

MODUL PRAKTIKUM
PEMROGRAMAN BERORIENTASI OBJEK
Minggu 7



Exception handling dan GUI

Disusun Oleh :

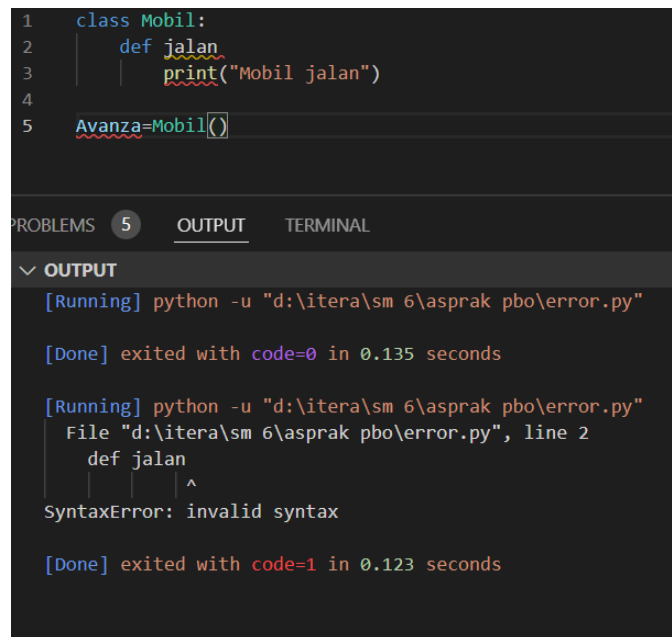
Andhika Putra Pratama	119140224
Andhika Wibawa B.	119140218
Nurul Aulia Larasati	119140088
Ihza Fajrur Rachman H.	119140130
Enrico Johanes S.	119140021
M. Ammar Fadhila R.	119140029

PROGRAM STUDI TEKNIK INFORMATIKA
JURUSAN TEKNIK ELEKTRO, INFORMATIKA DAN SISTEM FISIS
INSTITUT TEKNOLOGI SUMATERA
2022

1. Exception Handling

A. Syntax error

Syntax error terjadi karena adanya pelanggaran peraturan penulisan saat eksekusi.



```
1 class Mobil:
2     def jalan
3         print("Mobil jalan")
4
5 Avanza=Mobil()
```

PROBLEMS 5 OUTPUT TERMINAL

▼ OUTPUT

[Running] python -u "d:\itera\sm 6\asprak pbo\error.py"

[Done] exited with code=0 in 0.135 seconds

[Running] python -u "d:\itera\sm 6\asprak pbo\error.py"

File "d:\itera\sm 6\asprak pbo\error.py", line 2
def jalan
 ^
SyntaxError: invalid syntax

[Done] exited with code=1 in 0.123 seconds

Gambar 1. Contoh Syntax error

Ketika terjadi error ini, parser akan mengulangi baris yang melanggar dan menampilkan 'panah' kecil yang menunjuk pada titik paling awal di baris tempat kesalahan terdeteksi. Jenis error (SyntaxError) ini cukup berbeda dengan error lainnya karena kode yang akan dieksekusi Python menjadi tidak lengkap, sehingga program harus dibatalkan (*terminate*).

B. Exceptions

Kesalahan (Exception) terjadi ketika jika terdapat suatu pernyataan yang secara sintaks benar, namun menyebabkan kesalahan ketika dilakukan upaya untuk mengeksekusinya. Error ini umumnya bisa diakali agar tidak menghentikan eksekusi program, tapi akan mengubah alur program.

Contoh-contoh exception adalah ketika membagi angka dengan angka nol, mengakses file/folder yang tidak ada, menambahkan 2 tipe data yang tidak kompatibel, mengakses indeks yang tidak ada pada array, dan lain-lain.

