Pandas Series:  Series([], dtype: object)
```

```
# Membuat array menggunakan list biasa berisi daftar nama buah
data = ['Apel', 'Jeruk', 'Mangga', 'Pisang', 'Anggur']
# Memasukkan array ke dalam series
series = pd.Series(data)
print("Pandas Series:\n", series)
#output:
Pandas Series:
 0      Apel
 1      Jeruk
 2      Mangga
 3      Pisang
 4      Anggur
dtype: object
```

```
# Mengakses data berdasarkan indeks
print("\nData pada indeks ke-2:", series[2]) #ingat index mulai dari 0
#output:
Data pada indeks ke-2: Mangga
```

b. Percobaan 2: Membuat Series dari Dictionary

Selain dari array, kita juga bisa membuat Series dari dictionary. Hal ini sangat berguna kalau kita sudah punya data dalam bentuk pasangan key–value. Dengan menggunakan dictionary untuk membuat Series, kita dapat dengan cepat dan efisien mengorganisir data yang terstruktur.

```
# Membuat dictionary berisi daftar menu dan harga
daftar_menu = {
    "Nasi Goreng": 12000,
    "Es Jeruk": 3000,
    "Dimsum": 15000,
    "Bakso": 13000,
    "Es Teh": 2000
}
```

```
# Membuat Series dari dictionary
series_harga = pd.Series(daftar_menu)
print("Daftar menu dan harganya:\n",series_harga)
#output:
Daftar menu dan harganya:
Nasi Goreng      12000
Es Jeruk         3000
Dimsum           15000
Bakso             13000
Es Teh            2000
dtype: int64
```

Pada kode diatas, saat kita ubah menjadi Series dengan `pd.Series()`, daftar menu (key) otomatis menjadi label (indeks) dan harga (value) menjadi nilai datanya. Nah karena label pada Series ini adalah nama menu, kita bisa mengakses data tidak hanya dengan nomor indeks, tapi juga dengan nama label-nya langsung. Contohnya seperti ini:

```
print("Harga Es teh :", series_harga['Es Teh'])
#output:
Harga Es teh : 2000
```

Kode di atas akan langsung menampilkan harga dari menu Es Teh tanpa perlu tahu posisi (indeks) datanya. Dengan begitu, kita bisa mengambil data dengan cara yang lebih mudah dan jelas, apalagi kalau dataset-nya cukup besar.

Jika diperhatikan lebih lanjut, sejauh ini penggunaan pandas series tidak ada bedanya dengan data aslinya (list dan dictionary). Lalu, untuk apa kita perlu pandas library? Coba perhatikan kode berikut:

```
# Mengakses data sesuai statistik
print("\nHarga tertinggi:", series_harga.max())
print("\nHarga terendah:", series_harga.min())
print("\nHarga rata-rata:", series_harga.mean())
#output
Harga tertinggi: 15000
Harga terendah: 2000
Harga rata-rata: 9000.0
```

Pada contoh sebelumnya, kita melihat bahwa membuat Series dari list atau dictionary terlihat mirip dengan struktur datanya. Namun, perbedaan dan kelebihan Pandas Series terlihat ketika kita mulai melakukan operasi analisis data.

Dengan Pandas, kita bisa menghitung nilai statistik seperti nilai maksimum, minimum, dan rata-rata hanya dengan satu baris kode. Jika kita menggunakan list atau dictionary, kita perlu melakukan perhitungan tersebut secara manual, misalnya dengan `max()`, `min()`, atau bahkan menulis loop sendiri untuk mencari nilai rata-rata menggunakan `reduce` atau rekursif. Tentu saja, kode menjadi lebih panjang, imperatif, dan kurang praktis untuk analisis data.

Naahh, dengan library Pandas, semua fungsi statistik ini sudah tersedia langsung sebagai metode bawaan pada Series. Jadi, meskipun pada awalnya Series terlihat seperti list atau dictionary dengan tambahan label, sebenarnya Pandas menyediakan alat analisis yang jauh lebih lengkap, efisien, dan mudah digunakan, terutama ketika dataset semakin besar dan kompleks.

3.2. Dataframe Pandas

Kalau Series tadi bisa diibaratkan sebagai satu kolom data, maka DataFrame adalah kumpulan beberapa kolom yang membentuk sebuah tabel – mirip seperti lembar Excel. Setiap kolom di dalam DataFrame sebenarnya adalah Series juga, tapi disusun berdampingan dengan label kolom dan baris. Dataframe ini dibuat dengan memuat kumpulan data dari storage yang ada (bisa berupa database SQL, file CSV, atau file Excel). Dataframe Pandas dapat dibuat dari list, dictionary, kamus, list of dictionaries, dll.

a. Percobaan 1: Membuat Dataframe dari Array Sederhana

Membuat Dataframe dari Array Sederhana adalah salah satu cara dasar dalam menggunakan library Pandas untuk memanipulasi dan menganalisis data.

