Modul Praktikum

Algoritma Pemrograman Dasar

10

"Error Handling dan File Eksternal"

10.1.1. Tujuan

- Mahasiswa mampu memahami kegunaan dari error
- Mahasiswa mampu menangani error yang muncul
- Mahasiswa mampu menyimpan data secara non-temporer
- Mahasiswa mampu menerapkan error handling pada program yang dibuat
- Mahasiswa mampu memahami sifat file serta struktur file dan folder pada komputer

10.1.2. Penjelasan

Pernahkah kalian terkejut ketika program yang telah kita buat mengalami error sehingga tidak berjalan sebagaimana mestinya dan juga apakah kalian pernah penasaran bagaimana caranya kita menyimpan data secara permanen dan tidak langsung hilang ketika program telah terhenti. Pada modul kali ini kita akan membahas bagaimana cara mengatasi serta memanfaatkan apa yang dinamakan error dan membuat file yang dapat menampung data baik itu input maupun output pada suatu program.

10.1.3. Jenis - Jenis Error

Nama error dapat muncul baik pada terminal maupun pada saat kita mengarahkan cursor pada sebuah keyword pada IDE/Code Editor, Error yang terdapat pada python adalah:

Nama Error	**Deskripsi**
SyntaxError	Terjadi ketika interpreter menemui sintaks yang tidak valid.
IndexError	Terjadi ketika kita salah memberikan index untuk mengambil data dari list
AssertionError	Terjadi ketika kita ingin melakukan pengecekan
AttributeError	Terjadi ketika melakukan operasi yang

	tidak sesuai pada suatu data
ImportError	Terjadi jika module yang ingin kita import tidak ditemukan
KeyError	Terjadi jika key pada dictionary tidak ditemukan
NameError	Terjadi jika sebuah variable belum diinisialisasi
MemoryError	Terjadi jika sebuah program kehabisan memory untuk digunakan
TypeError	Terjadi jika sebuah fungsi dan operasi tidak digunakan sebagaimana mestinya
IndentationError	Terjadi saat tab atau spasi dalam kode tidak mengikuti pola yang diharapkan.
FileNotFoundError	Terjadi ketika file tidak ditemukan

10.1.4. Mengetahui Error yang dialami

Ketika kita dihadapkan dengan error, kita perlu mengetahui jenis error apa yang terjadi dan di mana error itu terjadi.

Contoh:

```
angka = int(input("Masukkan angka: "))
print(angka)

#input “Hello”
```

Error yang didapat:

```
Acer@LAPTOP-72AGSKBS E:\Praktikum APD > py .\main.py
Masukkan angka: Hello
Traceback (most recent call last):
  File "E:\Praktikum APD\main.py", line 1, in <module>
    angka = int(input("Masukkan angka: "))
            ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
ValueError: invalid literal for int() with base 10: 'Hello'
```

Hal yang pertama dilakukan ketika mendapat error, perhatikan error terjadi di **file** apa dan di **line** berapa, selanjutnya identifikasi error apa yang didapat.

Pada kasus di atas, terjadi error di file **main.py** pada baris ke **1**, dan tipe error berupa **ValueError.**

10.1.5. Pengenalan Try-Except

Dalam pengembangan perangkat lunak, Try-Except adalah mekanisme yang penting untuk menangani exception atau kesalahan yang mungkin terjadi selama eksekusi kode. Blok Try digunakan untuk mengelompokkan potongan kode yang mungkin menyebabkan exception, sementara blok Except memberikan penanganan khusus untuk setiap jenis exception yang mungkin muncul.

10.1.6. Penggunaan Dasar Try-Except

Contoh sederhana penggunaan Try-Except adalah ketika kita mengisi variable integer dengan string. Dengan Try-Except, kita dapat mengantisipasi **ValueError.**

```
try:
    angka = int(input("Masukkan angka: "))
except ValueError:
    print("Input yang anda masukkan bukan angka")
```

Blok Try akan mencoba mengeksekusi kode di dalamnya, dan jika terjadi kesalahan/error di dalam blok Try maka, blok kode Except akan dieksekusi. Sehingga program akan tetap berjalan dan tidak berhenti.

```
Acer@LAPTOP-72AGSKBS E:\Praktikum APD > py .\main.py
Masukkan angka: Hello
Input yang anda masukkan bukan angka
```