Untuk mengelola berbagai jenis Exception pada Python, terdapat syntax-syntax/statement sebagai berikut:

- **try:** Digunakan untuk mengisolasi kode yang mungkin dapat menimbulkan kesalahan (untuk kemudian masuk ke blok *except* atau *else*)

- **except:** Digunakan untuk mengeksekusi kode jika terjadi kesalahan
- **else:** Digunakan untuk mengeksekusi kode ketika tidak ada kesalahan
- **finally:** Digunakan untuk mengeksekusi kode, tidak bergantung dari hasil akhir *try/except/else* yang dihasilkan (akan selalu dieksekusi)
- **raise:** Digunakan untuk menjadikan kondisi tertentu sebagai kesalahan, atau mengangkat kode error tertentu yang sudah ada (bawaan Python)
- **assert:** Digunakan untuk mendeteksi kesalahan di awal

Baris kode yang kemungkinan akan terjadi suatu exception dapat ditulis dalam statement *try* dan statement yang menghandle exception tersebut dapat ditulis statement *except*. Berikut ini adalah contoh dasar penggunaan *try except* (kurang baik karena semua kode error yang tertangkap tidak spesifik):

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

An exception occurred

Pada kode di atas dapat dilihat bahwa kode pada statement **try else** tidak dijalankan, namun statement **try except** dijalankan. Lalu, setelah kita teliti lebih dalam bisa dilihat bahwa coding pada statement **try** seharusnya menghasilkan error (karena x belum diinisialisasi), namun tidak terjadi error pada output di console. Hal ini terjadi dikarenakan terdapat statement **except** yang mengatasi jika terjadi suatu error pada codingan di statement **try**. Contoh lain penggunaan exception handling yaitu:

```
1  try:
2      x=int("hello")
3      print(x)
4  except NameError:
5      print("Variable x is not defined")
6  except ValueError:
7      print("Nilai bukan angka")
8
```

[Running] python -u "d:\itera\sm 6\asprak pbo\error.py"
Nilai bukan angka

Sintaks di atas menjalankan statement `except ValueError` karena `x` memiliki tipe data string dan ingin diubah menjadi integer. Karena nilai `x` tidak bisa di convert, muncullah error. Contoh lainnya lagi yaitu:

```
x = 10
try:
    print(x)
except:
    print("An exception occurred")
else:
    print("x is defined")
```

```
10
x is defined
```

Sedangkan, kode di atas dapat dilihat bahwa kode pada statement `try` dan `else` berhasil dijalankan. Hal tersebut terjadi karena pada statement `try` tidak terjadi error. Alhasil, statement `except` tidak dijalankan dan statement `else` dijalankan.

Finally selalu dieksekusi setelah statement `try` dan `except` baik ketika ada error maupun tidak. statement ini umumnya digunakan untuk clean-up.

```
1 def Bagi(x,y):
2     try:
3         hasil=x/y
4     except ZeroDivisionError:
5         print("Tidak Bisa Dibagi Dengan Nol")
6     else:
7         print("hasilnya ", hasil)
8     finally:
9         print("sudah selesai")
10
11 Bagi(6,3)
12 Bagi(6,0)
13 Bagi("3","2")
```

```
[Running] python -u "d:\itera\sm 6\asprak pbo\error.py"
hasilnya 2.0
sudah selesai
Tidak Bisa Dibagi Dengan Nol
sudah selesai
sudah selesai
Traceback (most recent call last):
  File "d:\itera\sm 6\asprak pbo\error.py", line 13, in <module>
    Bagi("3","2")
  File "d:\itera\sm 6\asprak pbo\error.py", line 3, in Bagi
    hasil=x/y
TypeError: unsupported operand type(s) for /: 'str' and 'str'
```

Pada sintaks di atas, terlihat bahwa ada maupun tidak adanya error, statement `finally` akan selalu di eksekusi. Bila tidak ada kesalahan, maka statement `else`

dieksekusi. jika ada exception yang tidak dihandle oleh statement except, exception akan dimunculkan setelah statement **Finally** di eksekusi.