```
import pandas as pd
# Membuat list berisi praktikum di sem 5
praktikum = ['Fungsional', 'Jaringan Komputer', 'Web', 'Mobile', 'Piranti Cerdas']
# Membuat DataFrame dari list
df_matkul = pd.DataFrame(praktikum, columns=['Praktikum'])
print("DataFrame Praktikum:\n", df_matkul)
#output
DataFrame Praktikum:
  Praktikum
0      Fungsional
1  Jaringan Komputer
2          Web
3        Mobile
4    Piranti Cerdas
```



List **praktikum** diubah menjadi DataFrame dengan satu kolom bernama "Praktikum". Hasilnya mirip seperti daftar praktikum dalam tabel.

b. Percobaan 2: Membuat Dataframe dari Dictionary Pandas

Dictionary di Python adalah struktur data yang terdiri dari pasangan kunci dan nilai, di mana kunci dapat digunakan sebagai nama kolom di dalam DataFrame. Dengan menggunakan dictionary, pengguna dapat dengan mudah mengorganisasi data dan membuat Dataframe yang rapi, memungkinkan manipulasi dan analisis data yang lebih efisien.

Sekarang kita akan mengembangkan data dari percobaan sebelumnya. Kita sudah punya daftar nama praktikum (**praktikum**), dan sekarang kita tambahkan informasi baru seperti nama mahasiswa dan nilai akhir untuk tiap praktikum.

pastiin kalian sudah run cell percobaan 1 sebelumnya ya

```
# Membuat dictionary berisi data mahasiswa dan nilai
data_praktikum = {
    'Nama Mahasiswa': ['Ana', 'Ayu', 'Nia', 'Lia', 'Asa'],
    'Praktikum': praktikum,
    'Nilai': [90, 85, 88, 92, 80]
}

# Membuat DataFrame dari dictionary
df_praktikum = pd.DataFrame(data_praktikum)
print("DataFrame Nilai Praktikum Mahasiswa:\n", df_praktikum)

#output
DataFrame Nilai Praktikum Mahasiswa:
   Nama Mahasiswa      Praktikum  Nilai
0          Ana        Fungsional    90
1         Ayu  Jaringan Komputer    85
2          Nia           Web     88
3          Lia        Mobile     92
4          Asa  Piranti Cerdas    80
```

c. Percobaan 3: Membaca Data dengan Pandas

Salah satu langkah penting dalam analisis data adalah kemampuan untuk membaca dan mengolah data dari berbagai sumber. Pandas, sebagai library yang kuat untuk analisis data di Python, menyediakan berbagai fungsi untuk memudahkan proses ini.

Salah satu format data yang paling umum digunakan adalah CSV (Comma-Separated Values) – yaitu format file teks sederhana yang berisi data

berbentuk tabel dan bisa dibuka oleh berbagai aplikasi seperti Excel, Google Sheets, atau Python.

Dalam percobaan kali ini, silahkan download file CSV [berikut](#). Source: Kaggle. Pastikan kalian run cell berikut secara berurutan sampai selesai! Cara Upload File CSV ke Google Colab bisa langsung kalian baca di [colab](#) ya

Naah gimana kalau kalian pakai laptop sendiri / lokal??

Kalau kalian menjalankan program Python di lokal (bukan di Google Colab), maka cara membaca file CSV sedikit berbeda. Kalian tidak perlu meng-upload file – cukup pastikan file .csv berada di folder yang sama dengan file Python (.py/.ipynb) kalian.

Pandas akan otomatis mencari file .csv di direktori kerja (folder) tempat file .py atau notebook kalian disimpan. Jika file berada di folder lain, kalian bisa menulis path lengkapnya, contohnya:

