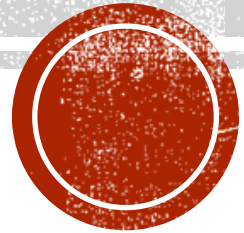
PENGOLAHAN DATA DAN VISUALISASI DENGAN PYTHON



Week 2

Dosen: Yusuf Fadlila Rachman, S.Kom., M.Kom

INSTALASI LIBRARY PYTHON

- -> Open Anaconda Prompt / CMD

- Pip install pandas
- Pip install numpy
- Pip install matplotlib

- -> Open Jupyter Notebook

ARRAY, LIST, DATAFRAME

Fitur	Array	List	DataFrame
Definisi	Struktur data yang berisi elemen dengan tipe data yang sama.	Koleksi elemen yang dapat berisi tipe data berbeda.	Struktur data dua dimensi dengan baris dan kolom, biasanya digunakan untuk data tabular.
Tipe Data	Homogen (semua elemen harus bertipe sama).	Heterogen (elemen bisa memiliki tipe berbeda).	Homogen untuk setiap kolom, tetapi bisa berbeda antar kolom.
Ukuran	Ukuran tetap setelah deklarasi.	Ukuran bisa dinamis, bisa bertambah atau berkurang.	Ukuran bisa dinamis, tetapi lebih terstruktur dan besar.

ARRAY, LIST, DATAFRAME

Fitur	Array	List	DataFrame
Operasi Matematika	Mendukung operasi vektor dan matriks secara langsung (terutama pada **NumPy**).	Tidak mendukung operasi matematika langsung.	Mendukung operasi matematika dan statistik, terutama menggunakan **pandas**.
Fungsi dan Metode	Terbatas pada fungsi dasar yang disediakan oleh pustaka seperti **NumPy**.	Memiliki beberapa fungsi dasar untuk manipulasi data.	Memiliki banyak fungsi untuk manipulasi dan analisis data, seperti **groupby**, **merge**, **pivot**, dll.
Akses Elemen	Indeks berbasis integer.	Indeks berbasis integer atau elemen bisa diakses dengan indeks atau nilai.	Akses elemen bisa menggunakan nama kolom atau indeks baris.

ARRAY, LIST, DATAFRAME

Fitur	Array	List	DataFrame
Kecepatan	Lebih cepat dalam pemrosesan matematis, terutama untuk data numerik.	Relatif lebih lambat jika digunakan untuk operasi numerik kompleks.	Sedikit lebih lambat dibanding array atau list dalam operasi numerik, tetapi sangat efisien untuk data tabular besar.
Penggunaan Umum	Digunakan untuk operasi numerik, komputasi ilmiah, dan algoritma berbasis angka.	Digunakan untuk koleksi data heterogen atau data yang tidak terstruktur.	Digunakan untuk analisis data dan statistik, terutama dengan data tabular besar.

NUMPY

Code Examples :

```
import numpy as np

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

std_dev = np.std(data)
variance = np.var(data)

print(f"Standar Deviasi: {std_dev}")
print(f"Varians: {variance}")
```

Fungsi / Metode	Deskripsi	Contoh Penggunaan
`np.mean()`	Menghitung rata-rata dari array.	`np.mean([1, 2, 3, 4, 5])` akan menghasilkan `3.0`.
`np.median()`	Menghitung median dari array.	`np.median([1, 2, 3, 4, 5])` akan menghasilkan `3.0`.
`np.std()`	Menghitung deviasi standar dari array (penyebaran data).	`np.std([1, 2, 3, 4, 5])` akan menghasilkan `1.41`.
`np.var()`	Menghitung varians dari array.	`np.var([1, 2, 3, 4, 5])` akan menghasilkan `2.0`.
`np.min()` dan `np.max()`	Menghitung nilai minimum dan maksimum dari array.	`np.min([1, 2, 3, 4, 5])` akan menghasilkan `1` dan `np.max()` menghasilkan `5`.
`np.percentile()`	Menghitung persentil ke-n dari array.	`np.percentile([1, 2, 3, 4, 5], 50)` akan menghasilkan `3.0`.
`np.corrcoef()`	Menghitung koefisien korelas. .itara	`np.corrcoef([1, 2, 3], [4, 5, 6])` akan

