KUSUMA WEB | EDU SERIES

MODUL 09: MANIPULASI FILE EKSTERNAL

MASTERCLASS PYTHON DASAR

Penyimpanan Persistensi Data melalui Manipulasi Berkas Lokal. Eksplorasi Aturan Mode Akses File `'w'`, `'r'`, `'a'`, Operasi Aman Menggunakan Blok Manajer Konteks `with open()`, serta Pengelolaan Log Aktivitas.

Kusuma Web Edu Series
Kelas Dasar Pemrograman — Modul 9 dari 10
Tanggal Rilis: 29 Juni 2026

Konsep Persistensi & Mode Akses Berkas

Sejauh ini, seluruh data variabel yang kita olah di dalam program bersifat sementara (*temporary*) di dalam RAM komputer. Begitu program kita matikan, seluruh data tersebut akan lenyap. Agar data dapat disimpan abadi, kita perlu menyimpannya ke dalam media penyimpanan fisik komputer menggunakan sistem **File I/O (Input/Output)**.

1. Apa itu Operasi File I/O?

Fungsionalitas dasar untuk berinteraksi dengan media penyimpanan eksternal, yang mencakup operasi pembuatan dokumen baru, membaca isi teks yang tersimpan, hingga menambah catatan baru di bagian akhir dokumen.

2. Aturan Tiga Mode Utama Akses Berkas

Saat membuka file menggunakan fungsi `open(filename, mode)`, kita harus menetapkan izin operasinya secara tegas:

- **Mode `'w'` (Write):** Membuka file untuk menulis data baru. Jika dokumen yang dituju belum ada, sistem akan langsung menciptakannya secara otomatis. Jika dokumen sudah ada, seluruh isi konten lama akan langsung ditimpa dan dihapus (*overwrite*).
- **Mode `'r'` (Read):** Membuka file untuk membaca isi konten yang sudah ada. Jika dokumen ternyata tidak ditemukan, sistem akan memicu error fatal `FileNotFoundError`.
- **Mode `'a'` (Append):** Menambahkan teks data baru di baris paling akhir file tanpa menghapus isi tulisan lama yang sudah tersimpan sebelumnya.

Operasi File Aman Menggunakan Manajer Konteks

Cara tradisional dalam membuka berkas di Python adalah dengan menggunakan sintaksis `f = open()` dan diakhiri dengan instruksi wajib `f.close()`. Namun, pendekatan klasik ini sangat berbahaya karena jika terjadi error di tengah program, instruksi penutupan file terlewati dan berkas akan terkunci di memori.

1. Praktik Terbaik: Blok `with open()`

Sangat disarankan menggunakan struktur Manajer Konteks (*Context Manager*) menggunakan kata kunci **`with`**. Metode ini menjamin bahwa berkas yang Anda buka **pasti akan ditutup secara aman** dari sistem oleh Python secara otomatis begitu eksekusi keluar dari blok instruksi, bahkan jika terjadi error sekalipun.

2. Contoh Praktik Menulis Dokumen Baru (`'w'`)

```
# Membuat dan menulis data ke berkas catatan_belajar.txt
with open("catatan_belajar.txt", "w") as file_saya:
    file_saya.write("Kusuma Web Edu Series - Python Masterclass.\n")
    file_saya.write("Saya sedang mempraktikkan materi File I/O hari ini.\n")

print("File berhasil ditulis dan ditutup secara otomatis!")
```

Membaca Berkas Secara Efisien & Aman

Membaca berkas teks berukuran sangat raksasa membutuhkan teknik khusus agar penggunaan memori RAM komputer Anda tidak melonjak tinggi secara tiba-tiba.

1. Tiga Metode Membaca Berkas di Python

- **`read()`:** Membaca seluruh isi berkas teks sekaligus sebagai satu string tunggal. Kurang disarankan untuk file berukuran gigabyte.
- **`readline()`:** Membaca berkas secara bertahap baris demi baris, hanya baris aktif yang diproses di memori RAM.
- **`readlines()`:** Membaca seluruh baris berkas dan mengonversinya menjadi kumpulan elemen di dalam tipe data `List`.

2. Contoh Kode Membaca File yang Tangguh

```
# Membaca berkas dengan penanganan error jika file hilang
try:
    with open("catatan_belajar.txt", "r") as file_baca:
        print("=== ISI BERKAS CATATAN ===")
        # Iterasi membaca baris demi baris secara hemat memori
        for baris in file_baca:
            # Menggunakan strip() untuk menghapus spasi/newline berlebih
            print(baris.strip())
        print("==========================")
except FileNotFoundError:
    print("[ERROR] Berkas target tidak ditemukan pada sistem penyimpanan.")
```

Studi Kasus: Sistem Pencatat Log Aktivitas Aplikasi

Mari kita bangun sebuah program nyata yang mengotomatisasi pencatatan rekam jejak (*log tracker*) aktivitas login pengguna ke dalam file log eksternal menggunakan mode penambahan konten (*Append*).

1. Alur Kerja Log Tracker

Setiap kali ada user melakukan interaksi di sistem, program akan otomatis menambahkan catatan teks baru yang merinci nama pelaku beserta timestamp waktu kejadiannya di akhir file `aktivitas.log`.

2. Kode Lengkap Program

```python
# Program Logger Aktivitas Pengguna Kusuma Web
import datetime # Library internal pelacak waktu

def catat_log(nama_user, status_aksi):
    # Mengambil format waktu saat ini
    waktu_sekarang = datetime.datetime.now().strftime("%Y-%m-%d %H:%M:%S")
    baris_log = f"[{waktu_sekarang}] User: {nama_user} | Aksi: {status_aksi}\n"

    # Membuka file dengan mode append 'a' agar tidak menghapus log lama
    with open("aktivitas.log", "a") as file_log:
        file_log.write(baris_log)

# Simulasi Berjalannya Aktivitas Program
print("=== LAYANAN LOGGER KUSUMA WEB AKTIF ===")
nama_input = input("Siapa nama Anda?: ")

print("Mencatat aktivitas login Anda...")
catat_log(nama_input, "LOGIN BERHASIL")

print("Mencatat aktivitas pengunduhan modul...")
catat_log(nama_input, "MENGUNDUH MODUL 09 PYTHON")

print("\nSukses! Silakan periksa file 'aktivitas.log' di folder Anda.")
```