```
data = pd.read_csv('C:/Users>NamaKalian/Documents/nama_csv.csv')
```

d. Percobaan 4: Operasi Dasar pada DataFrame

Sekarang kita akan belajar beberapa operasi dasar pada DataFrame untuk mengenal isi data lebih dalam. Setelah file CSV berhasil dibaca dan dimasukkan ke dalam variabel data, kita bisa mulai melakukan eksplorasi sederhana untuk memahami struktur dan isi dari dataset tersebut.

- **Menampilkan informasi umum tentang DataFrame:**

```
data.info()
#output:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 200 entries, 0 to 199
Data columns (total 9 columns):
 #   Column           Non-Null Count  Dtype  
--- 
 0   Transaction ID  200 non-null    int64  
 1   Date             200 non-null    object  
 2   Customer ID     200 non-null    object  
 3   Gender           200 non-null    object  
 4   Age              200 non-null    int64  
 5   Product Category 200 non-null    object  
 6   Quantity         200 non-null    int64  
 7   Price per Unit   200 non-null    int64  
 8   Total Amount     200 non-null    int64  
dtypes: int64(5), object(4)
```

memory usage: 14.2+ KB

Perintah ini menampilkan informasi dasar tentang DataFrame — seperti jumlah baris data, jumlah kolom, nama kolom, tipe data, dan berapa banyak nilai yang tidak kosong (non-null). Contohnya, kita bisa tahu apakah ada data yang hilang, serta seberapa besar memori yang digunakan untuk menyimpan DataFrame tersebut.

- **Melihat statistik dasar**

```
data.describe()  
#output:
```

	Transaction ID	Age	Quantity	Price per Unit	Total Amount
count	200.000000	200.000000	200.000000	200.000000	200.000000
mean	100.500000	40.120000	2.565000	195.30000	518.175000
std	57.879185	13.704916	1.105389	193.94583	607.239651
min	1.000000	18.000000	1.000000	25.00000	25.000000
25%	50.750000	28.750000	2.000000	30.00000	75.000000
50%	100.500000	40.000000	3.000000	50.00000	175.000000
75%	150.250000	51.000000	4.000000	300.00000	900.000000
max	200.000000	64.000000	4.000000	500.00000	2000.000000

Fungsi describe() memberikan ringkasan statistik dari kolom-kolom numerik dalam DataFrame, seperti:

- nilai minimum dan maksimum,
- rata-rata (mean),
- standar deviasi (std),
- dan persentil (25%, 50%, 75%).

Dengan cara ini, kita bisa mendapat gambaran cepat tentang data — misalnya, apakah ada nilai yang terlalu ekstrem (outlier) atau distribusinya seimbang.

- **Memilih kolom tertentu**

Sering kali kita tidak perlu melihat semua kolom sekaligus. Kita bisa fokus ke kolom tertentu menggunakan nama kolomnya dalam tanda kurung siku []. Misalnya, kalau di dataset kita ada kolom TotalAmount, maka:

```
print(data['Total Amount'])
```



Perintah ini akan menampilkan seluruh nilai dari kolom TotalAmount saja. Kita juga bisa menampilkan beberapa kolom sekaligus dengan menulisnya dalam bentuk list:

```
print(data[['Customer ID', 'Total Amount']])
```

- **Memilih baris dengan kondisi tertentu**

Kita juga bisa memfilter baris berdasarkan kondisi tertentu. Memilih baris dengan kondisi memungkinkan kita untuk fokus pada subset data yang relevan, sehingga analisis yang dilakukan menjadi lebih spesifik dan informatif. Dengan menggunakan teknik ini, kita dapat mengekstrak data yang memenuhi kriteria tertentu dan melakukan analisis lebih lanjut terhadap data tersebut.

Misalnya, kalau kita ingin melihat data transaksi dengan total pembelian lebih dari 500, maka:

