

TUGAS PYTHON

MODUL IV-VII

Nama : Nessa Kartika

NIM : 211001039

Kelas : 4 D

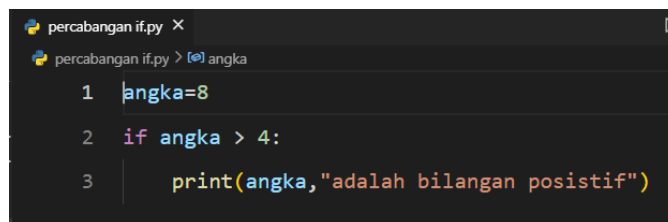
MODUL IV

1. Percabangan

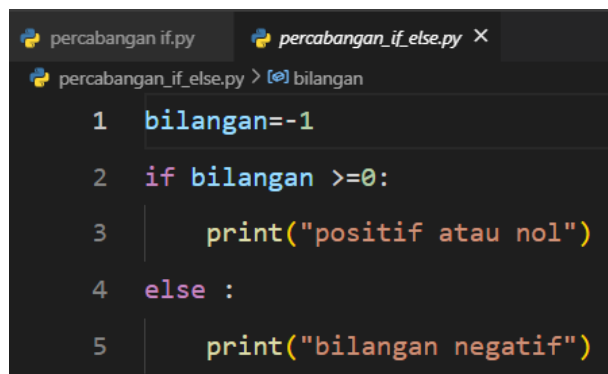
Percabangan adalah cara yang digunakan untuk mengambil keputusan apabila di dalam program dihadapkan pada kondisi tertentu. Jumlah kondisinya bisa satu, dua atau lebih. Hasil dari pengecekan kondisi adalah **True** atau **False**.

Di Python ada 3 jenis pernyataan yang digunakan untuk percabangan, yaitu:

- **if.** Pernyataan **if** terdiri dari ekspresi boolean diikuti oleh satu baris atau lebih pernyataan.
- **if...else.** Bila pernyataan **if** benar, maka blok pernyataan **if** dieksekusi. Bila salah, maka blok pernyataan **else** yang dieksekusi.
- **if...elif...else.** Disebut juga **if** bercabang. Bila ada kemungkinan beberapa kondisi bisa benar maka digunakan pernyataan **if...elif** atau **if...elif...else**.



```
percabangan if.py x
percabangan if.py > angka
1 angka=8
2 if angka > 4:
3     print(angka,"adalah bilangan positif")
```



```
percabangan if.py  percabangan_if_else.py x
percabangan_if_else.py > bilangan
1 bilangan=-1
2 if bilangan >=0:
3     print("positif atau nol")
4 else :
5     print("bilangan negatif")
```

```

percabangan_if_else.py X
percabangan_if_else.py > [?] bilangan
1  bilangan=5.5
2  ~ if bilangan >0:
3      print ("bilangan positif")
4  ~ elif bilangan ==0:
5      print("nol")
6  ~ else :
7      print("bilangan negatif")

```

2. Tambahan if bersarang

```

percabangan if.py  percabangan_if_bersarang.py X
percabangan_if_bersarang.py > [?] gaji
1  gaji=10000000
2  berkeluarga=True
3  punyaRumah=True
4  if gaji>3000000:
5      print("gaji sudah diatas umr")
6      if berkeluarga:
7          print("wajib ikut asuransi dan menabung untuk pensiun")
8      else:
9          print("tidak perlu ikut asuransi")
10     if punyaRumah:
11         print("wajib bayar pajak rumah")
12     else:
13         print("tidak wajib bayar pajak rumah")
14 else:
15     print("gaji belum umr")

```

3. Contoh Program Percabangan Indeks Nilai Statis

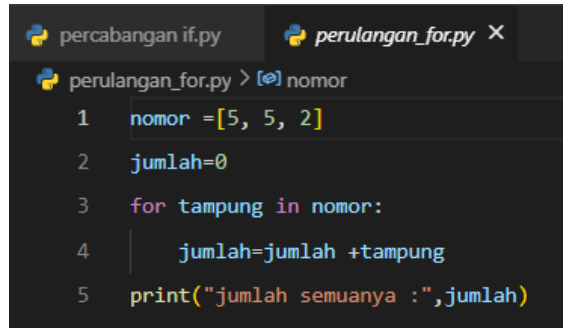
```

percabangan if.py  latihan_percabangan_indeks_nilai.py X
latihan_percabangan_indeks_nilai.py > [?] nilai
1  nilai = 80
2
3  if nilai >= 85 and nilai <=100:
4      print("Nilai A")
5  elif nilai >= 70 and nilai <= 84:
6      print("Nilai B")
7  elif nilai >= 55 and nilai <= 69:
8      print("Nilai D")

