Pengenalan NumPy, Pandas, dan Matplotlib

Menguasai tiga library fundamental untuk analisis data dan komputasi ilmiah dalam Python

Mengapa Kita Butuh NumPy?

Python List memang fleksibel, tetapi memiliki keterbatasan serius untuk komputasi numerik. NumPy hadir sebagai solusi yang jauh lebih efisien dan powerful.

Kecepatan Luar Biasa

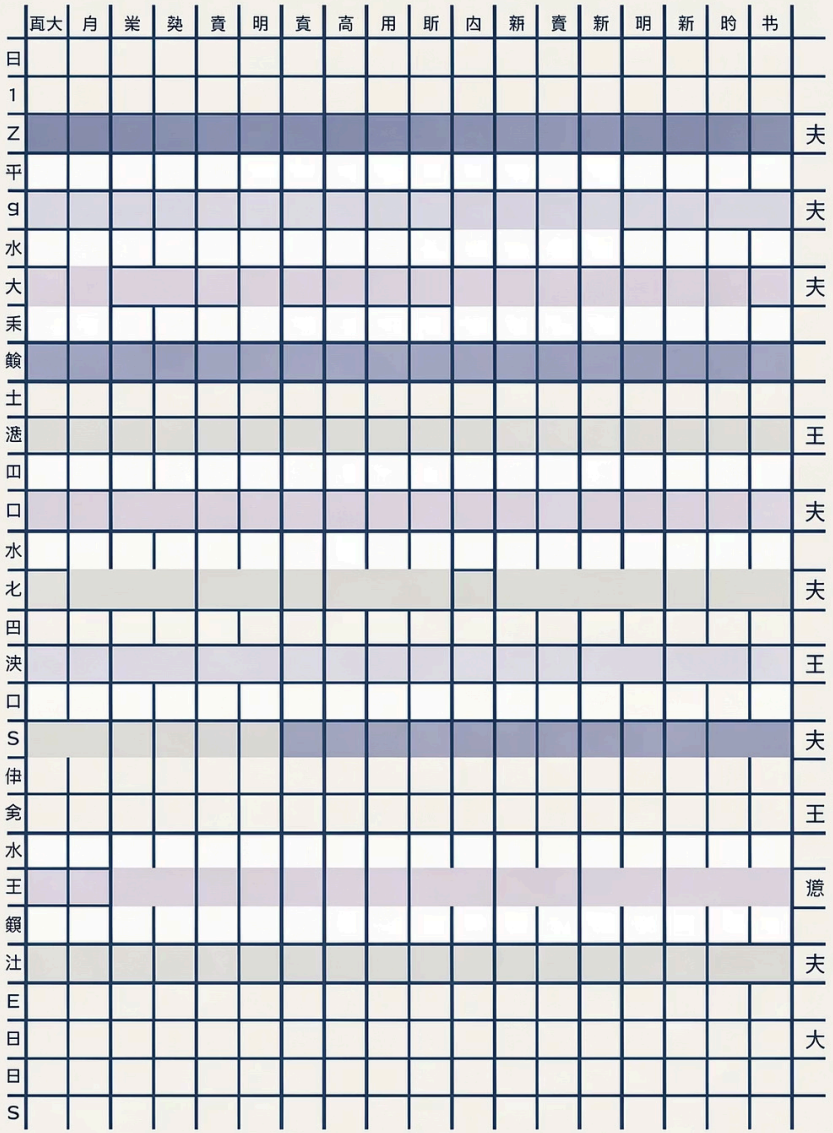
NumPy ditulis dalam bahasa C, menghasilkan performa 10-100x lebih cepat dibanding List Python biasa untuk operasi matematika.

Efisiensi Memori

Menggunakan blok memori yang padat dan terstruktur, menghemat RAM hingga 5-10x lipat dibanding List standar.

Vectorization Magic

Operasi matematika pada seluruh array sekaligus tanpa perlu looping manual. Kode lebih ringkas dan jauh lebih cepat!



Struktur Data NumPy: Ndarray

Ndarray adalah fondasi NumPy - array multi-dimensi yang menyimpan data dalam bentuk matriks terstruktur.

Axis 0 (Baris)

Berjalan secara **vertikal** dari atas ke bawah. Setiap baris adalah satu record atau observasi data.

- Operasi pada axis=0 bekerja per kolom
- Contoh: menghitung total setiap kolom

Axis 1 (Kolom)

Berjalan secara **horizontal** dari kiri ke kanan. Setiap kolom adalah satu fitur atau variabel.

- Operasi pada axis=1 bekerja per baris
- Contoh: menghitung rata-rata setiap baris

Praktik Dasar NumPy

Mari lihat operasi fundamental yang akan sering Anda gunakan dalam analisis data sehari-hari.

