MODUL 4

Try Except, Input, Metode, Format, dan File Handling Pada Python

4.1 Tujuan Praktikum

Adapun tujuan pada praktikum modul 4 yaitu:

1. Mampu menggunakan PIP pada Python.
2. Mampu memahami konsep dari Try Except, Input, Metode Format, dan File Handling pada Python.

4.2 Alat dan Bahan

1. Laptop
2. Mouse
3. Software Visual Studio Code

4.3 Dasar Teori

4.3.1 PIP (Package Installer for Python)

PIP merupakan program untuk manajemen paket di Python. Tugasnya untuk menginstal, menghapus, upgrade paket Python, dll. Paket Pyhon itu sendiri merupakan sebuah modul yang berisi kode-kode python dan isipaket ini bisa diimpor ke dalam program yang dibuat.
Command untuk install pip :

```
python get-pip.py
```

Contoh penginstallan menggunakan pip :

```
pip install numpy
```

4.3.2 Try Except

Kode yang benar secara sintaksis tidak selalu berarti akan berjalan tanpa masalah. Salah satu cara Python untuk mengatasi masalah ini adalah dengan menggunakan struktur try...except. Dengan s truktur ini, kita bisa mengisolasi bagian kode yang rawan error dalam blok try. Jika terjadi kesalahan di dalam blok try, program akan langsung berpindah ke blok except untuk menjalankan kode penanganan kesalahan yang telah kita siapkan. Penggunaan try...except sangat berguna untuk menangani berbagai jenis kesalahan, seperti kesalahan saat berinteraksi dengan file, database, atau saat mengakses data dalam list atau dictionary.

4.3.3 Handling Exception

```
try:
  print(x)
except:
  print("An exception occurred")
```

Blok try akan menghasilkan exception karena x tidak ditentukan, karena blok try menghasilkan error, maka blok except akan dieksekusi, jika tidak diberi blok try maka program dapat mengalami crash.

4.3.4 Many exception

Exception dapat dipakai sebanyak yang diinginkan,misalnya jika ingin menjalankanblok kode untuk jenis kesalahan khusus.

```
try:
  print(x)
except NameError:
  print("Variable x is not defined")
except:
  print("Something else went wrong")
```

4.3.5 Else

Dapat juga untuk menggunakan kata kunci else untuk menentukan blok kode yang akandieksekusi jika tidak ada kesalahan yang muncul.

```
try:
  print("Hello")
except:
  print("Something went wrong")
else:
  print("Nothing went wrong")
```

4.3.6 Finally

Blok Finally memungkinkan untuk mengeksekusi kode, terlepas dari hasil blok try dan except.

```
try:
  print(x)
except:
  print("Something went wrong")
finally:
  print("The 'try except' is finished")
```

4.3.7 Raise an Exception

Satu hal lagi yang dapat kita manfaatkan untuk membantu penangan error adalah dengan menggunakan statement raise yang dapat mengeluarkan error secara sengaja. Biasanya raise ini digunakan bersama dengan if..else atau pemeriksaan kondisi lainnya.

```
x = -1

if x < 0:
  raise Exception("Sorry, no numbers below zero")

x = "hello"

if not type(x) is int:
  raise TypeError("Only integers are allowed")
```

Kata kunci raise digunakan untuk menampilkan pengecualian. Kita dapat menentukan jenis kesalahan apa yang akan dimunculkan, dan teks yang akan dicetak.

4.4 User Input

Input adalah masukan yang kita berikan ke program kemudian program akan memprosesnya dan menampilkan hasil outputnya.

```
username = input("Enter username:")
print("Username is: " + username)
```

```
Enter username:
```

4.5 Python String Formating

Metode format() memungkinkan untuk memformat bagian string yang dipilih. Terkadang ada bagian teks yang tidak bisa kendalikan, bisa berasal dari database, atau input pengguna. Untuk mengontrol nilai tersebut, ditambahkan placeholder (tanda kurung kurawal {})dalam teks, dan jalankan nilai melalui metode format() :

```
price = 49
txt = "The price is {} dollars"
print(txt.format(price))
```

4.5.1 Multiple Values

```
quantity = 3
itemno = 567
price = 49
myorder = "I want {} pieces of item number {} for {:.2f} dollars."
print(myorder.format(quantity, itemno, price))
```

4.5.2 Index Numbers

Kita dapat menggunakan nomor indeks (angka di dalam tanda kurung kurawal {0}) untuk memastikan nilai ditempatkan di placeholders yang benar.

```
quantity = 3
itemno = 567
price = 49
myorder = "I want {0} pieces of item number {1} for {2:.2f} dollars."
print(myorder.format(quantity, itemno, price))
```

4.5.3 Named Indexes

Menggunakan indeks bernama dengan memasukkan nama di dalam kurung kurawal {}.

```
myorder = "I have a {carname}, it is a {model}."
print(myorder.format(carname = "Ford", model = "Mustang"))
```

4.6 Python File Handling

File adalah data yang ada pada komputer, baik teks, gambar, angka, suara, video, dan lain sebagainya. File Handling adalah bagian dimana kita bisa melakukan beberapa aktivitas pada file dan direktori telah yang kita buat, aktivitas yang dapat kita lakukan terhadap file, di antaranya adalah:

```
f = open("demofile.txt")
```