```

4. Perulangan dengan Menggunakan for

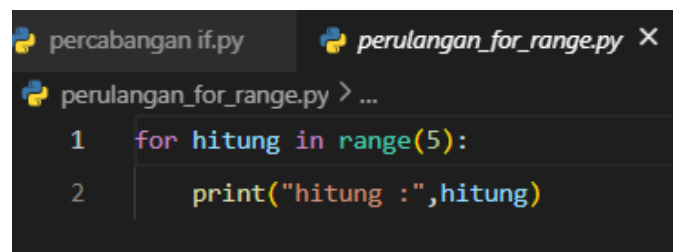
- **var** adalah variabel yang digunakan untuk penampung sementara nilai dari sequence pada saat terjadi perulangan.
- **Sequence** adalah tipe data berurut seperti string, list, dan tuple.



```
perulangan_for.py > [e] nomor
1 nomor =[5, 5, 2]
2 jumlah=0
3 for tampung in nomor:
4     jumlah=jumlah +tampung
5 print("jumlah semuanya :",jumlah)
```

5. Perulangan for dengan range

Fungsi **range()** dapat digunakan untuk menghasilkan deret bilangan.

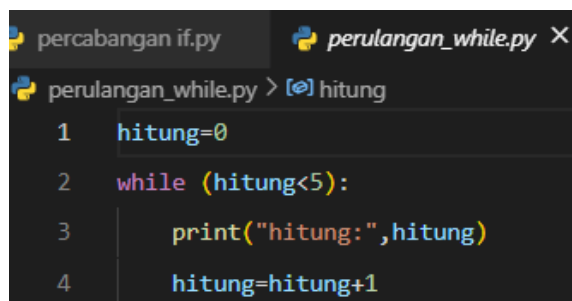


```
perulangan_for_range.py > ...
1 for hitung in range(5):
2     print("hitung :",hitung)
```

6. Perulangan Menggunakan while

Perulangan menggunakan **while** akan menjalankan blok pernyataan terus menerus selama kondisi bernilai **benar**.

Expression merupakan ekspresi atau kondisi apa saja, dan untuk nilai selain nol dianggap True.



```
perulangan_while.py > [e] hitung
1 hitung=0
2 while (hitung<5):
3     print("hitung:",hitung)
4     hitung=hitung+1
```

7. Contoh Program Kelipatan Bilangan Genap

Pengulangan dengan **for**, menampilkan bilangan genap dari 0 hingga batas terakhir bilangan input. Misalnya menginput angka 10 maka akan tampil 0, 2, 4, 6, 8.

```
percabangan if.py  latihan_perulangan_kelipatan_genap.py X
latihan_perulangan_kelipatan_genap.py > [?] i
1  i=0
2  n=int (input("masukkan batas:"))
3
4  for i in range(n):
5      if i%2==0:
6          print("bilangan:",i)
7      i=i+1
8
```

Latihan:

Buatlah program kelipatan bilangan genap dengan menampilkan banyaknya jumlah. Misalnya, apabila diinput 10, maka yang tampil adalah 0 2 4 6 8 10 12 14 16 18 (10 bilangan).

```
9  #latihan
10 print("Latihan")
11 n = int(input("Masukkan nilai n: "))
12
13 # Mencari kelipatan bilangan genap dari 0 hingga n
14 count = 0
15 for i in range(n+1):
16     if i % 2 == 0:
17         count += 1
18
19 # Menampilkan kelipatan bilangan genap dan jumlah
20 print("Kelipatan bilangan genap dari 0 hingga", n, "adalah:")
21 for i in range(0, (count * 2) + 1, 2):
22     print(i, end=" ")
23 print("(", count, "bilangan)")
24
```

8. Fungsi

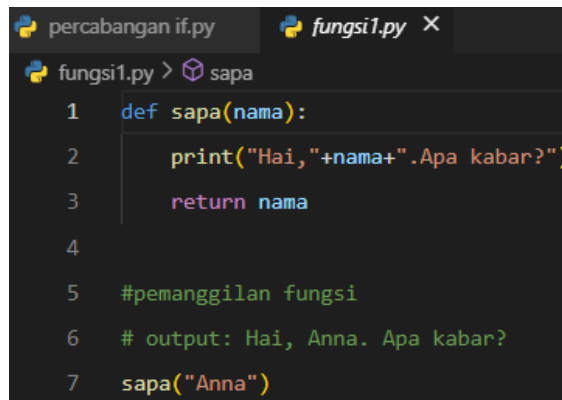
Berikut adalah sintaks yang digunakan untuk membuat fungsi :

def function_name(parameters):

"""function_docstring"""

statement(s)

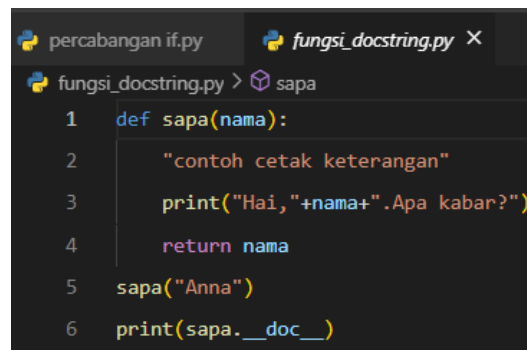
return [expression]



```
percabangan if.py  fungsi1.py X
fungsi1.py > sapa
1 def sapa(nama):
2     print("Hai, "+nama+". Apa kabar?")
3     return nama
4
5 #pemanggilan fungsi
6 # output: Hai, Anna. Apa kabar?
7 sapa("Anna")
```