1

Membuat Array

```
import numpy as np
arr = np.array([1, 2, 3, 4, 5])
print(arr)
```

Fungsi dasar untuk mengubah List Python menjadi NumPy array yang siap diproses.

2

Vectorization

```
hasil = arr * 2
# Output: [2, 4, 6, 8, 10]
```

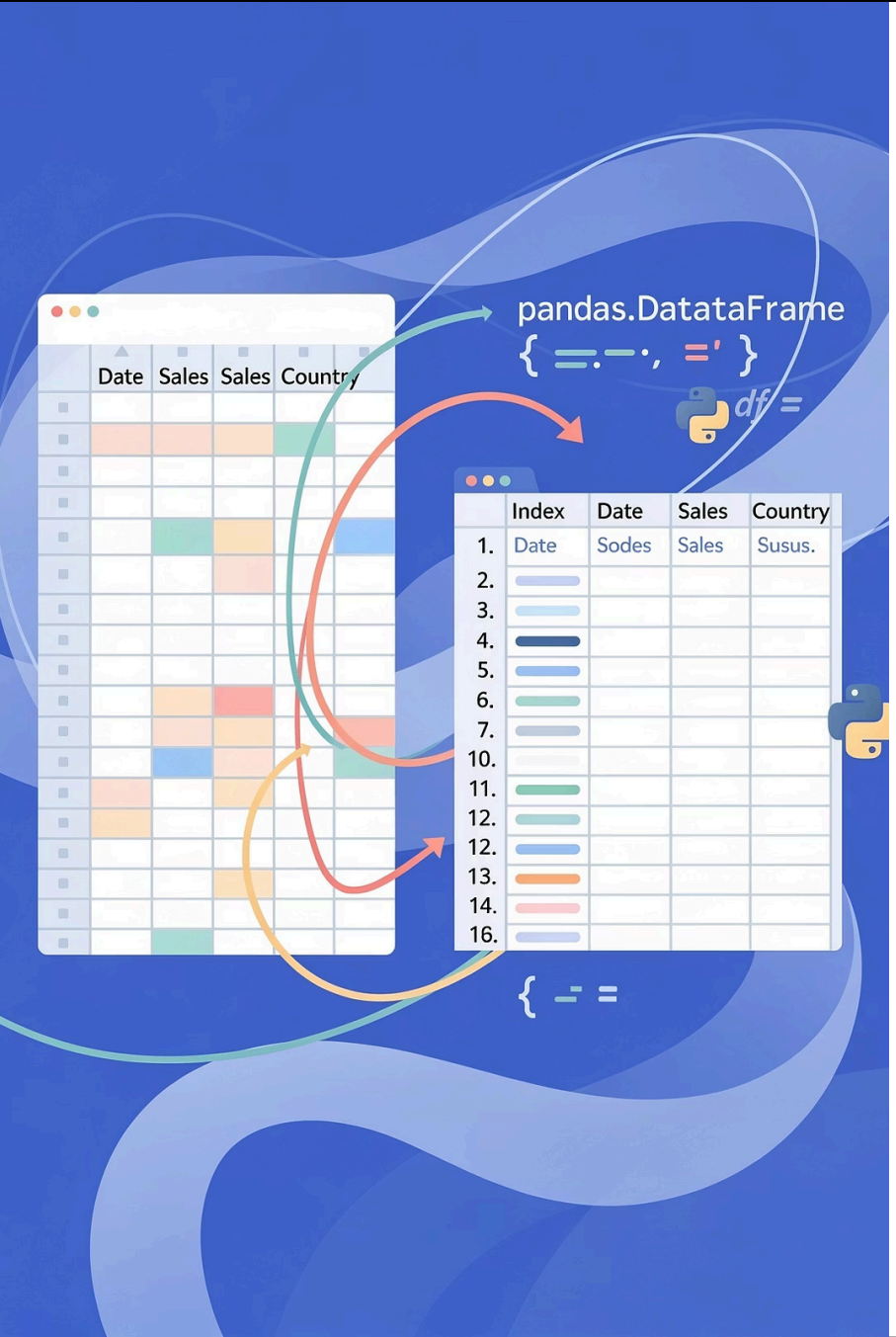
Semua elemen dikalikan sekaligus tanpa loop! Inilah kekuatan vectorization NumPy.

3

Statistik Instan

```
print(arr.sum())    # 15
print(arr.mean())   # 3.0
print(arr.max())    # 5
```

Fungsi statistik built-in yang langsung menghitung agregasi data dengan satu baris kode.



Pandas: Excel-nya Python

Pandas adalah library paling populer untuk manipulasi dan analisis data terstruktur. Bayangkan Excel, tapi dengan kekuatan penuh pemrograman Python!