```
filtered_data = data[data['Total Amount'] > 500]
display(filtered_data)
```

Kode diatas menggunakan `filtered_data` untuk menyimpan hasil modifikasi sehingga data asli tidak ikut termodifikasi dan tetap mempertahankan data *immutability*

e. Percobaan 5: Operasi pada Kolom DataFrame

Selain menampilkan dan memfilter data, kita juga bisa melakukan operasi aritmatika langsung pada kolom DataFrame. Hal ini sangat berguna untuk menambah, mengurangi, mengalikan, atau membuat kolom baru berdasarkan hasil perhitungan dari kolom yang sudah ada.

Misalnya, kita ingin menambahkan kolom baru bernama Total Setelah Pajak, di mana nilainya didapat dari Total Amount ditambah pajak 10%.

```
# Menambahkan kolom baru 'Total Setelah Pajak' hasil perhitungan
filtered_data['Total Setelah Pajak'] = filtered_data['Total Amount'] *
1.10

# Menampilkan beberapa baris pertama
filtered_data[['Total Amount', 'Total Setelah Pajak']].head()
```

Pada kode di atas, `= filtered_data['Total Amount'] * 1.10` berarti setiap nilai di kolom Total Amount dikalikan dengan 1.10 (menambah 10% pajak).

Hasilnya langsung disimpan ke kolom baru `Total Setelah Pajak`. Operasi ini otomatis berlaku ke seluruh baris (vectorized operation), tanpa perlu looping.

```
# Mengubah nama kolom
filtered_data.rename(columns={'Total Amount': 'Total Pembelian'},
inplace=True)
filtered_data.head()
```

Kita juga bisa membuat kolom baru hasil operasi antar dua kolom. Misalnya, menghitung rata-rata pembelian per item:

```
# Misal dataset punya kolom 'Total Amount' dan 'Quantity'
filtered_data['Rata-rata per Item'] = filtered_data['Total Amount'] /
filtered_data['Quantity']
filtered_data[['Total Amount', 'Quantity', 'Rata-rata per Item']].head()
```

Dari hasil ini, setiap baris akan berisi nilai baru hasil pembagian antara Total Amount dan Quantity. Kita juga bisa melakukan operasi seperti penjumlahan atau pengurangan antar kolom atau antar baris, Atau jika ingin menambah semua nilai dalam satu kolom (misalnya total seluruh transaksi):