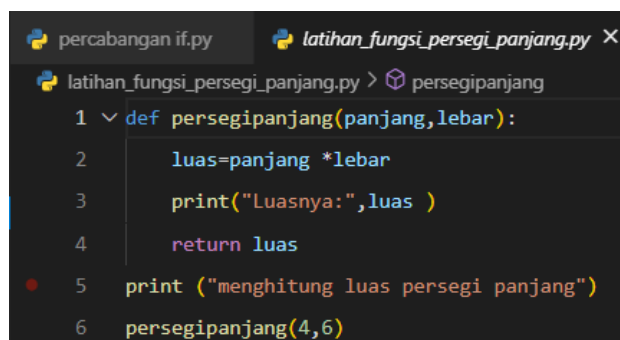
9. Docstring

Docstring adalah singkatan dari documentation string. Ini berfungsi sebagai dokumentasi atau keterangan singkat tentang fungsi yang kita buat. Cara mengaksesnya adalah dengan menggunakan format **namafungsi.__doc__**



```
percabangan if.py  fungsi_docstring.py X
fungsi_docstring.py > sapa
1 def sapa(nama):
2     "contoh cetak keterangan"
3     print("Hai, "+nama+". Apa kabar?")
4     return nama
5 sapa("Anna")
6 print(sapa.__doc__)
```

10. Contoh Program Luas Persegi Panjang dengan Fungsi



```
percabangan if.py  latihan_fungsi_persegi_panjang.py X
latihan_fungsi_persegi_panjang.py > persegipanjang
1 def persegipanjang(panjang, lebar):
2     luas=panjang *lebar
3     print("Luasnya:",luas )
4     return luas
5 print ("menghitung luas persegi panjang")
6 persegipanjang(4,6)
```

```
percabangan if.py latihan_fungsi_persegi_panjang_dinamis.py X
latihan_fungsi_persegi_panjang_dinamis.py > ...
1 def persegipanjang(panjang,lebar):
2     luas=panjang *lebar
3     print("Luasnya:",luas )
4     return luas
5
6 print ("menghitung luas persegi panjang")
7 a=int(input("Masukkan nilai panjang:"))
8 b=int(input("Masukkan nilai lebar :"))
9 persegipanjang(a,b)
```

Latihan:

Buatlah program dinamis menghitung luas persegi panjang dan persegi dengan menggunakan 1 fungsi. Misalnya, apabila diinput panjang = 4 dan lebar 3, maka tampil luas persegi panjang = 12. Dan apabila diinput sisi persegi = 3, maka tampil luas persegi = 9. Contoh tampilan terminal seperti gambar dibawah.

```
11 #Latihan
12 print("Latihan")
13 print("menghitung luas persegi panjang & persegi")
14 def persegipanjang(panjang,lebar):
15     luas=panjang *lebar
16     print("Luasnya:",luas )
17     return luas
18 print ("persegi panjang")
19 a=int(input("masukan panjang:"))
20 b=int(input("Masukan lebar :"))
21 persegipanjang(a,b)
22 def persegi(sisi):
23     luas=sisi *sisi
24     print("Luasnya:",luas )
25     return luas
26 print ("persegi ")
27 a=int(input("masukan sisi:"))
28 persegi(a)
```

```
Latihan
menghitung luas persegi panjang & persegi
persegi panjang
masukan panjang:4
Masukan lebar :3
Luasnya: 12
persegi
masukan sisi:3
Luasnya: 9
PS C:\Nessa Kartika\Python\Tugas Python\Modul4> |
```

MODUL V

1. Object Oriented Programming

Object Oriented Programming (OOP) merupakan suatu konsep pemrograman yang menekankan pada paradigma atau cara pandang terhadap suatu masalah berdasarkan "object". Dalam konsep OOP semua yang ada di dunia ini adalah object dan direpresentasikan dalam bentuk object. Object di dunia nyata memiliki ciri atau attribute dan juga aksi atau kelakuan (behaviour).