NUMPY

Code Example :

```
import numpy as np
import pandas as pd
from scipy import stats

data = [5, 7, 8, 9, 9]

mean = np.mean(data)
median = np.median(data)
mode = stats.mode(data)[0][0]

print(f"Mean: {mean}")
print(f"Median: {median}")
print(f"Mode: {mode}")
```

`np.corrcoef()`	Menghitung koefisien korelasi antara dua variabel.	`np.corrcoef([1, 2, 3], [4, 5, 6])` akan menghasilkan matriks korelasi.
`np.random.normal()`	Membuat data acak yang terdistribusi normal.	`np.random.normal(0, 1, 1000)` menghasilkan data acak dengan distribusi normal.
`np.random.binomial()`	Membuat data acak berdasarkan distribusi binomial.	`np.random.binomial(10, 0.5, 1000)` menghasilkan distribusi binomial dengan n=10 dan p=0.5.
`np.random.uniform()`	Membuat data acak dengan distribusi seragam.	`np.random.uniform(0, 1, 1000)` menghasilkan angka acak antara 0 dan 1.
`np.random.choice()`	Memilih elemen secara acak dari array atau list.	`np.random.choice([1, 2, 3, 4, 5], size=3)` menghasilkan 3 elemen acak dari array tersebut.
`np.random.seed()`	Menetapkan nilai awal (seed) untuk generator bilangan acak, memungkinkan replikasi hasil	`np.random.seed(42)` memastikan hasil acak dapat direplikasi.

PANDAS

Code Example :

```
import pandas as pd

data = {'Nama': ['Andi', 'Budi', 'Cici', 'Diana', 'Eka'],
        'Usia': [22, 23, 21, 25, 24]}

df = pd.DataFrame(data)

rata_rata = df['Usia'].mean()
varians = df['Usia'].var()

print("Rata-rata Usia:", rata_rata)
print("Varians Usia:", varians)
```

Fungsi / Metode	Deskripsi	Contoh Penggunaan
`df.mean()`	Menghitung rata-rata dari setiap kolom dalam DataFrame.	`df['age'].mean()` akan menghasilkan rata-rata usia dari kolom `age`.
`df.median()`	Menghitung median dari setiap kolom dalam DataFrame.	`df['age'].median()` akan menghasilkan median usia dari kolom `age`.
`df.std()`	Menghitung deviasi standar dari setiap kolom dalam DataFrame.	`df['age'].std()` akan menghasilkan deviasi standar usia dari kolom `age`.
`df.var()`	Menghitung varians dari setiap kolom dalam DataFrame.	`df['age'].var()` akan menghasilkan varians usia dari kolom `age`.
`df.min()` dan `df.max()`	Menghitung nilai minimum dan maksimum dari setiap kolom dalam DataFrame.	`df['age'].min()` menghasilkan nilai usia minimum, `df['age'].max()` menghasilkan nilai usia maksimum.
`df.describe()`	Memberikan ringkasan statistik dari setiap kolom numerik dalam DataFrame (mean, std, min, max, dll).	`df.describe()` akan memberikan ringkasan statistik untuk kolom numerik di DataFrame.

PANDAS

Code Example Menghitung Korelasi Antar Kolom :

```
data = {'Usia': [22, 23, 21, 25, 24],
        'Jam_Belajar': [15, 18, 10, 20, 17]}

df = pd.DataFrame(data)

# Menghitung korelasi antara Usia dan Jam Belajar
korelasi = df.corr()

print("Matriks Korelasi:")
print(korelasi)
```