```
# Menjumlahkan seluruh nilai di kolom
total_penjualan = filtered_data['Total Amount'].sum()
print("Total seluruh penjualan adalah:", total_penjualan)
#output:
Total seluruh penjualan adalah: 103635
```

Untuk operasi yg lain kalian bisa explore sendiri ya

f. Numpy & Pandas

Setelah mempelajari cara kerja NumPy dan Pandas, penting bagi kita untuk memahami perbedaan peran antara kedua library tersebut. NumPy dan Pandas memang sering digunakan bersama, tetapi masing-masing memiliki fungsi yang berbeda.

NumPy adalah library untuk komputasi numerik di Python. Library ini sangat cocok digunakan ketika kita bekerja dengan perhitungan matematika, vektor, matriks, atau data numerik dalam bentuk array. NumPy dirancang agar operasi numeriknya sangat cepat melalui teknik vectorization, dan biasanya jauh lebih cepat dan efisien dibandingkan Python list dalam operasi matematika.

Pandas lebih berfokus pada data yang terstruktur, terutama data yang umum ditemui sehari-hari seperti tabel Excel, CSV, atau dataset dari database. Pandas menyediakan fitur lengkap untuk membaca data, membersihkan data, memfilter baris, mengelompokkan data, hingga melakukan transformasi yang

kompleks. Karena itu, Pandas lebih cocok untuk pengolahan dan analisis data "dunia nyata" yang biasanya tidak rapi dan bervariasi. Dengan kata lain:

- Gunakan NumPy ketika fokus utama adalah numeric computing atau operasi matematika berbasis array.
- Gunakan Pandas ketika fokusnya adalah data manipulation dan analisis data berbentuk tabel.

Kedua library ini tidak saling menggantikan, tetapi justru saling melengkapi: NumPy memberikan kecepatan dan efisiensi perhitungan, sedangkan Pandas memberikan struktur dan kemudahan manipulasi data.

Aspect	Numpy	Pandas
Data focus	Numerical arrays	Tabular / structured data
Main object	ndarray	DataFrame
Good for	Math, vectors, matrices	Data cleaning, analysis
Speed	Very fast	Slight overhead (built on NumPy)
Data focus	Numerical arrays	Tabular / structured data

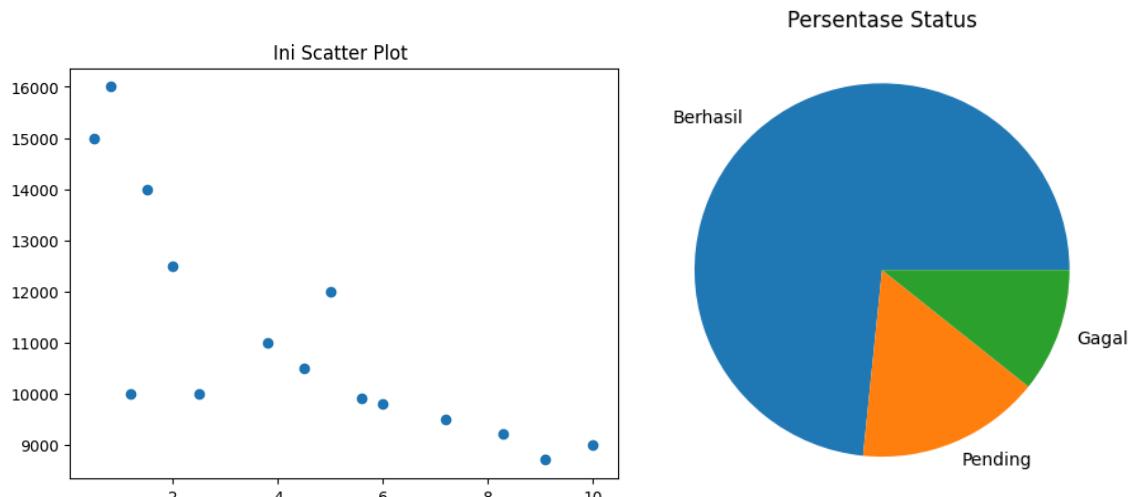
CODELAB

Untuk modul ini tidak ada laporan ya, wajib dikerjakan di lab. Latihan 1-4 dan codelab 1 & 2

CODELAB 1

jangan menggunakan library numpy dan pandas. **gunakan** matplotlib, list comprehension, filter, atau lambda expression.

Pada **TUGAS 2 MODUL 3** kemarin lanjutkan dan buat visualisasi seperti



coba kalian buat visualisasi dan modifikasi sekreatif mungkin supaya menarik dan informatif, seperti menambahkan label, value dalam pie, explode, dll

CODELAB 2

dari dataset 'retail sales' coba kalian lakukan

1. Urutkan data berdasarkan salah satu kolom numerik (misalnya Total Amount atau Quantity), secara: ascending (NIM ganjil) dan descending (NIM genap)
2. Lakukan filter dengan dua kondisi, Quantity > 5 dan Total Amount > 300 Tampilkan hasilnya.
3. Hapus satu kolom yang menurut kalian tidak dibutuhkan dan tampilkan hasil DataFrame baru.

Setelah itu coba kalian buat tampilan visualisasi bebas minimal 3 plot dan modifikasi sekreatif mungkin. tampilkan dalam satu figure (menggunakan subplot)

TUGAS

TUGAS 1

Buatlah sebuah dictionary yang berisi data dari tema studi kasus yang telah kalian gunakan pada Modul 1–4, minimal 10 data / baris. Kemudian ubahlah dictionary tersebut menjadi sebuah DataFrame menggunakan Pandas, dan tampilkan hasil DataFrame-nya.