10.1.7. Penggunaan Else dalam Blok Try-Except

Blok Else digunakan untuk menentukan kode yang akan dieksekusi jika tidak ada blok Except yang dieksekusi artinya di dalam blok Try tidak ada terjadi error.

```
try:
    angka = int(input("Masukkan angka: "))
except ValueError:
    print("Input yang anda masukkan bukan angka")
else:
    print(f"Angka yang kamu input: {angka}")
```

```
Acer@LAPTOP-72AGSKBS E:\Praktikum APD > py .\main.py
Masukkan angka: 24
Angka yang kamu input: 24
```

10.1.8. Penggunaan Finaly dalam Blok Try-Except

Blok Finaly dalam Try-Except pasti akan dijalankan mau ada error atau tidak ada di dalam blok Try pasti akan dijalankan.

```
try:
    angka = int(input("Masukkan angka: "))
except ValueError:
    print("Input yang anda masukkan bukan angka")
else:
    print(f"Angka yang kamu input: {angka}")
finally:
    print("Program selesai")
```

Contoh Try-Except Error:

```
Acer@LAPTOP-72AGSKBS E:\Praktikum APD > py .\main.py
Masukkan angka: Hello
Input yang anda masukkan bukan angka
Program selesai
```

Contoh Try-Except tidak Error:

```
Acer@LAPTOP-72AGSKBS E:\Praktikum APD > py .\main.py
Masukkan angka: 24
Angka yang kamu input: 24
Program selesai
```

10.1.9. Membuat Kustom Error dengan Raise

Keyword `raise` memungkinkan kita membuat error secara manual. Ini berguna ketika kita ingin mengindikasikan suatu kondisi khusus yang memerlukan perhatian atau penanganan tambahan. Dengan menggunakan raise, kita dapat membuat kode lebih responsif terhadap situasi tertentu.

Contoh:

```
try:
    nama = input("Hello, what's your name? ")
    if len(nama) > 5:
        raise ValueError("Nama tidak boleh lebih dari 5
karakter")
except ValueError as e:
    print(e)
```

Output:

```
Acer@LAPTOP-72AGSKBS E:\Praktikum APD > py .\main.py
Hello, what's your name? StevenSyifai
Nama tidak boleh lebih dari 5 karakter
```

Penjelasan:

- Pada blok `try` terdapat variable nama dengan nilai dari input user.
- Variable `nama` dicek apakah panjangnya lebih dari 5, jika true maka buat error manual dengan menggunakan keyword `raise` dengan error bertipe `ValueError` dengan pesan “Nama tidak boleh lebih dari 5 karakter”.
- Kemudian pesan error tersebut ditangkap oleh blok `except` dengan exception ValueError, dan tampilkan pesan errornya.

"File Eksternal"

10.2.1. Tujuan:

- Mahasiswa mampu menyimpan data secara non-temporer.
- Mahasiswa mampu memahami sifat file serta struktur file dan folder pada komputer.
- Mampu membuka, membaca, menulis, dan menutup file menggunakan Python.
- Menguasai berbagai mode akses file, seperti read, write, append, dan sebagainya.
- Memahami cara bekerja dengan file CSV.

10.2.2. Pengenalan File Eksternal

a. Pengertian File Eksternal

File eksternal adalah file yang disimpan di luar program utama. Berbeda dengan data yang disimpan di memori sementara (RAM), data dalam file eksternal disimpan secara permanen di media seperti hard drive atau SSD. File eksternal digunakan untuk menyimpan data yang perlu diakses kembali setelah program ditutup.

b. Jenis-Jenis File Eksternal

1. File Teks: File yang berisi data dalam format teks. Contoh: .txt, .csv.
2. File Biner: File yang menyimpan data dalam format biner. Contoh: .bin, .jpg, .png.

