KUSUMA WEB | EDU SERIES

# MODUL 08: PENANGANAN ERROR & EKSEPSI

## MASTERCLASS PYTHON DASAR

Mencegah Aplikasi Crash dengan Mekanisme Defensif.
Eksplorasi Ragam Kesalahan Sistem, Penanganan Menggunakan Blok Komprehensif `try-except`, Penyelesaian Siklus `finally`, serta Teknik Pelemparan Error Secara Eksplisit.

**Kusuma Web Edu Series**
Kelas Dasar Pemrograman — Modul 8 dari 10
Tanggal Rilis: 29 Juni 2026

# Klasifikasi Jenis Error di Python

Kesalahan (*error*) di dunia pemrograman adalah hal wajar yang tidak bisa dihindari sepenuhnya. Secara garis besar, tipe kesalahan di Python terbagi ke dalam dua kelompok utama yang memiliki perilaku berbeda di dalam sistem.

## 1. Syntax Error (Kesalahan Tata Bahasa)

Terjadi saat Anda menuliskan kode yang melanggar aturan penulisan resmi bahasa Python (misalnya lupa menutup tanda kurung atau salah meletakkan tanda titik dua). Jenis error ini dideteksi sebelum program mulai dijalankan, sehingga program menolak beroperasi dari awal.

## 2. Exceptions / Runtime Error (Kesalahan Waktu Jalan)

Terjadi saat tata bahasa kode Anda sudah benar, namun muncul kegagalan internal saat instruksi program sedang aktif dijalankan di komputer (misalnya mencoba membagi angka dengan nol, atau mencoba mengakses file yang tidak ada). Hal ini menyebabkan aplikasi mati secara mendadak (*crash*).

```
# Contoh Syntax Error (Gagal di awal)
# if True print("Halo") -> Kurang titik dua ':'

# Contoh Exception Error (Crash saat jalan)
angka = 10
# hasil = angka / 0 -> Memicu ZeroDivisionError!
```

# Mekanisme Blok Defensif: `try-except`

Agar aplikasi yang kita bangun tetap tangguh dan tidak mengalami *crash* saat menemui kegagalan tidak terduga, kita harus menangkap kegagalan tersebut menggunakan blok **`try-except`**.

## 1. Cara Kerja Deteksi Eksepsi

- **Blok `try`:** Kita menaruh baris kode yang kita curigai berpotensi memicu kegagalan di dalam blok ini untuk dipantau secara ketat.
- **Blok `except`:** Jika error benar-benar terjadi pada blok `try`, program tidak akan mati. Eksekusi langsung dialihkan ke dalam blok `except` ini untuk ditangani secara damai.

## 2. Contoh Implementasi Penanganan Error Pembagian

```
try:
    pembilang = 100
    penyebut = int(input("Masukkan angka pembagi: "))
    hasil = pembilang / penyebut
    print(f"Hasil pembagian sukses: {hasil}")
except ZeroDivisionError:
    print("[ERROR] Masukan salah! Anda tidak boleh membagi angka dengan nol.")
except ValueError:
    print("[ERROR] Harap hanya memasukkan karakter angka bulat numerik!")
```

# Siklus Penutup `finally` & Pelemparan Error

Untuk memastikan manajemen memori komputer berjalan bersih dan efisien, Python menyediakan alur penyelesaian terstruktur yang menjamin proses pembersihan aset aplikasi dilakukan tanpa celah.

## 1. Peran Penting Blok `finally`

Blok `finally` diletakkan di baris paling bawah setelah blok `except`. Baris kode di dalam blok `finally` ini dijamin **pasti akan selalu dijalankan** oleh komputer, baik ketika program berjalan lancar tanpa hambatan, maupun ketika program mengalami error di tengah proses. Sangat ideal untuk meletakkan kode pembersih aset seperti menutup file eksternal atau memutus koneksi database.

## 2. Pelemparan Error Secara Sengaja (`raise`)

Terkadang, demi alasan penegakan logika bisnis, kita perlu memicu timbulnya error secara sengaja menggunakan kata kunci `raise`.

## 3. Contoh Penerapan Kode

```
def validasi_usia(umur):
    if umur < 0:
        # Melempar error ValueError secara sengaja
        raise ValueError("Usia tidak boleh bertanda minus!")
    return f"Pendaftaran berhasil untuk usia: {umur}"

try:
    print(validasi_usia(-5))
except ValueError as e:
    print(f"[VALIDASI GAGAL] {e}")
```

# Studi Kasus: Sistem Validasi Masukan Pendaftaran

Mari kita bangun sebuah program interaktif pendaftaran akun belajar di Kusuma Web yang memiliki sistem validasi input defensif agar bebas dari ancaman kegagalan eksekusi (*crash*).

## 1. Aturan Validasi Keamanan

- Input nomor telepon pendaftar wajib berupa karakter angka bulat (`int`).
- Jika masukan mengandung huruf, sistem akan memicu `ValueError` dan mengulang form pendaftaran dengan aman tanpa menghentikan aplikasi secara paksa.

## 2. Kode Lengkap Program

```
# Form Registrasi Kusuma Web Bebas Crash
while True:
    print("\n=== FORM REGISTRASI ANGGOTA KUSUMA WEB ===")
    nama = input("Masukkan Nama Anda: ")

    try:
        # Mencoba melakukan casting ke integer yang rentan error ValueError
        telp = int(input("Masukkan Nomor Telepon Anda (hanya angka): "))
    except ValueError:
        print("\n[PERINGATAN GAGAL] Format nomor telepon salah! Harap tidak
            memasukkan huruf.")
        print("Sistem akan merestart form pendaftaran kembali...")
        print("----------------------------------------")
        continue # Melompati sisa kode, kembali ke atas form

    # Dijalankan hanya jika blok try sukses sepenuhnya
    print("\n----------------------------------------")
    print(f"Selamat {nama}! Pendaftaran Akun Anda Berhasil.")
    print(f"Kontak Terdaftar: {telp}")
    print("========================================")
    break # Keluar dari perulangan
```