Series

Array 1 dimensi dengan index. Seperti satu kolom dalam spreadsheet yang memiliki label untuk setiap barisnya.

DataFrame

Tabel 2 dimensi lengkap dengan baris dan kolom. Ini adalah struktur utama yang akan Anda gunakan untuk analisis data.

Fun Fact: Nama "Pandas" berasal dari "Panel Data" - istilah ekonometrik untuk dataset multi-dimensi!

Load & Inspect Data

Langkah pertama dalam setiap proyek data science: membaca dan memahami struktur dataset Anda.

Import & Load Data

```
import pandas as pd

df = pd.read_csv('data.csv')
```

Intip Data Cepat

```
df.head()   # 5 baris pertama
df.tail()   # 5 baris terakhir
df.shape    # (rows, cols)
```

Pemeriksaan Mendalam

```
df.info()
# Tipe data & nilai null

df.describe()
# Statistik: mean, std,
# min, max, quartiles
```

Fungsi `.describe()` memberikan ringkasan statistik lengkap dalam sekejap!

Filtering Data dengan Pandas

Salah satu operasi paling powerful: mengambil subset data berdasarkan kondisi tertentu, tanpa perlu looping manual.

01

Definisikan Kondisi

Tentukan kriteria filtering menggunakan operator perbandingan pada kolom DataFrame.

02

Terapkan Filter

Masukkan kondisi ke dalam bracket notation untuk mendapatkan baris yang memenuhi syarat.

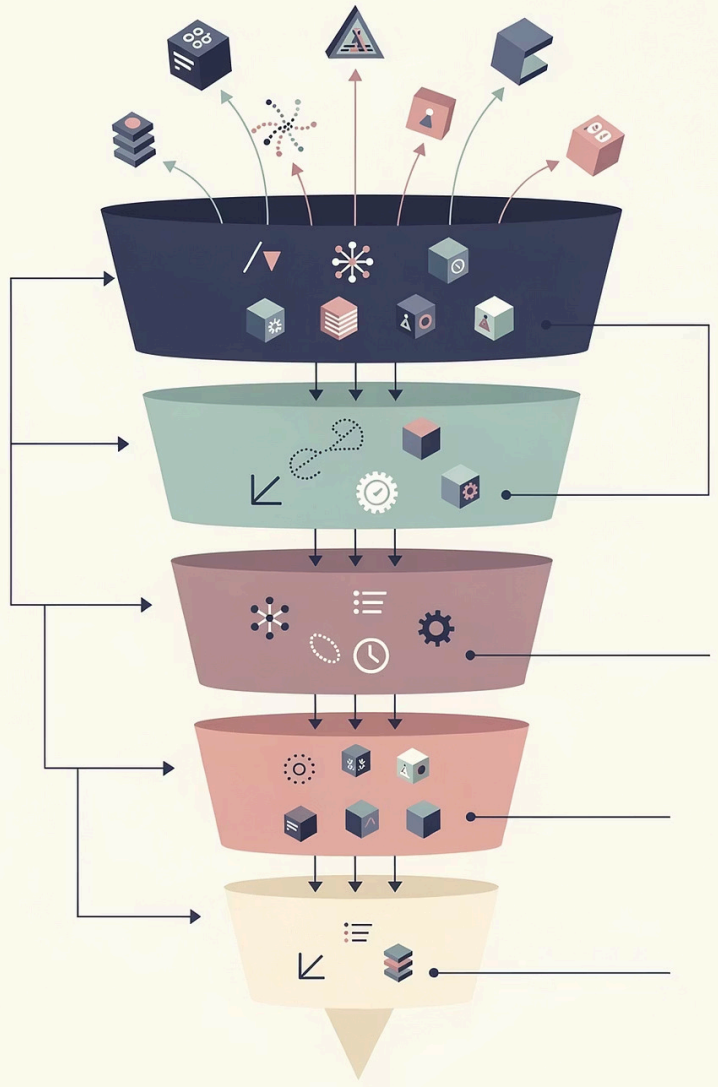
03

Dapatkan Hasil

DataFrame baru berisi hanya data yang sesuai dengan filter Anda.