`df.corr()`	Menghitung koefisien korelasi antara semua kolom numerik dalam DataFrame.	`df.corr()` menghasilkan matriks korelasi antar kolom numerik.
`df.cov()`	Menghitung kovarians antar kolom dalam DataFrame.	`df.cov()` menghasilkan matriks kovarians antar kolom numerik.
`df.value_counts()`	Menghitung jumlah nilai unik dalam suatu kolom.	`df['gender'].value_counts()` menghitung frekuensi nilai unik dalam kolom `gender`.
`df.pivot_table()`	Membuat pivot table untuk menganalisis data berdasarkan agregasi.	`df.pivot_table(values='age', index='gender', aggfunc='mean')` akan menghitung rata-rata usia berdasarkan gender.
`df.groupby()`	Mengelompokkan data berdasarkan satu atau lebih kolom dan menerapkan agregasi statistik.	`df.groupby('gender')['age'].mean()` menghitung rata-rata usia berdasarkan gender.
`df.quantile()`	Menghitung kuantil ke-n (misalnya kuartil) dari suatu kolom.	`df['age'].quantile(0.25)` menghasilkan kuartil pertama dari kolom `age`.

PANDAS

`df.sample()`	Mengambil sampel acak dari DataFrame.	`df.sample(5)` menghasilkan 5 sampel acak dari DataFrame.
`df.isna()`	Mengidentifikasi nilai yang hilang (missing values) dalam DataFrame.	`df.isna()` menghasilkan DataFrame boolean yang menunjukkan nilai yang hilang.
`df.fillna()`	Mengisi nilai yang hilang dengan nilai yang ditentukan.	`df['age'].fillna(df['age'].mean())` mengisi nilai hilang pada kolom `age` dengan rata-ratanya.
`df.random_state`	Menetapkan nilai acak pada operasi seperti sampling atau pemisahan data untuk reproduksi.	`df.sample(5, random_state=42)` memastikan pengambilan sampel yang dapat direplikasi.

Jika kita ingin mengetahui kuartil pertama (Q1), kuartil ketiga (Q3), dan persentil ke-90 dari data usia mahasiswa, kita bisa menggunakan fungsi quantile().

Code Example Hitung Quantile

```
Q1 = df['Usia'].quantile(0.25)
Q3 = df['Usia'].quantile(0.75)
persentil_90 =df['Usia'].quantile(0.90)
print("Kuartil 1:", Q1)
print("Kuartil 3:", Q3)
print("Persentil ke-90:", persentil_90)
```

MATPLOTLIB

Code Examples :

```
import matplotlib.pyplot as plt

# Data contoh
data = [2, 3, 1, 4, 3, 2, 4, 1, 3, 2]

# Membuat histogram
plt.hist(data)

# Menambahkan label dan judul
plt.xlabel('Nilai')
plt.ylabel('Frekuensi')
plt.title('Histogram Distribusi Nilai')

# Menampilkan grafik
plt.show()
```

Fungsi / Metode	Deskripsi	Contoh Penggunaan
`plt.plot()`	Membuat plot garis untuk data.	`plt.plot([1, 2, 3, 4], [10, 20, 25, 30])` menghasilkan plot garis antara dua array.
`plt.hist()`	Membuat histogram untuk visualisasi distribusi data.	`plt.hist([1, 2, 2, 3, 3, 3, 4], bins=4)` menghasilkan histogram dari data dengan 4 bin.
`plt.boxplot()`	Membuat diagram kotak (boxplot) untuk menggambarkan distribusi data dan outliers.	`plt.boxplot([1, 2, 3, 4, 5])` menghasilkan boxplot dari data tersebut.
`plt.scatter()`	Membuat plot sebar (scatter plot) untuk menggambarkan hubungan antar dua variabel.	`plt.scatter([1, 2, 3], [10, 20, 25])` menghasilkan scatter plot antara dua array.
`plt.bar()`	Membuat diagram batang untuk data kategorikal.	`plt.bar(['A', 'B', 'C'], [5, 7, 3])` menghasilkan diagram batang berdasarkan kategori.
`plt.errorbar()`	Membuat plot dengan garis error untuk menampilkan ketidakpastian atau variasi dalam data.	`plt.errorbar([1, 2, 3], [10, 20, 30], yerr=[1, 2, 1])` menghasilkan plot dengan error bar pada sumbu y.

