

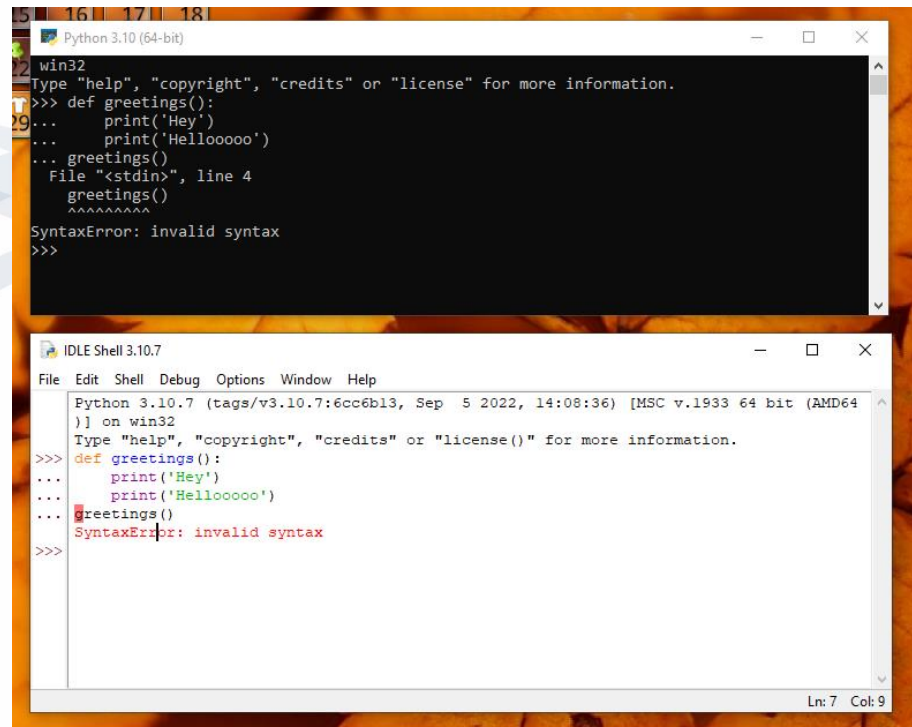
Handling Error

Chapter 14

Syaeful Anas Aklani, M.Kom

Pengertian Penanganan Kesalahan

Penanganan kesalahan adalah proses menangani situasi di mana program mengalami kesalahan atau pengecualian (exception) saat dijalankan. Ini penting untuk menjaga agar program tidak berhenti secara tiba-tiba dan memberikan umpan balik yang berguna kepada pengguna.



The image shows two screenshots of a Python IDLE environment. The top screenshot is a terminal window titled 'Python 3.10 (64-bit)' showing a SyntaxError: invalid syntax. The bottom screenshot is the IDLE Shell window titled 'IDLE Shell 3.10.7' showing the same code and error message.

```
Python 3.10 (64-bit)
win32
Type "help", "copyright", "credits" or "license" for more information.
>>> def greetings():
...     print('Hey')
...     print('Hellooooo')
... greetings()
File "<stdin>", line 4
    greetings()
    ^^^^^^^^^
SyntaxError: invalid syntax
>>>
```

```
IDLE Shell 3.10.7
File Edit Shell Debug Options Window Help
Python 3.10.7 (tags/v3.10.7:6cc6b13, Sep 5 2022, 14:08:36) [MSC v.1933 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> def greetings():
...     print('Hey')
...     print('Hellooooo')
... greetings()
SyntaxError: invalid syntax
>>>
```

SyntaxError: Kesalahan dalam sintaksis kode.

TypeError: Kesalahan yang terjadi ketika operasi dilakukan pada tipe data yang tidak sesuai.

ValueError: Kesalahan yang terjadi ketika fungsi menerima argumen dengan tipe yang benar tetapi nilai yang tidak sesuai.

IndexError: Kesalahan yang terjadi ketika mencoba mengakses indeks yang tidak ada dalam list.

KeyError: Kesalahan yang terjadi ketika mencoba mengakses kunci yang tidak ada dalam dictionary.

FileNotFoundError: Kesalahan yang terjadi ketika mencoba membuka file yang tidak ada.