Mode	Deskripsi
'r'	Membuka file untuk dibaca. (default)
'w'	Membuka file untuk ditulis. Membuat file baru jika file belum tersedia atau menimpa isi file jika file sudah ada
'x'	Membuka file untuk pembuatan eksklusif. Jika file sudah ada, maka operasi akan gagal
'a'	Membuka file dan menambahkan karakter di ujung file lama (tanpa menghapus isinya). Membuat file baru bila file belum tersedia
't'	Membuka dalam mode teks. (default)
'b'	Membuka file dalam mode biner
'+'	Membuka file untuk diupdate (membaca dan menulis)

4.7 Read Files

Untuk membuka file, kita menggunakan fungsi open(). Fungsi open() akan memanggil file, dan ditambah dengan metode read() untukmembaca konten file.

```
f = open("demofile.txt", "r")
print(f.read())
```

Jika file terletak di lokasi (folder) yang berbeda, maka kita harus menspesifikasikan file pathnya.

```
f = open("D:\\myfiles\welcome.txt", "r")
print(f.read())
```

4.7.1 Read Only Parts of the File

Selain digunakan untuk membaca seluruh teks, kita juga bisa menspesifikasikan berapabanyak karakter yang ingin di return.

```
f = open("demofile.txt", "r")
print(f.read(5))
```

Perintah di atas akan mengembalikan 5 karakter dari file yang diopen.

4.7.2 Read Lines

Selain itu kita juga bisa mengembalikan baris dari file yang diinginkam

```
f = open("demofile.txt", "r")
print(f.readline())
```

Perintah di atas akan mengembalikan satu baris dari file.

```
f = open("demofile.txt", "r")
print(f.readline())
print(f.readline())
```

Dengan menggunakan dua readline() maka kita akan mengembalikan dua baris dari file.Kita juga bisa melakukan looping diseluruh baris file

```
f = open("demofile.txt", "r")
for x in f:
  print(x)
```

4.7.3 Close Files

File yang sudah terbuka perlu ditutup kembali menggunakan metode close(). Dengan menutup file akan membebaskan memori yang terpakai. File yang dibuka akan tetap terbuka sampai kita menutupnya menggunakan metode close(). Sangat penting untuk menutup file yang tidak lagi digunakan karena alasan berikut:

- Ketika membuka file dalam skrip, sistem file biasanya menguncinya sehingga tidak ada program atau skrip lain yang dapat menggunakannya sampai file ditutup.
- Sistem file yang dipakai memiliki deskriptor file dalam jumlah terbatas yang dapat dibuat sebelum kehabisan. Meskipun jumlah ini mungkin tinggi, dimungkinkan untuk membuka banyak file dan menghabiskan sumber daya sistem file.
- Membiarkan banyak file terbuka dapat menyebabkan kondisi race atau balapan yang terjadi ketika beberapa proses mencoba untuk mengubah satu file pada saat yang sama dan dapat menyebabkan semua jenis perilaku yang tidak diharapkan.

```
f = open("demofile.txt", "r")
print(f.readline())
f.close()
```

Selain cara di atas ada cara yang lebih baik yaitu dengan manggunakan blok try, finally.

```
try:
    f = open("demofile.txt", "r")
    print(f.readline())

finally:
    f.close()
```

4.8 Python File Write

Untuk menulis pada file yang sudah ada, maka kita harus menambahkan parameter ke fungsi open():

“a” Append akan ditambahkan ke akhir file“w” Write akan menimpa konten yang ada

```
f = open("demofile2.txt", "a")
f.write("Now the file has more content!")
f.close()

#open and read the file after the appending:
f = open("demofile2.txt", "r")
print(f.read())
```

4.8.1 Create a New File

Untuk membuat file baru dengan Python, gunakan metode open(), dengan salah satuparameter berikut:
“x” Create akan membuat file, menampilkan kesalahan jika file tersebut ada “a” Append akan membuat file jika file yang ditentukan tidak ada
“w” Write akan membuat file jika file yang ditentukan tidak ada

```
f = open("myfile.txt", "x")
```

```
f = open("myfile.txt", "w")
```

Setelah skrip dijalankan maka file Bernama “myfile” dengan bentuk .txt akan dibuat

4.9 Python Delete File

Untuk menghapus file, yang harus dilakukan pertama adalah mengimpor modul OS, dan kemudian menjalankan fungsi os.remove().

```
import os
os.remove("demofile.txt")
```

4.9.1 Check if File exist

Untuk menghindari error pada kode program, hal yang harus dilakukan pertama saatingin menghapus file yaitu memeriksa apakah file tersebut ada.

```
import os
if os.path.exists("demofile.txt"):
  os.remove("demofile.txt")
else:
  print("The file does not exist")
```

4.9.2 Delete Folder

Selain menghapus file, Python juga memiliki metode untuk menghapus seluruh folder,yaitu menggunakan metode os.rmdir().

```
import os
os.rmdir("myfolder")
```