Module & Package serta File Operation

Topik Pembelajaran

01

Modularisasi Kode

Memecah program besar menjadi bagian-bagian kecil yang terkelola dengan baik

02

Module & Package

Menggunakan dan membuat module serta package untuk reusability kode

03

File Operation

Menyimpan dan membaca data dari file untuk persistence

04

Studi Kasus

Implementasi nyata dengan batch processing data mahasiswa

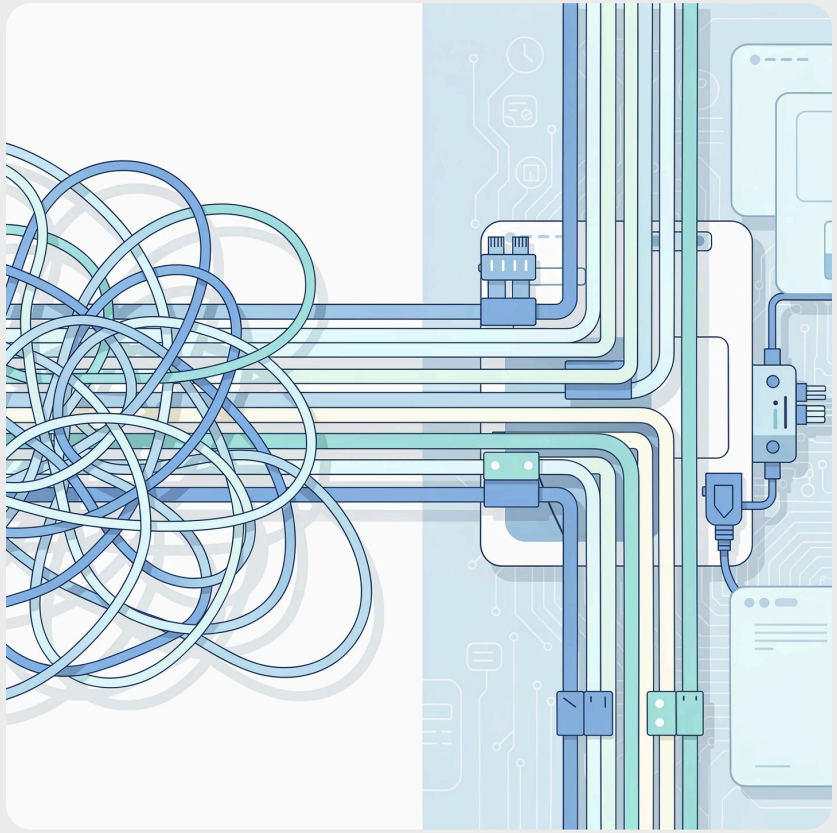
Konsep Modularisasi

Masalah: Kode dalam Satu File

- Susah dibaca dan dipahami
- Sulit di-debug ketika error
- Tidak efisien saat dikembangkan
- Kode berulang di mana-mana

Solusi: DRY Principle

Don't Repeat Yourself - Tulis kode sekali, gunakan berkali-kali



import pandas as pd

Cara Menggunakan Module

1

Basic Import

```
import my_module
my_module.fungsi()
```

Mengakses fungsi melalui nama module

2

Alias Import

```
import my_module as mm
mm.fungsi()
```

Mempersingkat penulisan nama module

3

Specific Import

```
from my_module import fungsi
fungsi()
```

Efisien memori, tapi hati-hati konflik nama

4

Import All

```
from my_module import *
```

Dianggap *bad practice* karena bisa menimpa fungsi lain

Standard Library Python

Module bawaan yang sering digunakan dalam algoritma dan pemrograman

math

Operasi matematika seperti `sqrt()`, `pi`, `ceil()`, `floor()` untuk perhitungan presisi tinggi

random

Simulasi angka acak dengan `random()`, `randint()`, `choice()` untuk testing dan game

datetime

Manipulasi waktu dan tanggal, menghitung durasi, formatting timestamp untuk aplikasi real-time

os & sys

Interaksi dengan sistem operasi, manajemen file dan direktori, akses environment variables

Package: Organisasi Tingkat Lanjut

Apa itu Package?