Mekanisme Penanganan Kesalahan

Blok try digunakan untuk mengeksekusi kode yang mungkin menyebabkan kesalahan. Jika terjadi kesalahan, eksekusi akan berpindah ke blok except.

```
def main():  
    while True:  
        try:  
            x = int(input("Masukkan angka (atau ketik 'exit' untuk keluar): "))  
            hasil = 10 / x  
            print("Hasil: ", hasil)  
        except ValueError:  
            print("Input tidak valid! Harap masukkan angka.")  
        except ZeroDivisionError:  
            print("Kesalahan: Pembagian dengan nol tidak diperbolehkan.")  
        except Exception as e:  
            print("Terjadi kesalahan: ", e)  
        keluar = input("Apakah Anda ingin keluar? (y/n): ")  
        if keluar.lower() == 'y':  
            print("Terima kasih! Program selesai.")  
            break  
  
if __name__ == "__main__":  
    main()
```

```
Masukkan angka (atau ketik 'exit' untuk keluar): 1  
Hasil: 10.0  
Apakah Anda ingin keluar? (y/n): y  
Terima kasih! Program selesai.
```

Membuat Exception Kustom

dapat membuat exception kustom dengan mewarisi dari kelas Exception.

```
class MyCustomError(Exception):  
    def __init__(self, message):  
        self.message = message  
        super().__init__(self.message)  
def risky_function():  
    raise MyCustomError("Ini adalah kesalahan kustom!")  
  
try:  
    risky_function()  
except MyCustomError as e:  
    print(f"Terjadi kesalahan: {e.message}")
```

Terjadi kesalahan: Ini adalah kesalahan kustom!

Else

Blok else dapat digunakan setelah blok except.

Kode dalam blok else akan dijalankan jika tidak ada kesalahan yang terjadi dalam blok try.

```
try:
    numerator = float(input("Masukkan pembilang: "))
    denominator = float(input("Masukkan penyebut: "))
    result = numerator / denominator
except ZeroDivisionError:
    print("Kesalahan: Tidak bisa membagi dengan nol!")
except ValueError:
    print("Kesalahan: Harap masukkan angka yang valid!")
else:
    print(f"Hasil: {result}")
```

```
Masukkan pembilang: 10
Masukkan penyebut: 3
Hasil: 3.3333333333333335
```


Blok finally

Blok finally akan selalu dijalankan, terlepas dari apakah terjadi kesalahan atau tidak. Ini berguna untuk mengeksekusi kode pembersihan, seperti menutup file atau koneksi.

```
try:
    file = open("files.txt", "r")
    content = file.read()
except FileNotFoundError:
    print("File tidak ditemukan!")
finally:
    file.close()
```

File tidak ditemukan!

Menangkap Beberapa Exception

Anda dapat menangkap beberapa jenis exception dengan menggunakan beberapa blok except.

```
try:
    value = int(input("Masukkan angka: "))
    result = 10 / value
except ValueError:
    print("Input tidak valid! Harap masukkan angka.")
except ZeroDivisionError:
    print("Tidak bisa membagi dengan nol!")
```

```
Masukkan angka: saya
Input tidak valid! Harap masukkan angka.
```


Membuat Exception Kustom

Seperti yang telah dijelaskan sebelumnya, Anda dapat membuat exception kustom dengan mewarisi dari kelas Exception.

```
class MyCustomError(Exception):  
    pass  
  
try:  
    raise MyCustomError("Ini adalah kesalahan")  
except MyCustomError as e:  
    print(f"Terjadi kesalahan: {e}")
```

Terjadi kesalahan: Ini adalah kesalahan

Menggunakan raise

raise adalah pernyataan yang digunakan untuk melempar (atau "mengangkat") sebuah exception. Ini memungkinkan Anda untuk menghasilkan kesalahan secara manual dalam kode Anda, yang dapat digunakan untuk menunjukkan bahwa suatu kondisi tertentu telah terjadi yang tidak dapat ditangani dengan cara biasa.

Anda dapat menggunakan **raise** untuk melempar exception secara manual.

```
def check_positive(number):  
    if number < 0:  
        raise ValueError("Angka harus positif!")  
  
try:  
    check_positive(-5)  
except ValueError as e:  
    print(e)
```

Angka harus positif!

Latihan

1. Soal satu buatlah fungsi untuk argument a dan b, untuk memeriksa apakah bilangan integer, jika tidak terdapat notifikasi = 'input harus berupa angka dan jika $b \neq 0$, tampilan pembagian dengan nol tidak boleh
2. Buatlah aplikasi looping dengan error handle di bawah ini :

```
Masukkan angka: -5
Output: Angka tidak boleh negatif.
Masukkan angka: abc
Output: Input tidak valid. Harap masukkan angka.
Masukkan angka: 2
Output:
2 x 1 = 2
2 x 2 = 4
2 x 3 = 6
2 x 4 = 8
2 x 5 = 10
2 x 6 = 12
2 x 7 = 14
2 x 8 = 16
2 x 9 = 18
2 x 10 = 20
```

Latihan

3. Buatlah aplikasi parkir dengan eror handle jika Motor: Rp2.000 per jam, Mobil: Rp5.000 per jam dan Bus: Rp10.000 per jam (kurang dari 10 menit parkir gratis)

```
Masukkan jenis kendaraan (motor/mobil/bus): bus
Masukkan durasi parkir (jam): e
Output: Durasi parkir harus berupa angka positif.
Masukkan jenis kendaraan (motor/mobil/bus): mobil
Masukkan durasi parkir (jam): 2
Output: Total tarif parkir: Rp10,000
```