Statement **Raise** digunakan untuk membuat/memunculkan exception secara paksa.

```
1 x = "hello"
2
3 if not type(x) is int:
4     raise TypeError("Only integers are allowed")
```

```
[Running] python -u "d:\itera\sm 6\asprak pbo\error.py"
Traceback (most recent call last):
  File "d:\itera\sm 6\asprak pbo\error.py", line 4, in <module>
    raise TypeError("Only integers are allowed")
TypeError: Only integers are allowed
```

Pada sintaks di samping, jika nilai variabel bukan integer, maka akan dibuat sebuah exception menggunakan **raise** yang menspesifikasikan bahwa error yang terjadi adalah type error. Normalnya, tidak terjadi error bila x bernilai string. tetapi, karena adanya statement **raise**, sintaks tersebut dapat dibuat seolah-olah menjadi error.

Terakhir, Statement **Assert** digunakan untuk membuat exception **jika** suatu kondisi terpenuhi (atau tidak terpenuhi). Statement ini bisa juga digunakan sebagai alternatif dari debugging secara langsung pada program (tidak memerlukan intervensi dari programmer).

```
x = "hello"

#if condition returns True, then nothing happens:
assert x == "hello"

#if condition returns False, AssertionError is raised:
assert x == "goodbye"
```

```
Traceback (most recent call last):
  File "demo_ref_keyword_assert.py", line 5, in <module>
    assert x == "goodbye"
AssertionError
```

Pada sintaks di atas dapat terlihat bahwa ketika line ke-3 dieksekusi tidak terjadi apapun, namun console mengeluarkan error pada line 5. Hal tersebut terjadi karena pada line ke-5 kondisi yang assert terpenuhi. Maka dari itu console akan mengembalikan "AssertionError".

```

1  def square(x):
2      assert x>=0, 'Only positive numbers are allowed'
3      return x*x
4
5  try:
6      square(-2)
7  except AssertionError as msg:
8      print(msg)
9

```

```

[Running] python -u "d:\itera\sm 6\asp
Only positive numbers are allowed

```

Pada sintaks di atas, terdapat fungsi `square` yang menerima parameter `x`. di dalamnya diberikan statement **assert** dengan kondisi bila `x` lebih dari atau sama dengan 0, akan terjadi assertion error yang memunculkan string “Only positive numbers are allowed”. bila tidak ada error, maka fungsi mengembalikan nilai `x` kuadrat.

Selanjutnya, terdapat statement **try** yang memanggil fungsi `square` dengan parameter `-2`. Bila terjadi assertion error, maka ada **except** yang handle dengan aksi yang dilakukan adalah mencetak pesan pada **assert**. Karena `-2` kurang dari 0, maka program akan mengeksekusi statement **except**.

a. Macam-macam exceptions

Jenis exception	Penyebab error
ImportError	Muncul ketika mengimpor suatu modul/file yang tidak ada/tidak dapat ditemukan
IndexError	Muncul ketika mengakses suatu indeks yang tidak ada/melebihi batas (misal pada suatu list)
KeyError	Muncul ketika mengakses suatu kunci yang tidak ada pada tipe data <i>dictionary</i> (ingat bahwa pada dictionary terdapat 2 komponen utama yaitu <i>key</i> /kunci dan <i>value</i>)
KeyboardInterrupt	Muncul ketika user menekan tombol interrupt (Ctrl+C or Delete).
NameError	Muncul ketika sebuah variabel tidak ditemukan pada scope local ataupun global
SyntaxError	Muncul ketika kode tidak lengkap/kesalahan penulisan pada kode (typo dsb)

TypeError	Muncul ketika suatu fungsi atau operasi diaplikasikan pada suatu objek dengan tipe yang salah (berhubungan dengan konversi tipe data)
ValueError	Muncul ketika suatu fungsi mendapatkan argumen dengan tipe yang benar namun value yang salah