Istilah - Istilah Dalam OOP:

- **Kelas** adalah cetak biru atau prototipe dari object dimana kita mendefinisikan atribut dari suatu object. Atribut ini terdiri dari data member (variabel) dan fungsi (metode).
- **Variabel Kelas** adalah variabel yang dishare atau dibagi oleh semua instance (turunan) dari kelas. Variabel kelas didefinisikan di dalam kelas, tapi di luar metode-metode yang ada dalam kelas tersebut.
- **Data member** adalah variabel yang menyimpan data yang berhubungan dengan kelas dan objeknya.
- **Overloading Fungsi** adalah fungsi yang memiliki nama yang sama di dalam kelas, tapi dengan jumlah dan tipe argumen yang berbeda sehingga dapat melakukan beberapa hal yang berbeda.
- **Overloading Operator** adalah pembuatan beberapa fungsi atau kegunaan untuk suatu operator. Misalnya operator + dibuat tidak hanya untuk penjumlahan, tapi juga untuk fungsi lain.
- **Variabel Instansiasi** adalah variabel yang didefinisikan di dalam suatu metode dan hanya menjadi milik dari instance kelas.
- **Pewarisan/Inheritansi (Inheritance)** adalah pewarisan karakteristik sebuah kelas ke kelas lain yang menjadi turunannya.
- **Instance** adalah istilah lain dari objek suatu kelas. Sebuah objek yang dibuat dari prototipe kelas Lingkaran misalnya disebut sebagai instance dari kelas tersebut.
- **Instansiasi** adalah pembuatan instance/objek dari suatu kelas.
- **Metode** adalah fungsi yang didefinisikan di dalam suatu kelas.
- **Objek** adalah instansiasi atau perwujudan dari sebuah kelas. Bila kelas adalah prototipenya, dan objek adalah barang jadinya.

2. Perkenalan Kelas & Object

```
oop_dasar.py X oop_init.py oop
oop_dasar.py > ...
1 #kelas
2 class Marvel:
3     pass
4
5 #object
6 marvel1 = Marvel()
7 marvel2 = Marvel()
8 marvel3 = Marvel()
9
10 marvel1.name = "Spiderman"
11 marvel1.health = "1000"
12
13 marvel2.name = "Hulk"
14 marvel2.health = "800"
15
16 marvel3.name = "Thor"
17 marvel3.helath = "900"
18
19 #pemanggilan
20 print(marvel1)
21 print(marvel1.__dict__)
22 print(marvel1.name)
```

3. Kelas dan Object Sederhana

```
oop_dasar.py oop_init.py X oop_kelas_instance.py oop_method.py ▶ ⌵ ☺ 🪴 ...
oop_init.py > ...
1 class Marvel:
2     def __init__(self,inputName,inputHealth,inputPower,inputArmor):
3         self.name = inputName
4         self.health = inputHealth
5         self.power = inputPower
6         self.armor = inputArmor
7
8     marvel1 = Marvel("Spiderman",100,10,90)
9     marvel2 = Marvel("Hulk",90,15,100)
10    marvel3 = Marvel("Thor",80,5,70)
11
12    print(marvel1.name)
13    print(marvel2.health)
14    print(marvel3.__dict__)
```


4. Variabel Kelas dan Object

```
oop_dasar.py  oop_init.py  oop_kelas_instance.py X  oop_method.py  ▶  🐞  📄  ...
oop_kelas_instance.py > ...
1  class Marvel:
2      #class variabel
3      jumlah = 0
4
5      def __init__(self,inputName,inputHealth,inputPower,inputArmor):
6          #instance variabel:
7          self.name=inputName
8          self.health=inputHealth
9          self.power=inputPower
10         self.armor=inputArmor
11         Marvel.jumlah += 1
12         print("Hero Marvel dengan nama :" + inputName)
13
14  marvel1 = Marvel("Spiderman",1000,900,800)
15  print(Marvel.jumlah)
16  marvel2=Marvel("Hulk",900,1000,900)
17  print(Marvel.jumlah)
18  marvel3=Marvel("Thor",800,700,600)
19  print(Marvel.jumlah)
```

5. Method

```
Welcome | oop_dasar.py | oop_init.py | oop_kelas_instance.py | o
oop_method.py > ...
1  class marvel:
2      def __init__(self,inputname,inputhealth,inputpower,inputarmor) :
3          # instance variable
4          self.name = inputname
5          self.health = inputhealth
6          self.power = inputpower
7          self.armor = inputarmor
8
9  #void function, method tanpa return
10     def siapa(self):
11         print("namaku adalah :" + self.name)
12
13     #method dengan argumen
14     def healthtambah(self,tambah):
15         self.health += tambah
16
17     #method dengan return
18     def gethealth(self):
19         return self.health
20
21     marvell1=marvel('iron man',1000,900,800)
22     marvell2=marvel("thor",900,1000,900)
23     marvell3=marvel("iron man",800,700,600)
24
25     #pemanggilan method
26     marvell1.siapa()
27
28     #pemakaian method dengan argumen
29     marvell1.healthtambah(10)
30     print(marvell1.health)
31
32     #mengembalikan nilai dengan method
33     print(marvell1.gethealth())
```

6. Game dengan OOP

```
oov_game.py > ...
1  class marvel:
2      def __init__(self,name,health,attackpower,armornumber):
3          self.name=name
4          self.health=health
5          self.attackpower=attackpower
6          self.armornumber=armornumber
7
8      def serang(self,lawan):
9          print(self.name + " menyerang " + lawan.name )
10         lawan.diserang(self,self.attackpower)
11
12     def diserang (self,lawan,attackpower_lawan):
13         print (self.name + " diserang " +lawan.name)
14         attact_diterima=attackpower_lawan
15         print( "serangan terasa : " + str(attact_diterima))
16         self.health-=attact_diterima
17         print("Darah " + self.name + " tersisa " + str(self.health))
18
19     ironman=marvel("iron man",100,10,5)
20     thor=marvel("thor",95,15,10)
21
22     #ironman.serang()
23     ironman.serang(thor)
24     #print("\n")
25     #ironman.serang(thor)
26     #print("\n")
27     #thor.serang(ironman)
```

