

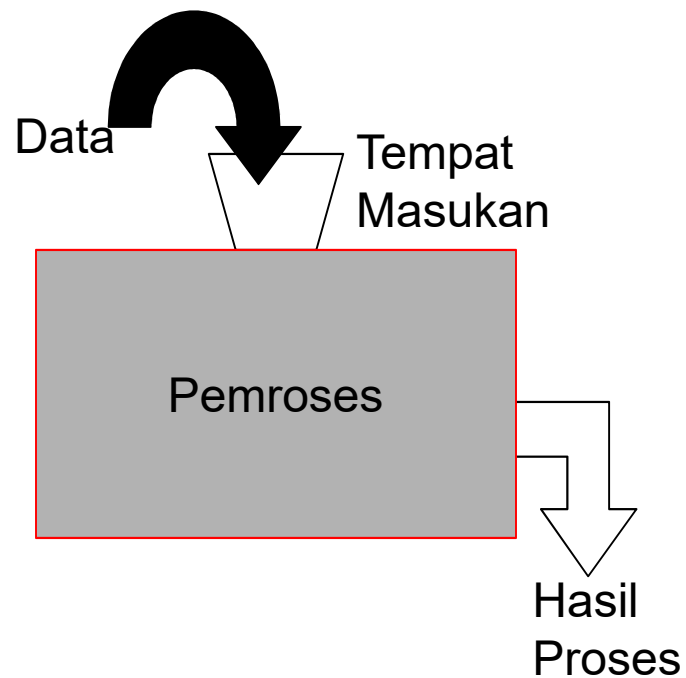
Fungsi

Wijanarto

Konsep/Model

Bayangkan FUNGSI sebagai MESIN YANG DAPAT MENGHASILKAN SUATU OBJECT

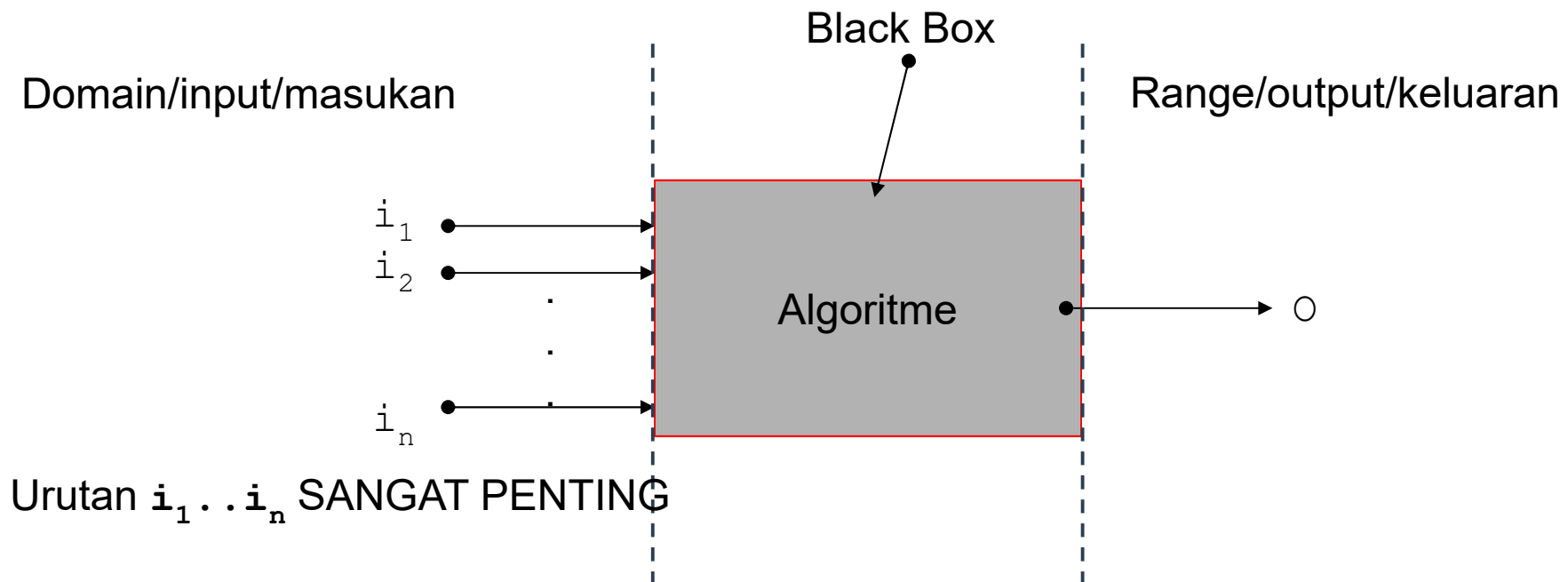
Input ?