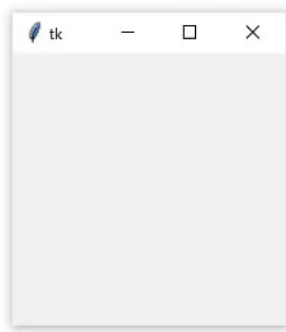
2. Graphical User Interface (GUI)

GUI merupakan singkatan dari Graphical User Interface, yang berfungsi sebagai “wajah” program agar pengguna dapat menggunakan program tersebut dengan mudah. Input atau perintah yang dimasukkan oleh pengguna yang diterima melalui GUI dikonversi oleh shell atau kode yang akan menjadi perintah untuk melakukan suatu tindakan tertentu. Contoh GUI yang sederhana adalah mesin ATM yang berada di ITERA. Interaksi antara program dan pengguna dijumpai oleh tombol atau layar sentuh yang sejajar dengan sejumlah menu yang tersedia. Di dalam Python, terdapat *package* GUI yang sudah bawaan ketika menginstal Python ke dalam mesin/PC, yaitu *tkinter*. Tkinter merupakan kependekan dari Toolkit interface, yang merupakan *library* GUI standar bawaan Python.

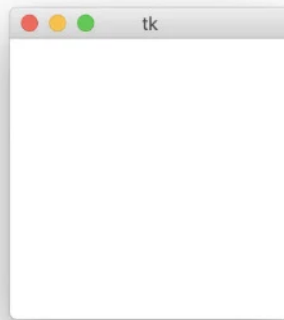
A. Membangun GUI dengan aplikasi Python.

Untuk membangun GUI dengan tkinter dalam kita dapat mengimport *library* tkinter secara langsung, karena pada dasarnya tkinter adalah *library* bawaan Python. Dalam tkinter kita akan diperkenalkan dengan elemen dasar dari GUI Tkinter yaitu **window** dan elemen GUI lainnya, seperti kotak teks, label, dan tombol, yang dikenal sebagai **widget**.

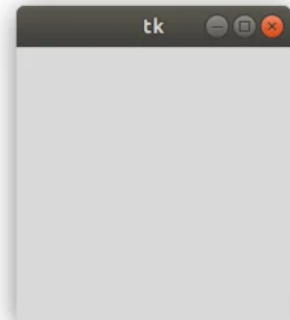
Pertama-tama untuk membangun sebuah tampilan GUI dapat memanggil fungsi Tk() untuk membuat **window** sehingga akan memunculkan tampilan dibawah ini sesuai dengan OS yang user gunakan.



(a) Windows



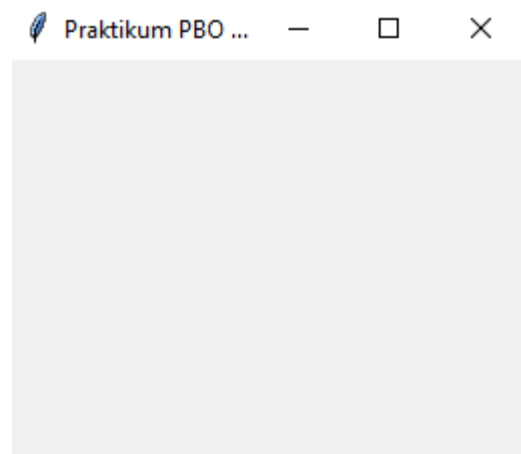
(b) macOS



(c) Ubuntu

Selain itu terdapat fungsi tambahan lain seperti fungsi `title()` untuk mengubah nama window dan `mainloop()` untuk *loop* tak terbatas yang digunakan untuk menjalankan aplikasi yang sedang dibuat. Contoh awal membuat tampilan GUI dengan Python dapat menggunakan kode dasar berikut:


```
1. from tkinter import *
2. window=Tk()
3. window.title("Praktikum PBO
   ITERA")
4. window.mainloop()
```