MATPLOTLIB

Code Examples :

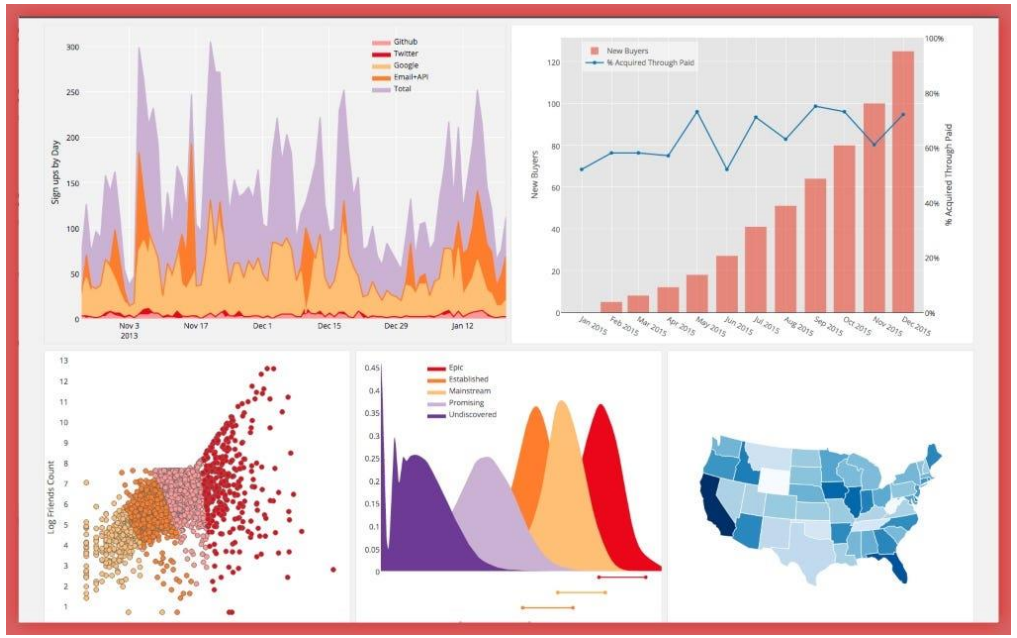
```
import matplotlib.pyplot as plt

# Data contoh
x = [1, 2, 3, 4, 5]
y = [2, 4, 6, 8, 10]

# Membuat plot garis
plt.plot(x, y)

# Menambahkan label dan judul
plt.xlabel('X')
plt.ylabel('Y')
plt.title('Plot Garis Hubungan X-Y')

# Menampilkan grafik
plt.show()
```