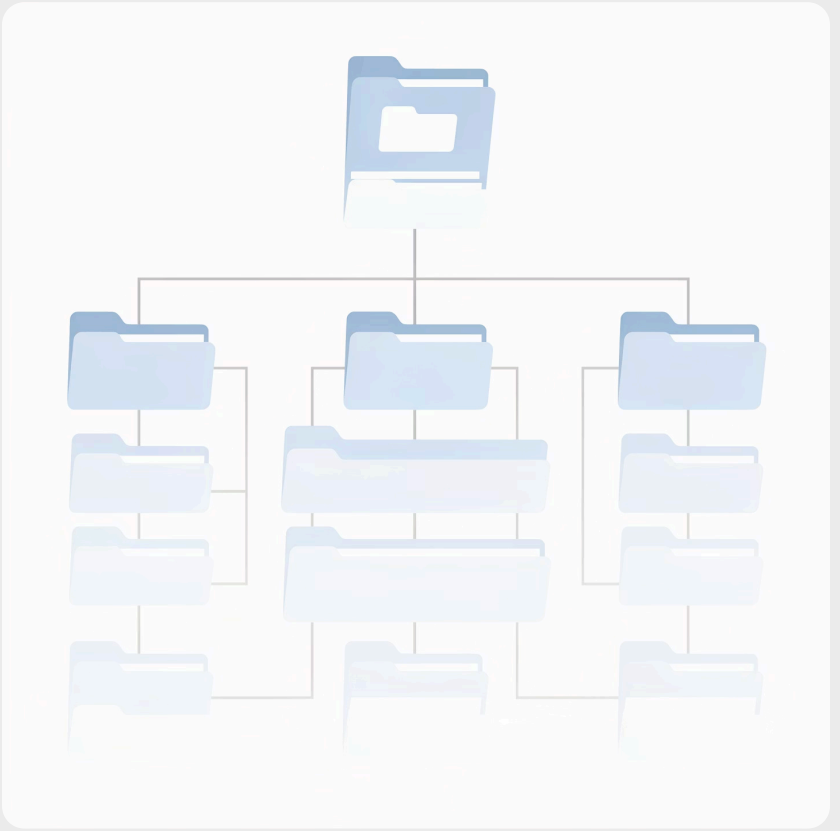
Kumpulan module dalam sebuah direktori yang membentuk hierarki terstruktur

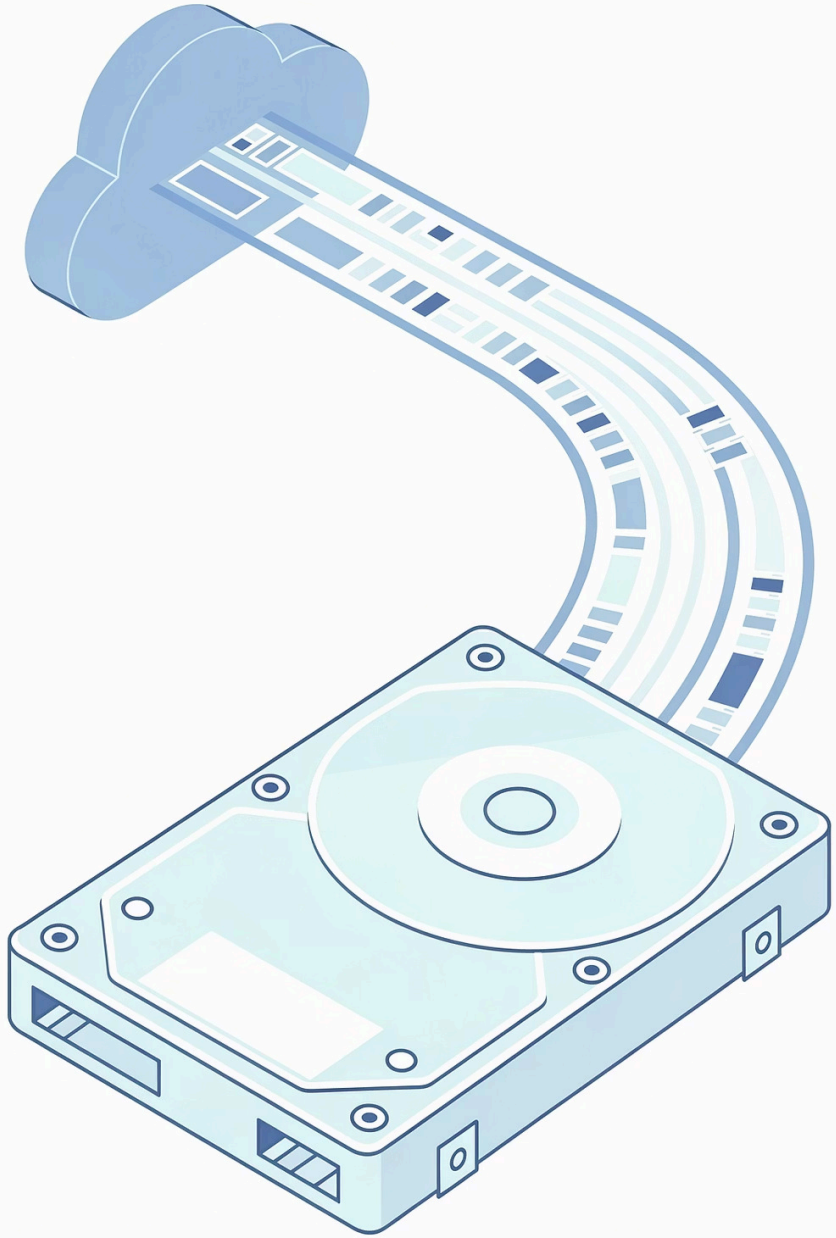
Struktur Package

```
aplikasi_toko/
├── __init__.py
├── kasir.py
└── gudang.py
```