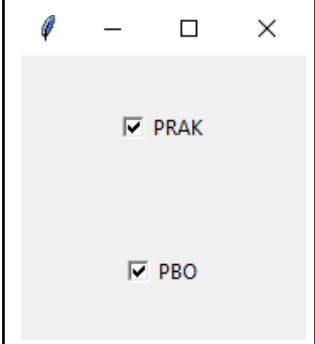
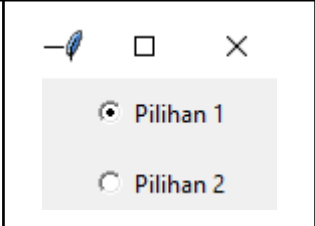
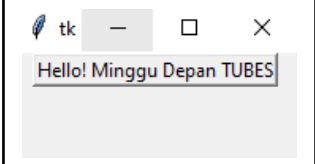

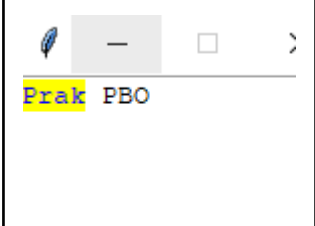
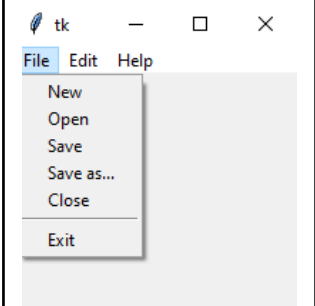
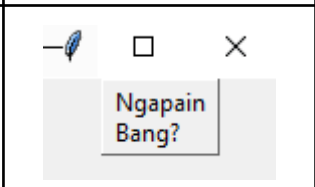



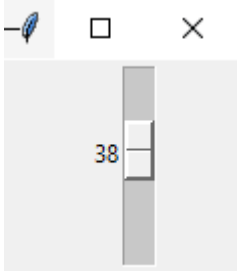
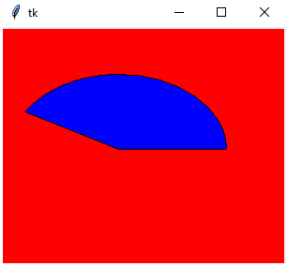
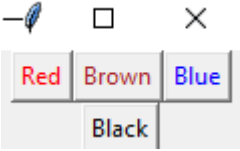
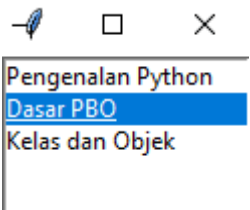
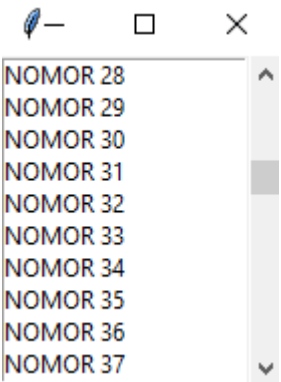
Output kode (Windows)

B. Membuat Widget

Widget adalah sebuah objek yang ditampilkan dalam GUI untuk berinteraksi dengan pengguna aplikasi. Dalam tkinter terdapat beberapa widget yang dapat ditambahkan pada GUI yang sedang kita bangun, diantaranya:

Widget	Deskripsi	Gambar
Button	Merupakan tombol sederhana yang digunakan untuk mengeksekusi suatu operasi atau perintah lainnya	

