KUSUMA WEB | EDU SERIES

MODUL 04: STRUKTUR KONTROL KEPUTUSAN

MASTERCLASS PYTHON DASAR

Algoritma Alur Logika Percabangan Kondisional di Python.
Eksplorasi Aturan Indentasi PEP 8, Kondisional Majemuk `if-elif-else`, Logika Bersarang, serta Penerapannya dalam Kasus Validasi Real-Time.

Kusuma Web Edu Series
Kelas Dasar Pemrograman — Modul 4 dari 10
Tanggal Rilis: 29 Juni 2026

Konsep Percabangan & Aturan Indentasi PEP 8

Pada pemrograman, alur eksekusi biasanya berjalan dari baris atas ke bawah secara berurutan. Namun, dalam aplikasi nyata kita sering kali memerlukan program untuk mengambil jalur keputusan berbeda berdasarkan kondisi yang ditemui.

1. Bagaimana Logika Percabangan Bekerja?

Sebuah kondisi akan dievaluasi dan menghasilkan tipe data `bool` (`True` atau `False`). Jika kondisi bernilai `True`, maka blok kode tertentu akan dijalankan; jika `False`, program akan melompati blok tersebut atau mengambil jalur alternatif.

2. Aturan Indentasi PEP 8

Di dalam Python, tanda kurung kurawal `{}` yang biasa dipakai di bahasa C++ atau Java ditiadakan. Sebagai gantinya, Python menggunakan **indentasi** (sejumlah spasi di awal baris) untuk mendefinisikan blok struktur program.

- Standar PEP 8 menetapkan penggunaan **4 spasi** (bukan tombol tab biasa) untuk indentasi.
- Ketidakkonsistenan indentasi akan langsung memicu error fatal `IndentationError`.

```
# Contoh Struktur Logika Sederhana
hari_hujan = True

if hari_hujan:
    print("Bawa payung saat keluar rumah!") # Bagian dari blok 'if'
print("Program selesai.") # Di luar blok 'if'
```

Menguasai Struktur Majemuk: `if-elif-else`

Ketika kita dihadapkan pada lebih dari dua opsi kondisi keputusan, menggunakan satu percabangan dasar tidaklah cukup. Python menyediakan struktur `elif` (kependekan dari *else-if*) untuk mengevaluasi serangkaian kondisi secara beruntun.

1. Sintaks Alur Evaluasi Majemuk

Sistem akan mengecek kondisi pertama. Jika bernilai `False`, sistem melompat mengecek kondisi `elif` di bawahnya. Jika seluruh kondisi tidak ada yang terpenuhi, maka blok `else` yang paling akhir akan dieksekusi secara otomatis sebagai jalur cadangan.

2. Contoh Implementasi Penilaian Karakter

```
# Evaluasi Skor Prestasi
skor = 78

if skor >= 90:
    predikat = "Istimewa"
elif skor >= 75:
    predikat = "Sangat Baik"
elif skor >= 60:
    predikat = "Cukup"
else:
    predikat = "Kurang"

print(f"Hasil Evaluasi Predikat Anda: {predikat}")
```

Aturan Penting Evaluasi

Interpreter Python mengevaluasi struktur `if-elif-else` dari atas ke bawah. Begitu salah satu kondisi bernilai `True` dan blok kodenya dieksekusi, Python akan langsung keluar dari seluruh rantai keputusan tersebut tanpa memeriksa sisa kondisi di bawahnya.

Percabangan Bersarang (*Nested IF*)

Percabangan bersarang (*Nested IF*) terjadi ketika suatu blok keputusan ditempatkan di dalam blok keputusan lainnya. Struktur ini digunakan untuk mengevaluasi kondisi bertingkat yang saling bergantung secara logis.

1. Kapan Harus Menggunakannya?

Gunakan percabangan bersarang saat keputusan kedua hanya bisa dinilai setelah prasyarat dari keputusan pertama berhasil terlampaui dengan sukses.

2. Contoh Logika Bersarang

```python
# Validasi Kelayakan Mengemudi
usia = 19
memiliki_sim = False

if usia >= 17:
    print("Usia Anda memenuhi persyaratan berkendara.")
    # Logika bersarang di dalam blok IF utama
    if memiliki_sim:
        print("Status Keamanan: Diperbolehkan Jalan.")
    else:
        print("Status Keamanan: Dilarang (Wajib mengurus SIM terlebih dahulu).
            ")
else:
    print("Status Keamanan: Dilarang (Usia belum mencukupi).")
```

Gunakan Secara Bijak

Meskipun *Nested IF* sangat kuat, penulisan tingkat bersarang yang terlalu dalam (lebih dari 3 tingkat) dapat merusak keterbacaan kode (*code readability*). Disarankan menggunakan operator logika seperti `and` untuk menyederhanakan logika bersarang jika memungkinkan.

Studi Kasus: Gerbang Keamanan Autentikasi Login

Mari kita bangun sebuah simulasi sistem login dinamis yang memvalidasi kredensial pengguna (*username* dan *password*) secara bertahap demi alasan keamanan.

1. Alur Logika Keamanan

- **Tahap 1:** Periksa apakah nama akun pengguna terdaftar pada basis data.
- **Tahap 2:** Jika akun terdaftar, lanjutkan untuk mencocokkan kata sandi (*password*).
- **Tahap 3:** Berikan pesan kesalahan yang spesifik agar mempermudah evaluasi keamanan.

2. Kode Lengkap Program

```
# Simulasi Sistem Autentikasi Kusuma Web
DB_USER = "admin_kusuma"
DB_PASS = "rahasia123"

print("=== PORTAL GERBANG UTAMA KUSUMA WEB ===")
input_user = input("Masukkan Username Anda: ")

# Validasi Tahap 1: Verifikasi Akun Pengguna
if input_user == DB_USER:
    print("Username ditemukan. Silakan isi sandi keamanan.")
    input_pass = input("Masukkan Password Anda: ")

    # Validasi Tahap 2: Verifikasi Sandi Akun
    if input_pass == DB_PASS:
        print("\n[SUKSES] Akses Diizinkan. Selamat datang admin!")
    else:
        print("\n[GAGAL] Password salah! Akses ditolak.")
else:
    print("\n[GAGAL] Username tidak terdaftar pada sistem kami.")
```