Contoh Kasus: Mencari Mahasiswa Lulus

```
lulus = df[df['Nilai'] > 80]
print(f"Total mahasiswa lulus: {len(lulus)}")
```



Matplotlib: Line Chart untuk Tren

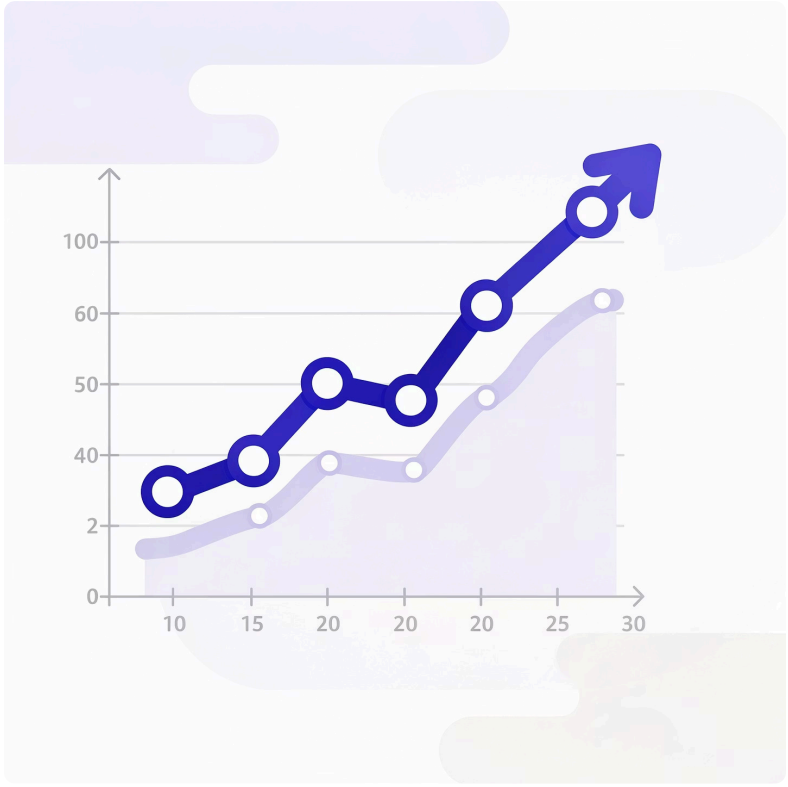
Kapan Menggunakan Line Chart?

- Menampilkan **perubahan data dari waktu ke waktu**
- Visualisasi tren dan pola temporal
- Membandingkan beberapa time series
- Data kontinyu seperti suhu, harga, atau traffic

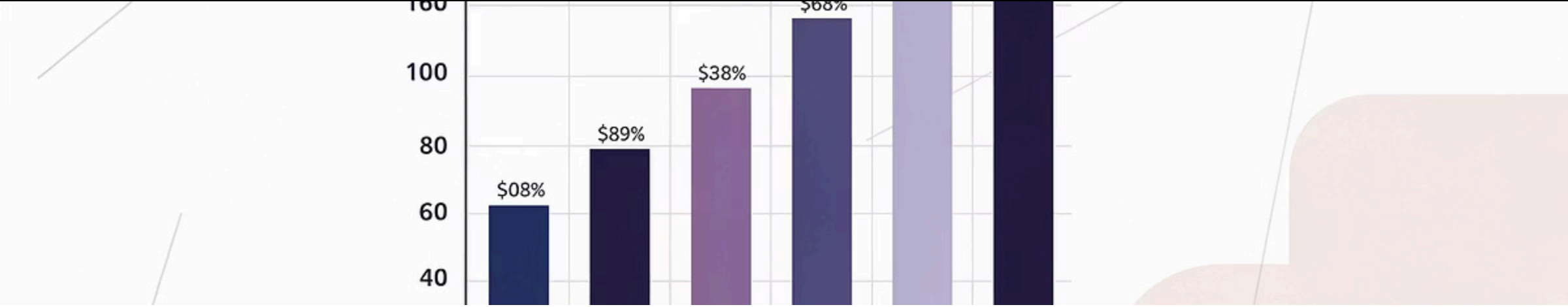
Kode Dasar

```
import matplotlib.pyplot as plt

plt.plot(x, y)
plt.xlabel('Waktu')
plt.ylabel('Nilai')
plt.title('Tren Data')
plt.show()
```



Line chart adalah pilihan terbaik untuk menceritakan bagaimana sesuatu berubah sepanjang periode tertentu.



Matplotlib: Bar Chart untuk Perbandingan

Fungsi Utama

Membandingkan **nilai antar kategori** yang berbeda secara visual dan mudah dipahami.

Use Cases

Jumlah penjualan per produk, distribusi gender, ranking nilai, atau perbandingan antar wilayah.

Implementasi

```
plt.bar(kategori, nilai)
plt.xticks(rotation=45)
plt.show()
```

Challenge Time!

Download Dataset

Ambil file CSV berisi data dari platform yang disediakan.

Analisis dengan Pandas

Gunakan Pandas untuk menemukan produk dengan total penjualan tertinggi bulan ini.

Visualisasi dengan Matplotlib

Buat bar chart yang menampilkan 10 produk terlaris beserta jumlah penjualannya.

> **Bonus Challenge:** Tambahkan line chart untuk menunjukkan tren penjualan harian produk terlaris! 🚀