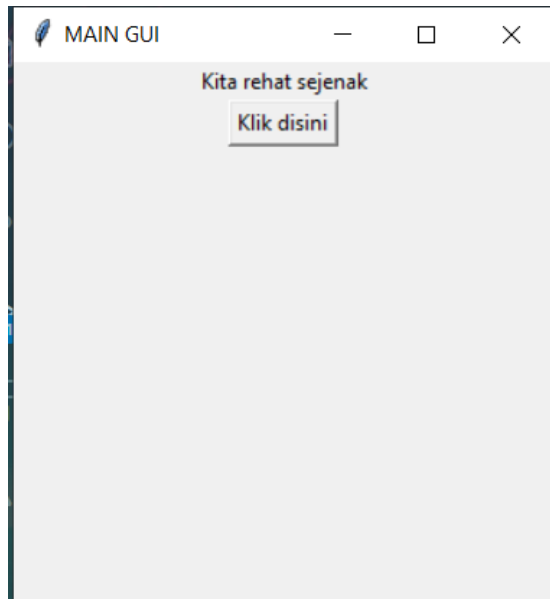
Checkbox	Mempresentasikan sebuah variabel yang dapat dipilih lebih dari dua pilihan	
RadioButton	Mempresentasikan suatu nilai dari variabel yang dapat memiliki satu nilai	
Label	Widget untuk menampilkan teks atau gambar	
Entry	Kolom memasukkan sebuah teks	
Text	Mengatur tampilan teks dan mengedit teks dengan gaya dan atribut sesuai keinginan, juga mendukung pemasangan image dan window	
Menu	Widget yang digunakan untuk membuat <i>pull-down</i> menu	
Message	Menampilkan sebuah teks pada window. Pada dasarnya sama seperti widget label, tetapi teks dapat diatur tata letaknya secara otomatis	
SpinBox	Memungkinkan user memilih nilai dari sekumpulan nilai dengan format rentang angka yang ditentukan	

Scale	Membuat panel drag untuk mengeset nilai numerik	
Canvas	Widget dapat digunakan untuk membuat grafik dan plot, membuat editor grafik, dan untuk mengimplementasikan pengubahan widget	
Frame	Membuat sebuah kotak dalam GUI, Frame dapat diberi border dan background serta dapat digunakan untuk mengelompokkan widget lainnya ketika membuat aplikasi atau layout dialog	
Listbox	Membuat sebuah daftar pilihan dari beberapa item	
Scrollbar	Digunakan untuk menggulung canvas, entry, listbox, dan teks	

C. Contoh Sederhana Penggunaan GUI

```
1  import tkinter as tk
2
3  nyoba = tk.Tk()
4  readimage = tk.PhotoImage(file="rehat.gif")
5  def pushed():
6      nyoh = tk.Label(nyoba, image=readimage)
7      nyoh.pack()
8
9  nyoba.geometry("300x300")
10 nyoba.title("MAIN GUI")
11 title = tk.Label(nyoba, text="Kita rehat sejenak")
12 button = tk.Button(nyoba, text="Klik disini", command=pushed)
13 title.pack()
14 button.pack()
15
16 nyoba.mainloop()
```

Hal pertama yang dilakukan adalah melakukan import library tkinter ke dalam kodingan. Agar lebih mudah digunakan, gunakan “as” dan tentukan nama pengganti tkinter yang mudah. Setelah itu buatlah sebuah variabel untuk menyimpan tk.Tk() sebagai window/widget utama program. PhotoImage() adalah fungsi untuk membaca file image yang berada di direktori yang sama dengan kodingan. Buat sebuah fungsi sebuah aksi ketika sebuah tombol ditekan, yaitu mengeluarkan gambar yang disimpan melalui PhotoImage sebelumnya. pack() berfungsi sebagai “wrapper” atau pembungkus tindakan yang membuat tindakan tersebut bisa muncul atau dilakukan nantinya. geometry(“www x hhh”) adalah fungsi yang digunakan untuk membuat window berukuran width x height. title() berfungsi membuat judul dari window. Label merupakan teks yang berada di dalam window dan berada di posisi center. Button berfungsi membuat tombol dan aksi yang dilakukan ketika tombol tersebut ditekan dipanggil menggunakan command.



Di atas merupakan tampilan ketika kodingan dijalankan. Dan ketika tombol “klik disini” ditekan, maka akan menghasilkan tampilan berikut :



References

<https://realpython.com/python-gui-tkinter/>

<https://www.tutorialspoint.com/python>

<https://www.programiz.com/python-programming/exceptions>

<https://www.geeksforgeeks.org/python-gui-tkinter/>

<https://docs.python.org/3/tutorial/errors.html#>

<https://www.tutorialsteacher.com/python/python-assert>