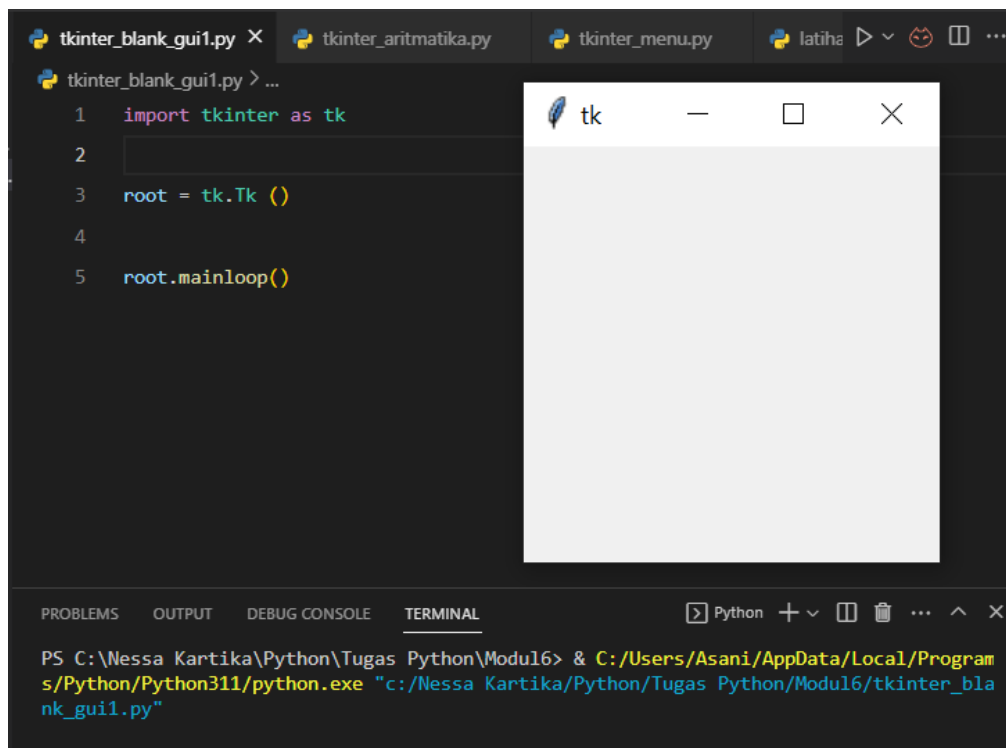
BAB VI

1. Pengertian

Tkinter

Tkinter adalah graphic user interface (GUI) standar python digunakan untuk membuat tampilan aplikasi dengan komponen-komponen yang ada di modul tkinter seperti Button, Textbox, Label, Frame, Window yang mana sangat mendukung dalam penciptaan aplikasi GUI. Untuk dapat memahami cara kerja Tkinter ini Anda harus paham menggunakan paradigma Object Oriented Programming dalam modul sebelumnya.

2. Tampilan GUI sederhana



3. Contoh Aplikasi Aritmatika Pertambahan

The image shows a Python IDE with a file explorer at the top containing files: `Welcome`, `tkinter_blank_gui1.py`, `tkinter_aritmatika.py` (active), `tkinter_menu.py`, and `latihan.py`. The editor displays the following Python code for `tkinter_aritmatika.py`:

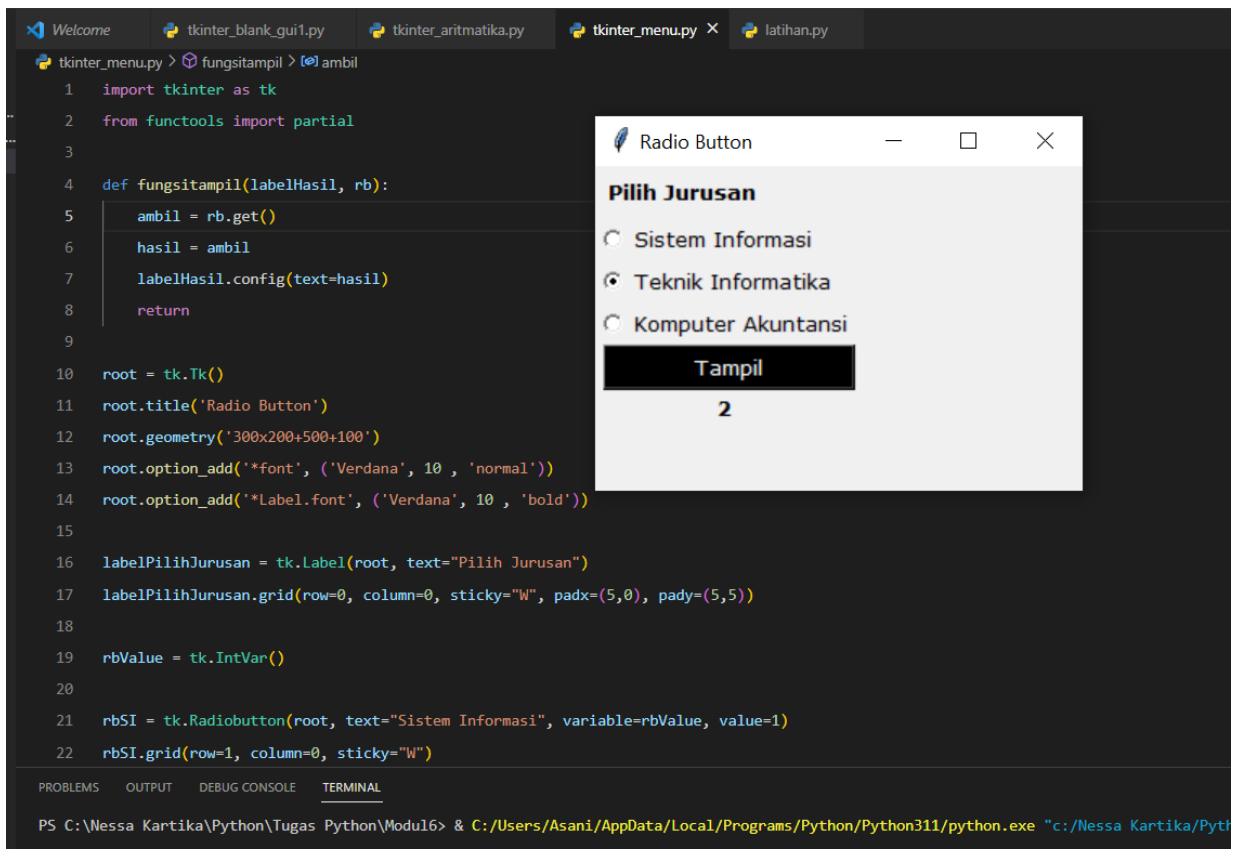
```
1 import tkinter as tk
2 from functools import partial
3
4 def pertambahan(labelHasil, bil1, bil2):
5     b1 = int(bil1.get())
6     b2 = int(bil2.get())
7     hasil = b1 + b2
8     labelHasil.config(text=hasil)
9     return
10
11 root = tk.Tk()
12
13 root.geometry('400x200+500+200')
14
15 root.option_add('*font', ('Verdana', 10, 'normal'))
16
17 root.title('Aritmatika')
18
19 labelBilangan1 = tk.Label(root, text="Masukkan Bilangan 1",)
20 labelBilangan1.grid(row=0, column=0, padx=(10,20))
21 labelBilangan2 = tk.Label(root, text="Masukkan Bilangan 2",)
22 labelBilangan2.grid(row=1, column=0, padx=(10,20))
```

The application window, titled "Aritmatika", is displayed over the code. It features two input fields with values 3 and 4, a "Tambah" button, and a display showing the result 7.

At the bottom, the terminal shows the command to run the application:

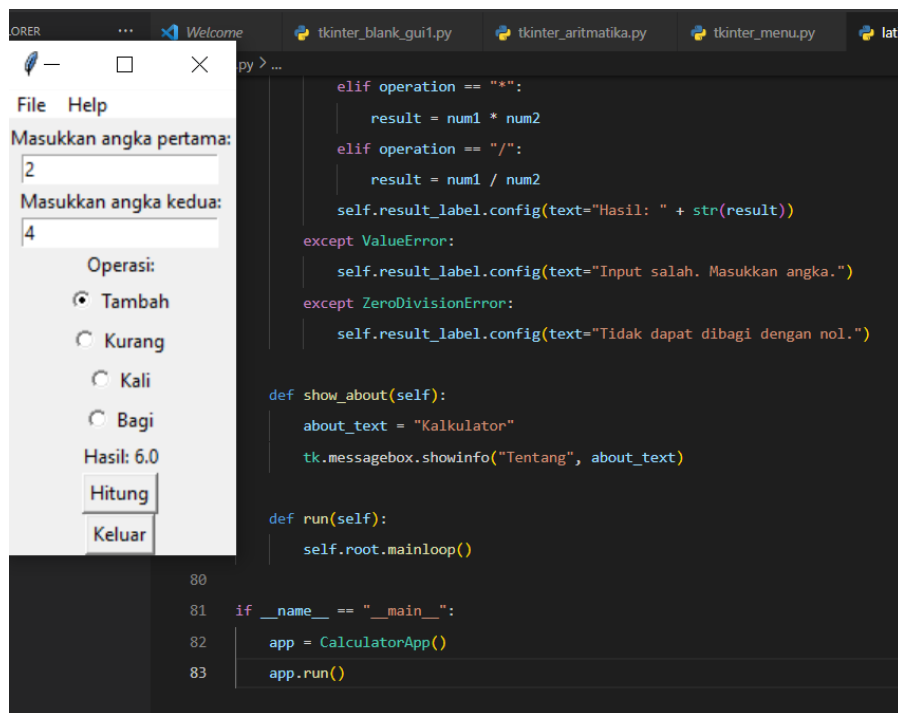
```
PS C:\Nessa Kartika\Python\Tugas Python\Modul6> & C:/Users/Asani/AppData/Local/Programs/Python/Python311/python.exe "c:/Nessa Kartika/Python/Tugas Python/Modul6/tkinter_aritmatika.py"
```