Peran __init__.py

- Penanda direktori sebagai package
- Inisialisasi package saat di-import
- Kontrol namespace package





File Operation: Persistence Data

1. **RAM (Sementara)**
   Variabel hilang saat program ditutup
2. **Disk (Permanen)**
   Data tersimpan dalam file

Alur Kerja File Operation

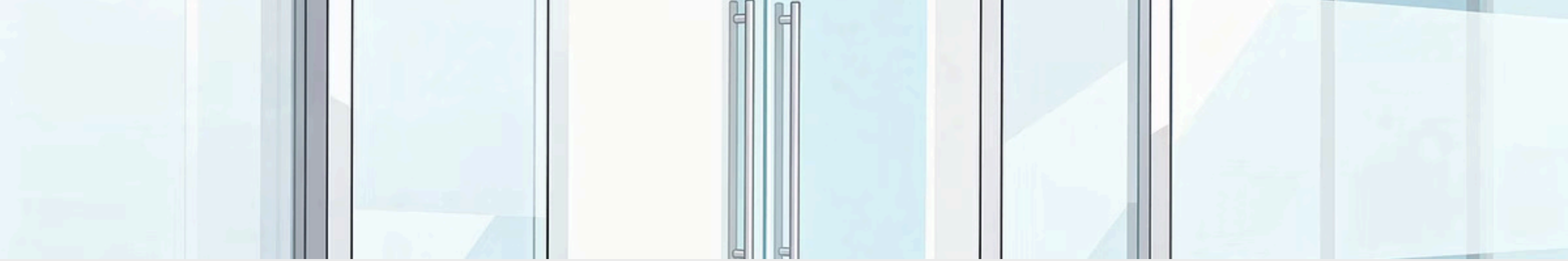
1. **Open**
   Buka file dengan mode tertentu
2. **Process**
   Baca atau tulis data
3. **Close**
   Tutup file untuk menyimpan perubahan

Mode Operasi File

Memahami mode file sangat penting agar data tidak sengaja tertimpa atau hilang

Mode	Deskripsi	File Tidak Ada	Isi File Lama
'r'	Read - Membaca file	Error/Exception	Aman (tidak berubah)
'w'	Write - Menulis file	Buat file baru	**Dihapus/Ditimpa**
'a'	Append - Menambah data	Buat file baru	Ditambah di akhir

Gunakan mode 'w' dengan hati-hati karena akan menghapus seluruh isi file yang sudah ada!



Best Practice: Context Manager

❌ Tanpa Context Manager

```
file = open('data.txt', 'r')
data = file.read()
file.close()  # Bisa lupa!
```

Masalah:

- Resource leak jika lupa `close()`
- Data bisa korup saat error
- File tetap terbuka di memori

✅ Dengan Context Manager

```
with open('data.txt', 'r') as file:
    data = file.read()
# Otomatis ditutup!
```

Keuntungan:

- File otomatis ditutup
- Aman meski terjadi error
- Kode lebih bersih dan pythonic

Studi Kasus: Sistem Rekap Nilai

Implementasi lengkap: Module + Package + File Operation

1

Input: data_nilai.txt

```
Budi,80
Siti,45
Joko,90
Rina,55
```

2

Module: statistik.py

Fungsi cek_kelulusan(nilai) dan konversi_huruf(nilai)

3

Process: main.py

Baca file → Split data → Panggil module → Proses logika

4

Output: laporan_akhir.txt

```
Budi - 80 - LULUS
Siti - 45 - REMEDIAL
Joko - 90 - LULUS
Rina - 55 - REMEDIAL
```

Pola Input → Process → Output adalah fondasi penting dalam pengembangan aplikasi berbasis data