Model Matematika

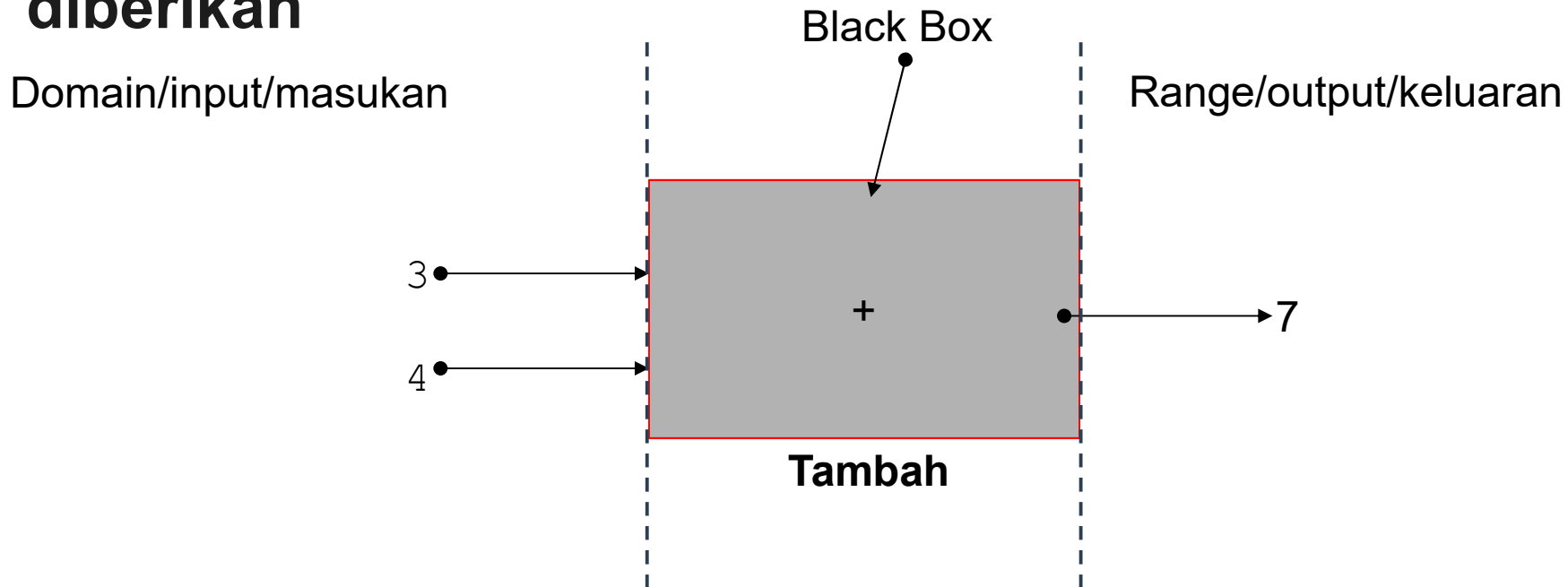
Pemetaan nilai domain ke range, input ke output

