nama:arul maulana sidik nim:20220040044 kelas TI22F

ImportError

importError muncul ketika Python tidak dapat menemukan modul yang diimpor atau tidak dapat memuat modul yang diimpor. Contoh:

Dalam contoh di bawah kenepa terjadi error sebab seharusnya yang di import tersebut adalah calender sehingga nantinya akan memunculkan output kalender pada bulan 12 tahun 2012 namun dalam import tersebut tidak keditek module tersebeut

In [3]:
```
import myarul
yy = 2012
mm = 12

print(myHerlan.month(yy, mm))
```

```
---------------------------------------------------------------------------
ModuleNotFoundError                       Traceback (most recent call last)
<ipython-input-3-72a9a6352a83> in <module>
----> 1 import myarul
      2 yy = 2012
      3 mm = 12
      4
      5 print(myHerlan.month(yy, mm))

ModuleNotFoundError: No module named 'myarul'
```

IOError

IOError muncul ketika terjadi masalah I/O (input/output) pada saat mencoba melakukan operasi I/O, seperti membuka file, menulis ke file, dll. Contoh: Dalam contoh di dibawah, Python mencoba membuka file "herlan.txt" dengan mode "r" (read), tetapi file tersebut tidak ditemukan, sehingga menyebabkan IOError.

In [7]:
```
data = open("arul.txt",'r')
```

```
---------------------------------------------------------------------------
FileNotFoundError                         Traceback (most recent call last)
<ipython-input-7-204c6b645504> in <module>
----> 1 data = open("arul.txt",'r')

FileNotFoundError: [Errno 2] No such file or directory: 'arul.txt'
```

NameError

NameError muncul ketika Python tidak dapat menemukan nama yang ditentukan dalam kode. Ini dapat terjadi karena beberapa alasan, seperti: _Anda menulis nama yang salah atau tidak sesuai _Anda tidak mengimport modul yang berisi nama yang Anda panggil _Anda mencoba untuk mengakses variabel yang tidak didefinisikan contoh

In [11]:
```
print(myarul)
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-11-418dde9df00c> in <module>
----> 1 print(myarul)

NameError: name 'myarul' is not defined
```

Dalam contoh di atas, kita mencoba untuk mencetak nilai dari variabel myHerlan, tetapi variabel tersebut tidak pernah didefinisikan sebelumnya, sehingga Python mengeluarkan NameError

zeroDivisionError

ZeroDivisionError muncul ketika Anda mencoba untuk membagi dengan nol (0) pada Python. Ini adalah kesalahan matematika yang tidak mungkin terjadi dalam dunia nyata dan tidak diizinkan dalam Python. Contoh:

In [12]:
```
p = 10
b = 0
print(p/b)
```

```
---------------------------------------------------------------------------
ZeroDivisionError                         Traceback (most recent call last)
<ipython-input-12-b097b78976dd> in <module>
      1 p = 10
      2 b = 0
----> 3 print(p/b)

ZeroDivisionError: division by zero
```

Dalam contoh di atas kita ingin membagi variabel p dengen variabel b,yang menyebabkan ZeroDivisionError.

2. Exception Handling

Exception handling adalah proses menangani dan menanggapi kesalahan (exception) yang mungkin terjadi pada saat program dijalankan. Ini memungkinkan program untuk terus berjalan meskipun kesalahan terjadi, sehingga program dapat menangani kesalahan secara efektif dan menghindari crash

In [14]:
```
try:
    f = open("myfile.txt", "r")
    content = f.read()
    f.close()
except IOError:
    print("Error: Could not read file")
```

```
Error: Could not read file
```

Dalam contoh ini, kita mencoba untuk membuka, membaca dan menutup file "myfile.txt" dalam blok try. Jika terjadi kesalahan saat mencoba untuk membuka atau membaca file (IOError), kode dalam blok except akan dieksekusi, yang mencetak pesan "Error: Could not read file" ke layar. Ini memungkinkan program untuk terus berjalan meskipun kesalahan terjadi saat mencoba untuk mengakses file.

Selain except, kita juga bisa menggunakan else dan finally untuk menangani kesalahan. else akan dijalankan jika tidak terjadi kesalahan di dalam try block, finally akan dijalankan setelah try block selesai dijalankan, apapun hasil dari try block

Menangkap exception

Menangkap exception adalah proses untuk menangani kesalahan yang mungkin terjadi pada saat program dijalankan. Ini dilakukan dengan menggunakan pernyataan try dan except. Pernyataan try digunakan untuk menempatkan kode yang mungkin menyebabkan kesalahan. Jika kesalahan terjadi, eksekusi kode akan dihentikan dan kontrol akan diteruskan ke blok except yang sesuai.

Contoh:

In [15]:
```
try:
    result = 5 / 0
except ZeroDivisionError:
    print("Error: Cannot divide by zero")
```

```
Error: Cannot divide by zero
```

Dalam contoh di atas, kita menempatkan kode yang mungkin menyebabkan kesalahan (membagi dengan nol) dalam blok try. Jika kesalahan (ZeroDivisionError) terjadi, kode dalam blok except akan dieksekusi, yang mencetak pesan "Error: Cannot divide by zero" ke layar. Dengan menangani kesalahan ini, program dapat terus berjalan tanpa crash.

Anda juga dapat menangkap lebih dari satu jenis kesalahan dengan menggunakan lebih dari satu blok except :

In [17]:
```
try:
    value = int(input("Enter a number: "))
    result = 5 / value
except ValueError:
    print("Error: Please enter a valid number")
except ZeroDivisionError:
    print("Error: Cannot divide by zero")
```

```
Enter a number:  valid number
Error: Please enter a valid number
```

Dalam contoh ini, kita mencoba untuk mengonversi input ke integer dan membagi 5 dengan nilai tersebut dalam blok try. Jika terjadi kesalahan saat mencoba untuk mengonversi input (ValueError), kode dalam blok except pertama akan dieksekusi. Jika terjadi kesalahan saat membagi dengan nol (ZeroDivisionError), kode dalam blok except kedua akan dieksekusi.