Gunakan DataFrame tersebut untuk membuat visualisasi data sekreatif mungkin.

Visualisasi data meliputi:

- Penggunaan plot garis, scatter plot, bar chart, dan pie chart, dll (pilih minimal 3)
- Gabungkan minimal tiga plot tersebut ke dalam satu figure menggunakan subplot

Poin penting:

- Studi kasus yang sudah dipilih tidak boleh berubah
- Diperbolehkan untuk menambah atau memodifikasi data agar lebih variatif
- Poin plus untuk modifikasi plot (seperti menambah label, mengubah warna, jenis garis, dll)
- Tidak boleh ada kesamaan dengan praktikan lain

Contoh bentuk dictionary 10 data dan tampilan dataframe bisa dilihat di [colab](#)

TUGAS 2

Melanjutkan dari materi **Kenapa harus NUMPY**

Coba kalian analisa dan jawab pertanyaan yang ada disana untuk tau perbedaan antara penggunaan data list biasa dengan `np.array` / numpy !

Jelaskan jawabanmu ke asisten!!



TUGAS 3

NIM GANJIL

Unduh dataset [berikut](#)

Gunakan Pandas dan Matplotlib untuk analisis dan visualisasi

1. Baca dataset menggunakan Pandas dan tampilkan 10 data pertama.
2. Tampilkan informasi umum tentang dataset
3. Hitung rata-rata skor tiap siswa (Math, Reading, Writing). Simpan hasil rata-rata tersebut ke dalam kolom baru bernama Nilai Total. Pastikan hasilnya tersimpan di DataFrame.
4. Buat visualisasi berupa diagram batang (bar chart) yang membandingkan rata-rata skor siswa laki-laki dan perempuan pada masing-masing dari ketiga mata pelajaran (Math, Reading, Writing).
5. Identifikasi siswa dengan nilai total tertinggi dan terendah lalu tampilkan
6. Buat satu visualisasi tambahan (boleh berupa pie chart, scatter plot, atau jenis grafik lainnya) yang menurut kalian menarik dan relevan untuk menggambarkan hubungan antar variabel dalam dataset ini. Kalian boleh menambahkan visualisasi baru dari kolom apa pun yang menurut kalian penting untuk menjelaskan pola dalam dataset ini dan ditampilkan.

Tambahkan judul, modifikasi tampilan agar lebih informatif dan menarik

NIM GENAP

Unduh dataset [berikut](#)

Gunakan Pandas dan Matplotlib untuk analisis dan visualisasi

1. Baca dataset menggunakan Pandas dan tampilkan 10 data pertama.
2. Tampilkan informasi umum tentang dataset
3. Tambahkan kolom baru bernama `Total`, yang dihitung dari masing-masing cust ('Unit price' * 'Quantity') + 'Tax` Pastikan hasilnya tersimpan di DataFrame.
4. Buat visualisasi diagram batang (bar chart) yang menampilkan rata-rata total penjualan per cabang (Branch).
5. Cari produk dengan total penjualan tertinggi dan terendah (berdasarkan kolom Total) dan tampilkan hasilnya.
6. Buat satu visualisasi tambahan (boleh berupa pie chart, scatter plot, atau jenis grafik lainnya) yang menurut kalian menarik dan relevan untuk menggambarkan hubungan antar variabel dalam dataset ini. Kalian boleh menambahkan visualisasi baru dari kolom apa pun yang menurut kalian penting untuk menjelaskan pola dalam dataset ini dan ditampilkan.

Tambahkan judul, modifikasi tampilan agar lebih informatif dan menarik



RUBRIK PENILAIAN

Komponen Penilaian	Bobot (%)
Codelab	20
Tugas 1	
Kode / Kelengkapan Fitur	10
Kreativitas	10
Pemahaman (Matplotlib)	15
Tugas 2	
Pemahaman (Numpy)	15
Tugas 3	
Kode / Kelengkapan Fitur	10
Kreativitas	5
Pemahaman (Pandas)	15
Total Akhir	100