$$f : x \rightarrow y$$



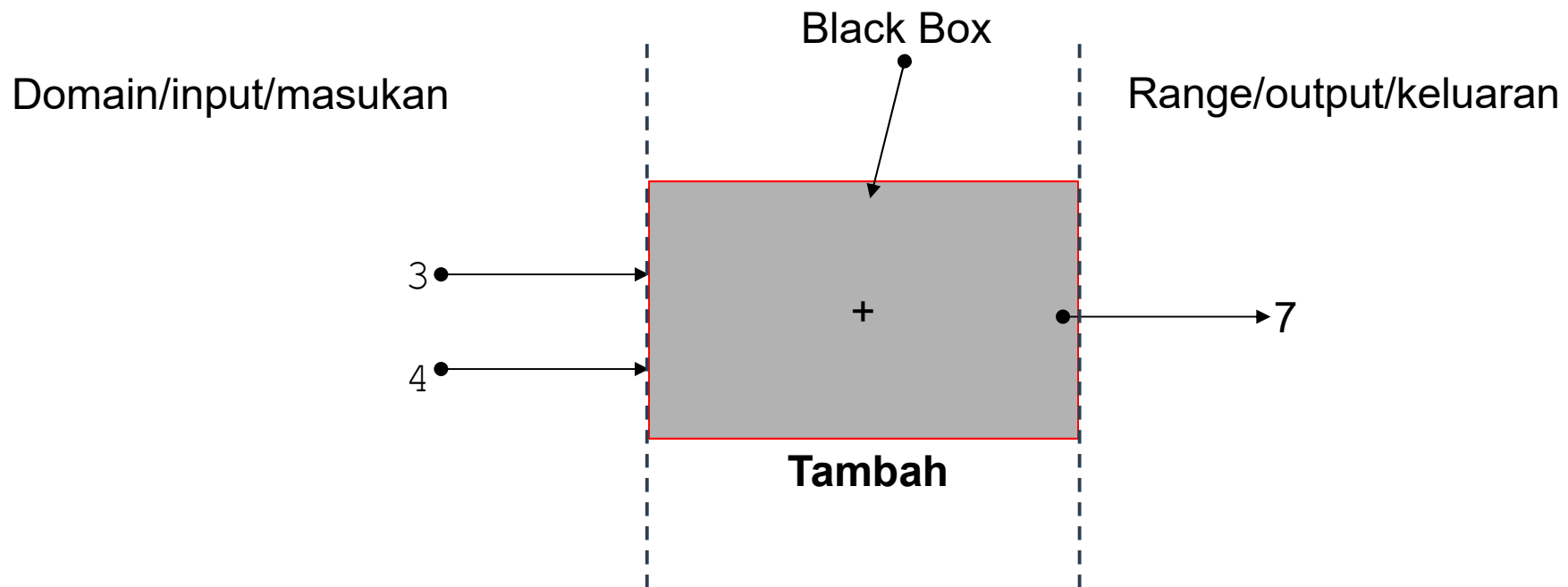
Fungsi dan Bilangan

Black Box merupakan pemroses yang dapat diatur oleh pembuat fungsi (misal, operasi aritmatika $+$). Fungsi Tambah akan mengambil 2 inputan (3 dan 4) dan menghasilkan jumlahan (7) dari kedua input yang diberikan



Keberlakuan Variabel dan Nilai

Semua proses yang terjadi dalam blackbox bersifat lokal bagi fungsi dan hanya berlaku didalamnya (kecuali terdapat atribut global)



Fungsi : Struktur dalam Python

```
def NamaFungsi (list-parameter) :
```

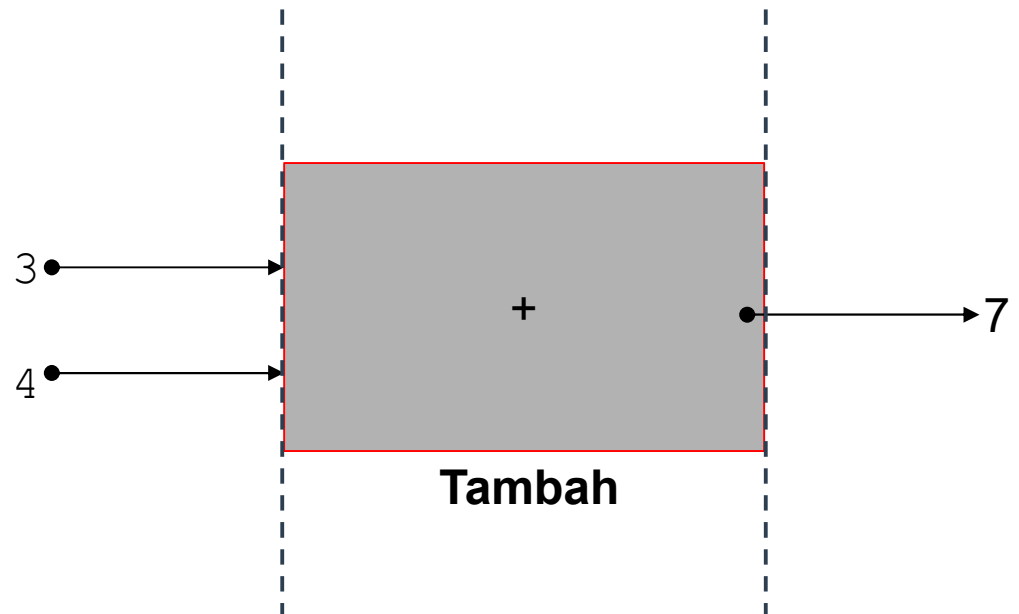
```
    Statemen
```

```
    .
```

```
    .
```

```
    .
```

```
    [return nilai]
```