10.2.3. Operasi Dasar Pada File

a. Membuka File

Sebelum kita dapat membaca atau menulis ke file, kita harus membukanya terlebih dahulu. Di Python, kita menggunakan fungsi open() untuk membuka file.

```
# Membuka file untuk dibaca
file = open('nama_file.txt', 'r')

# Membuka file untuk ditulis
file = open('nama_file.txt', 'w')

# Membuka file untuk menambah data
file = open('nama_file.txt', 'a')
```

Mode yang sering digunakan

- "r": Membuka file untuk membaca (default mode).
- "w": Membuka file untuk menulis. Jika file ada, akan dihapus dan file baru dibuat.
- "a": Membuka file untuk menambah data (append). Data baru akan ditambahkan di akhir file.
- "b": Membuka file dalam mode biner (binary mode). Digunakan untuk file non-teks, seperti gambar atau video.
- "x": Membuat file baru. Jika file sudah ada, akan menghasilkan error.

b. Membaca File

Ada beberapa cara untuk membaca file di Python:

File data.txt

```
Steven,28,pria
Michael,25,pria
```

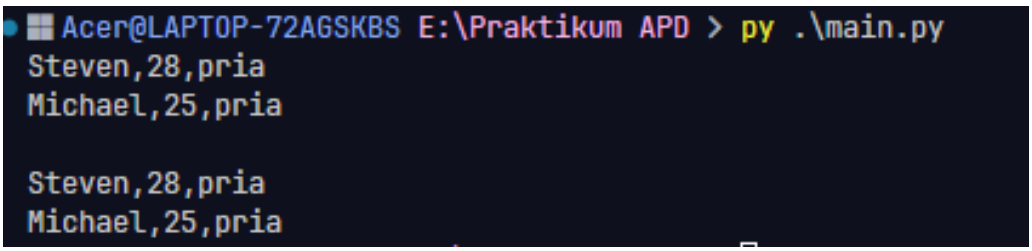
```
# Membaca seluruh isi file sekaligus
with open('nama_file.txt', 'r') as file:
    konten = file.read()
    print(konten)

# Membaca baris per baris
with open('nama_file.txt', 'r') as file:
    for baris in file:
        print(baris, end='')
```

Penjelasan:

- Keyword `with`: File akan otomatis ditutup setelah selesai digunakan.
- Fungsi `read()`: Membaca seluruh isi file sebagai string.

Output:



c. Menulis ke File

Untuk menulis data ke file, kita dapat menggunakan `write()` atau `writelines()`.

```
# Menulis teks ke file
with open('nama_file.txt', 'w') as file:
    file.write('Ini adalah baris pertama.\n')
    file.write('Ini adalah baris kedua.\n')
```

Penjelasan:

- Fungsi write(): Menulis string ke file. Jangan lupa menambahkan newline \n jika ingin memulai baris baru.
- Fungsi writelines(): Menulis daftar string ke file. Setiap elemen dalam daftar akan ditulis sebagai satu baris.

File data.txt sebelum ditulis:

```
Steven,28,pria
Michael,25,pria
```

File data.txt setelah ditulis:

```
Ini adalah baris pertama.
Ini adalah baris kedua.
```

d. Menutup File

Menutup file penting untuk memastikan bahwa semua perubahan disimpan dan sumber daya yang digunakan oleh file dibebaskan.

```
file.close()
```

Namun, lebih baik menggunakan context manager (with) yang secara otomatis menutup file setelah blok kode selesai dijalankan.

e. Penanganan Error

Ketika bekerja dengan file eksternal, ada banyak situasi yang dapat menyebabkan error, seperti file yang tidak ditemukan, masalah izin, atau disk penuh. Mengabaikan error ini dapat menyebabkan program crash atau data hilang.

Kita bisa menggunakan blok Try-Except untuk penanganan error.

```
try:
    with open("data.txt") as file:
        print(file.read())
except FileNotFoundError:
    print("File tidak ditemukan")
```

Output:

```
Acer@LAPTOP-72AGSKBS E:\Praktikum APD > py .\main.py
File tidak ditemukan
```

Studi kasus

1. Buatlah error handling untuk input nama dengan kriteria tidak boleh kosong atau hanya berisi spasi.
2. Bang Ucup ingin mengelola produk data jualan buah-buahannya. Bang Ucup membutuh program yang dapat menambahkan dan melihat produk jualannya. Buatlah dalam bentuk fungsi yang bisa menambahkan dan melihat data di file eksternal.