4. Membuat Menu



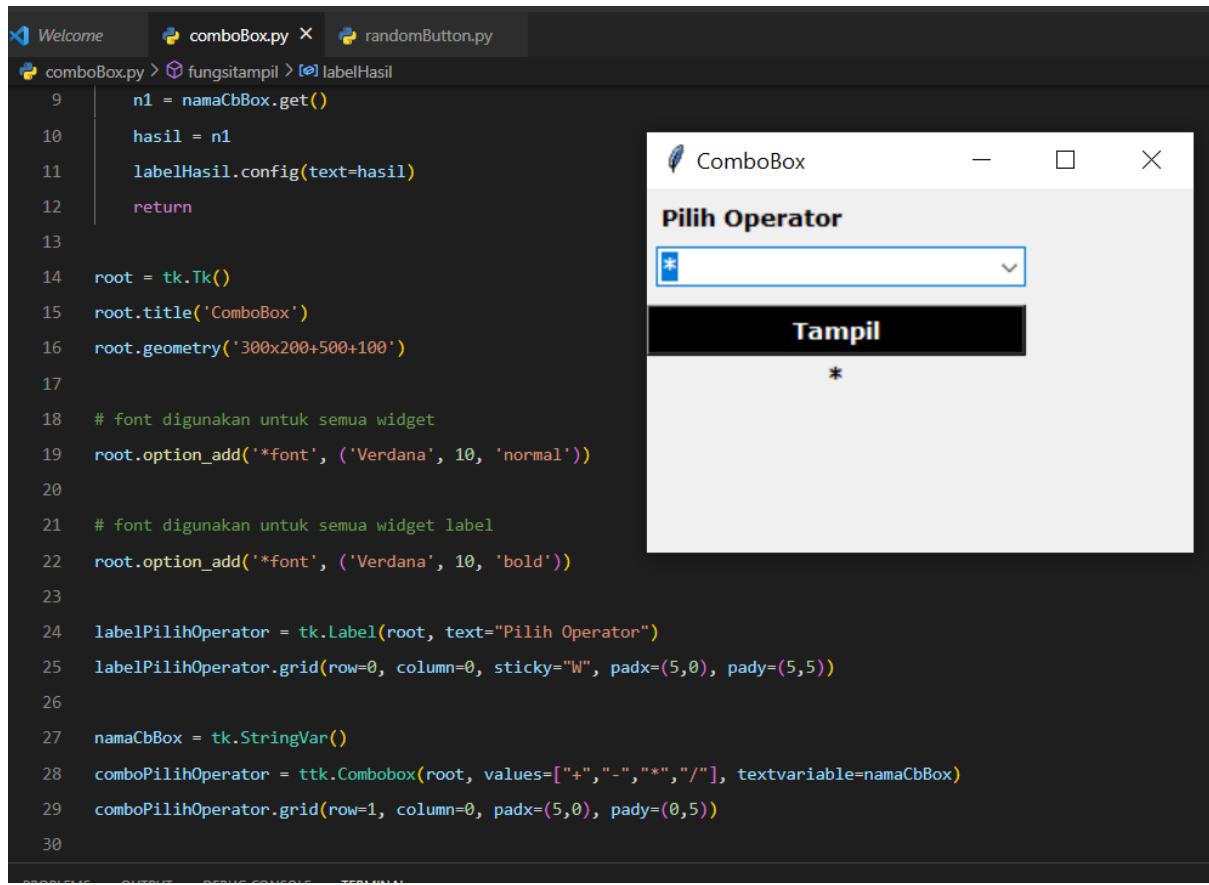
Latihan

Buatlah aplikasi aritmatika pertambahan, pengurangan, perkalian, dan pembagian seperti gambar dibawah.

