

a. Import pandas

```
import pandas as pd
```

- **Mengapa?**

Pandas adalah library utama untuk manipulasi data tabular di Python.

- **Alias pd :**

Memperpendek penulisan, misal `pd.DataFrame` daripada `pandas.DataFrame`.

- **Tip:**

Selalu import di baris paling atas script/notebook agar konsisten.

b. Membuat DataFrame

```
df = pd.DataFrame(  
    [[1, 2, 3],  
     [4, 5, 6],  
     [7, 8, 9]],  
    columns=["A", "B", "C"],  
    index=["x", "y", "z"]  
)
```

- **Struktur list bersarang:**

- Setiap list di dalam mewakili satu baris (row).
- Elemen dalam list → nilai kolom.

- **columns=**

Menetapkan nama kolom. Jika diabaikan, Pandas beri nama otomatis `0,1,2...`.

- **index=**



Menetapkan label baris. Berguna untuk akses data berbasis label (misal `.loc["y"]`).

- **Kenapa DataFrame?**



- Mirip spreadsheet: mudah difilter, di-join, di-aggregate.
 - Landasan analisis data Python.
-

c. Melihat informasi dasar DataFrame



1. `df.info()`

-  Menampilkan ringkasan: jumlah entries, kolom, non-null count, tipe data, memori.
-  Berguna deteksi kolom dengan missing values.



2. `df.describe()`

-  Statistik ringkas kolom numerik: count, mean, std, min, 25/50/75%, max.
-  Untuk melihat distribusi dan outlier awal.



3. `df['A'].nunique()`

-  Hitung nilai unik di kolom A.
-  Untuk mengerti cardinality fitur (misal kategori vs numerik tinggi).

4. `df.shape`



-  Tuple (baris, kolom).
-  Cepat cek dimensi data.

5. `df.size`

-  Total elemen (baris × kolom).
-  Cek skala data sebelum operasi besar.

d. Menampilkan sebagian data

Misal DataFrame bernama `kopi` :

- `kopi.head(n)` → n baris pertama (default `n=5`).
- `kopi.tail(n)` → n baris terakhir (default `n=5`).
- `kopi.sample(n)` → n baris acak.
- **Kegunaan:**
 -  `head / tail` : cek struktur & format kolom.
 -  `sample` : validasi acak untuk cek inkonsistensi.

e. Seleksi & ubah data dengan `.loc` & `.iloc`

`.loc` – Label-based

- **Sintaks dasar:** `df.loc[row_labels, col_labels]`
- **Contoh:**

```
kopi.loc[[0,1,5]]           # baris label 0,1,5
kopi.loc[0:2]               # baris label 0 sampai 2 (inklusif)
kopi.loc[0:2, ["Hari","Tipe Kopi"]] # rows & spesifik kolom
```

- **Mengubah nilai:**

```
kopi.loc[1, "Jumlah Terjual (gelas)"] = 10
```

- **Catatan:**

- Label bisa string, tanggal, dsb.
- Range label pada `.loc` bersifat inklusif (termasuk ujung).

`.iloc` – Position-based

- **Sintaks dasar:** `df.iloc[row_positions, col_positions]`

- **Contoh:**

```
kopi.iloc[0:2, [0,2]] # baris ke-0 dan ke-1, kolom ke-0 & ke-2
```

- **Mengubah nilai:**

```
kopi.iloc[2, 1] = 15 # baris pos 2, kolom pos 1
```

- **Catatan:**

- Posisi 0-based.
- Slice `.iloc[a:b]` bersifat setengah terbuka ($a \leq \text{idx} < b$).

f. Akses sel tunggal: `.at`, `.iat` & atribut kolom

Method	Basis Seleksi	Keunggulan	Mirip dengan
<code>.at</code>	Label baris & kolom	Sangat cepat untuk 1 elemen	<code>df.loc[label, label]</code>
<code>.iat</code>	Posisi integer	Sangat cepat untuk 1 elemen	<code>df.iloc[pos, pos]</code>
<code>.col</code>	Atribut Python	Ringkas, tapi hanya kolom valid	<code>df["col"]</code>

Contoh nyata

```
# DataFrame contoh
df = pd.DataFrame(
    [[10,20],[30,40],[50,60]],
    columns=["X","Y"], index=["a","b","c"]
)

# .at (by label)
nilai_at = df.at["b","Y"]    # → 40
df.at["c","X"] = 55           # ubah sel ("c","X") jadi 55

# .iat (by posisi)
nilai_iat = df.iat[1,0]       # → 30 (baris pos 1, kol pos 0)
df.iat[2,1] = 65              # ubah sel (2,1) jadi 65

# atribut kolom
seri_x = df.X                 # sama seperti df["X"]
```

g. Mengurutkan DataFrame

- **Dasar:** `df.sort_values(by, ascending=True/False)`
- **Contoh sederhana:**

```
kopi.sort_values("Jumlah Terjual (gelas)") # menaik
kopi.sort_values("Jumlah Terjual (gelas)", ascending=False) # menurun
```

- **Multi-kolom:**

```
kopi.sort_values(
    ["Jumlah Terjual (gelas)", "Tipe Kopi"],
    ascending=[True, False]
)
```

- **Tip:**
 - Untuk mengurutkan `in-place`, tambahkan `inplace=True`.
 - Gunakan `na_position="first" / "last"` untuk mengatur posisi NaN.
-

🌟 Dengan penjelasan lebih rinci dan emoji, semoga tiap konsep jadi semakin **mudah dipahami** dan **aplikatif**! Jika ada contoh lain yang ingin didalami, beri tahu saja 😊