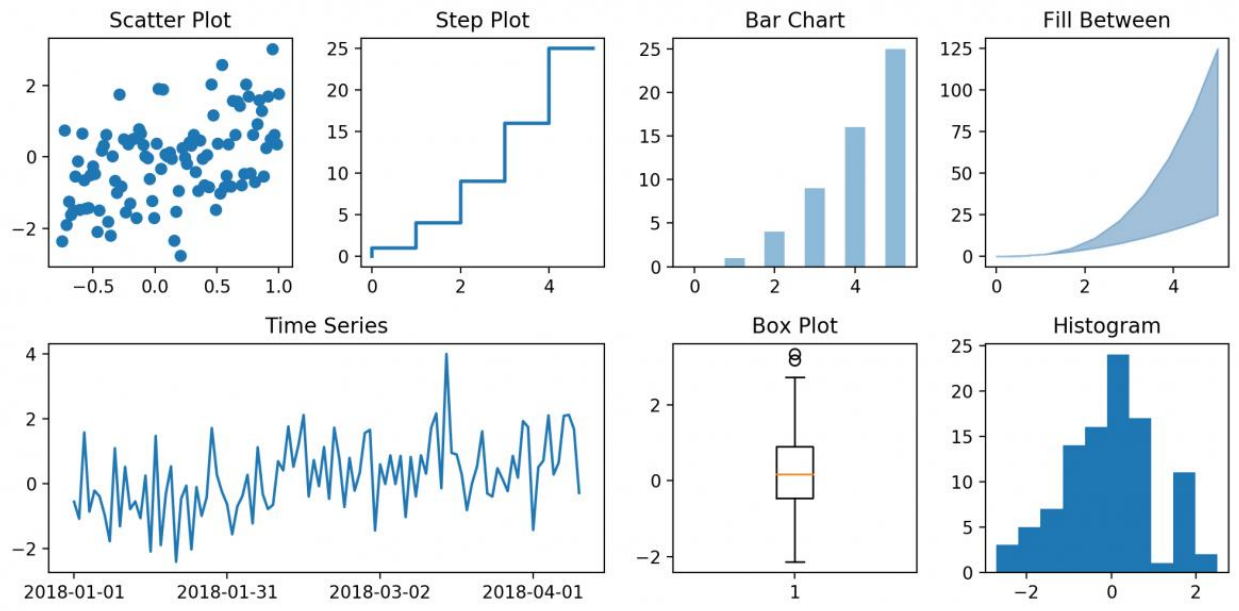
`plt.pie()`	Membuat diagram lingkaran (pie chart) untuk menggambarkan proporsi data.	`plt.pie([30, 30, 40], labels=['A', 'B', 'C'], autopct='%1.1f%%')` menghasilkan pie chart dengan tiga kategori.
`plt.subplot()`	Membuat beberapa subplot dalam satu figure untuk menampilkan beberapa grafik sekaligus.	`plt.subplot(1, 2, 1)` membuat subplot pertama dari dua, yang dapat digunakan untuk plot terpisah.
`plt.xlim()` **dan** `plt.ylim()`	Mengatur batas sumbu x dan y dalam plot.	`plt.xlim(0, 10); plt.ylim(0, 100)` mengatur batas sumbu x antara 0 dan 10 serta sumbu y antara 0 dan 100.
`plt.title()`	Menambahkan judul pada plot.	`plt.title("Distribusi Data Usia")` menambahkan judul pada plot.
`plt.grid()`	Menambahkan grid pada plot untuk mempermudah pembacaan data.	`plt.grid(True)` menambahkan grid pada plot yang sedang dibuat.
`plt.fill_between()`	Mengisi area antara dua kurva pada plot, biasanya digunakan untuk visualisasi distribusi.	`plt.fill_between([1, 2, 3], [10, 20, 15], [5, 10, 5], color='blue', alpha=0.3)` mengisi area antara dua garis dengan warna biru.

MATPLOTLIB

berguna dalam visualisasi data statistik dan probabilitas, membantu dalam menggambarkan distribusi data, hubungan antar variabel, dan memberikan representasi visual yang intuitif untuk analisis lebih lanjut.

`plt.legend()`	Menambahkan legenda pada plot untuk menunjukkan arti dari elemen yang ditampilkan.	`plt.plot([1, 2, 3], [10, 20, 25], label="Data 1"); plt.legend()` menambahkan legenda dengan label "Data 1".
`plt.barh()`	Membuat diagram batang horizontal untuk visualisasi data kategorikal.	`plt.barh(['A', 'B', 'C'], [5, 7, 3])` menghasilkan diagram batang horizontal berdasarkan kategori.





PLOT EXAMPLES

TUGAS

1. Diberikan data distribusi normal dengan mean 50 dan standar deviasi 5. Buatlah program Python untuk menghasilkan **100 nilai acak** dari distribusi normal ini dan hitung **mean**, **median**, dan **standar deviasi** dari data yang dihasilkan.
2. Diberikan DataFrame df dengan dua kolom x dan y yang memiliki data sebagai berikut:x: [5, 10, 15, 20, 25]y: [10, 20, 20, 30, 30]. Hitunglah mean, median, dan modus untuk setiap kolom x dan y menggunakan Pandas !
3. Visualisasikan soal nomor 1 dan 2 menggunakan Matplotlib !