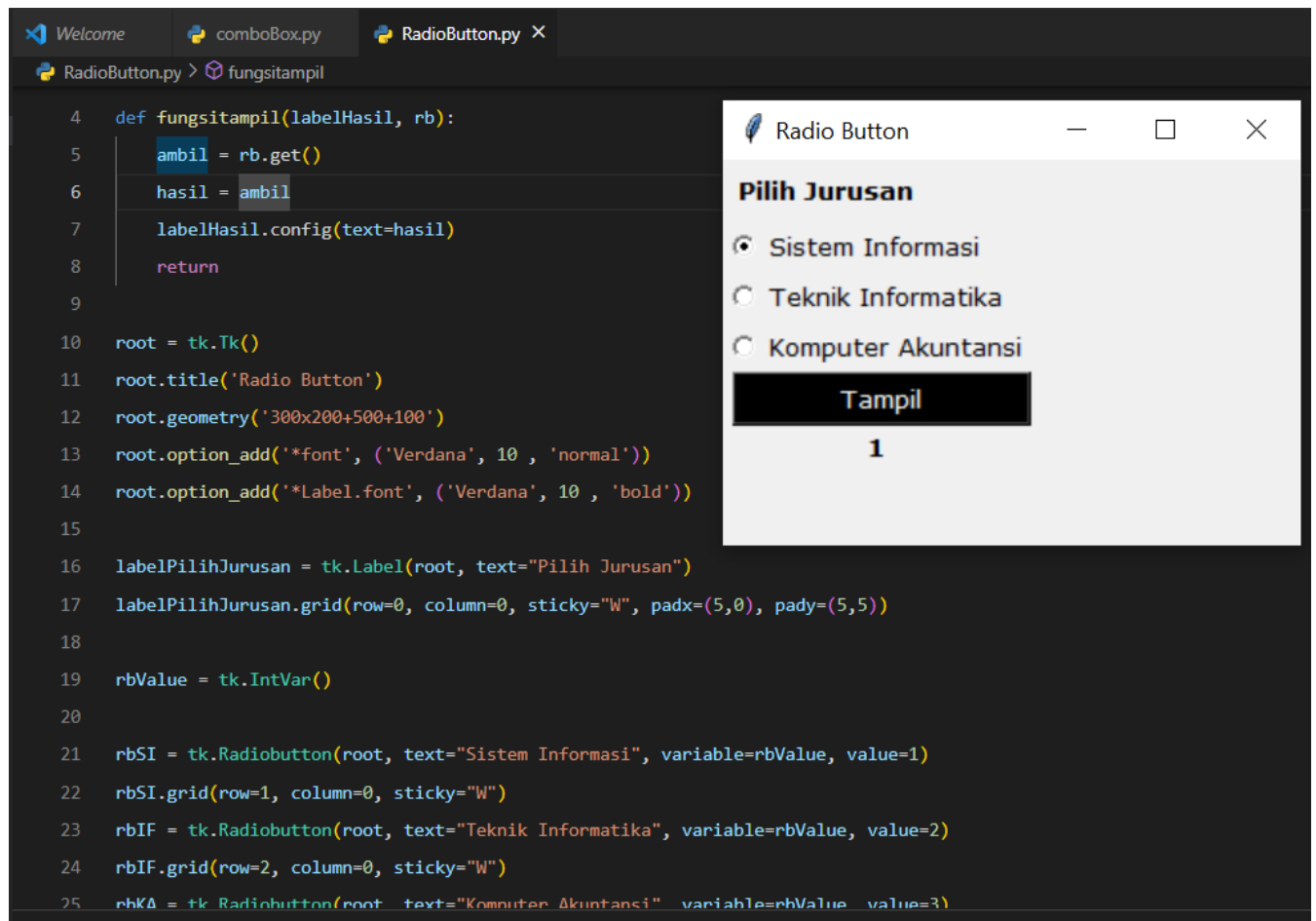


MODUL VII

1. ComboBox



2. Radio Button



The image shows a Python IDE with two windows. The main window, titled 'RadioButton.py', contains the following code:

```
4 def fungsitampil(labelHasil, rb):
5     ambil = rb.get()
6     hasil = ambil
7     labelHasil.config(text=hasil)
8     return
9
10 root = tk.Tk()
11 root.title('Radio Button')
12 root.geometry('300x200+500+100')
13 root.option_add('*font', ('Verdana', 10, 'normal'))
14 root.option_add('*Label.font', ('Verdana', 10, 'bold'))
15
16 labelPilihJurusan = tk.Label(root, text="Pilih Jurusan")
17 labelPilihJurusan.grid(row=0, column=0, sticky="W", padx=(5,0), pady=(5,5))
18
19 rbValue = tk.IntVar()
20
21 rbSI = tk.Radiobutton(root, text="Sistem Informasi", variable=rbValue, value=1)
22 rbSI.grid(row=1, column=0, sticky="W")
23 rbIF = tk.Radiobutton(root, text="Teknik Informatika", variable=rbValue, value=2)
24 rbIF.grid(row=2, column=0, sticky="W")
25 rbKA = tk.Radiobutton(root, text="Komputer Akuntansi", variable=rbValue, value=3)
```

The second window, titled 'Radio Button', displays the graphical user interface. It features a label 'Pilih Jurusan' at the top. Below it are three radio buttons: 'Sistem Informasi' (which is selected), 'Teknik Informatika', and 'Komputer Akuntansi'. At the bottom of the window is a 'Tampil' button. The number '1' is displayed below the 'Tampil' button, indicating the selected value.