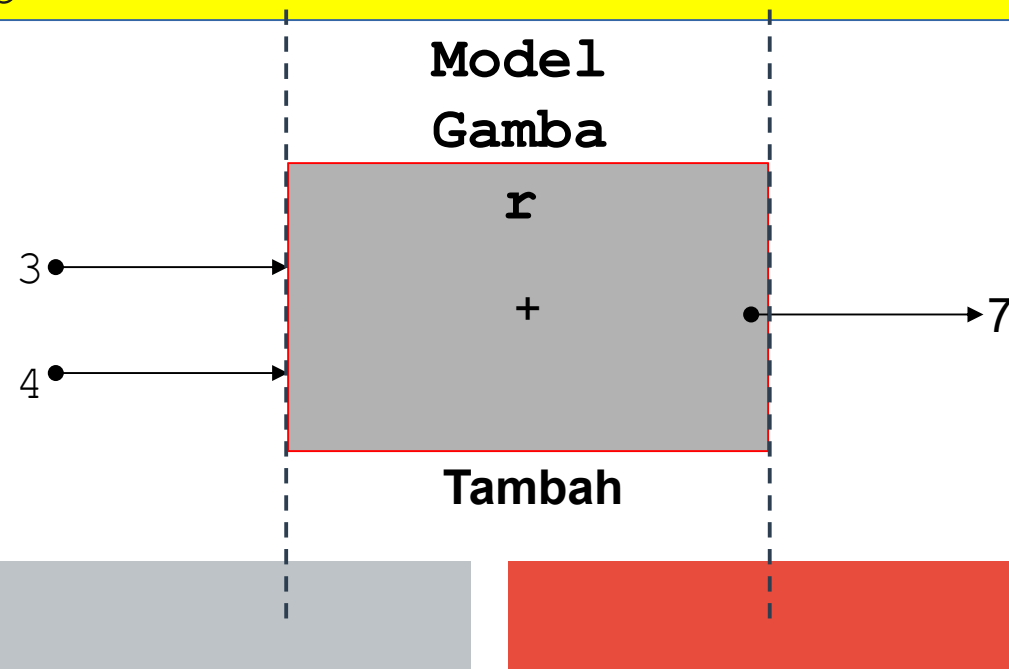


Fungsi : Spesifikasi dan Definisi

Spesifikasi : menulis kebutuhan fungsi (input - proses - output)

Definisi : menulis batasan dan solusi permasalahan

Tambah adalah fungsi dengan 2 variabel sebagai masukan dan akan mengembalikan keluaran jumlah dari variabel tersebut. Jika a dan b adalah parameter fungsi, maka hasil balik fungsi adalah $a+b$



Fungsi : Realisasi dan Aplikasi

```
def Tambah(a,b) :  
    return a+b
```

Realisasi

Realisasi:

Menulis kode berdasarkan **spesifikasi** dan **definisi** yang dibuat sebelumnya.

```
print(Tambah(3,4))  
x=Tambah(3,4)  
y=Tambah(3,4)+x
```

Aplikasi

Aplikasi:

Mengaplikasikan hasil **realisasi** pada **program**. Sering disebut sebagai **function call** atau **invoke function**

Fungsi : Tipe dan Operator

```
def Tambah (a,b) :  
    return a+b  
def isGenap (n) :  
    return n%2==0
```

Tipe:

Karena fungsi dapat menghasilkan suatu nilai, maka fungsi mempunyai tipe sesuai nilai yang dihasilkan

```
x = Tambah ( 3 , 4 )  
#x=7
```

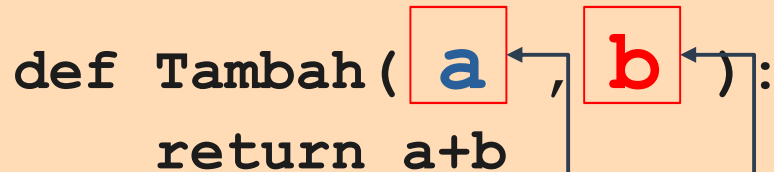
```
genap=isGenap(x)  
#genap=False
```

Operator:

Fungsi juga dapat berperan seperti operator dan dapat dioperasikan, misal aritmatika (+, -, *, /, //) atau logika , relasional (and, or, not, <, >, =, ≤, ≥, ≠)

Fungsi : Argumen Parameter

```
def Tambah ( a , b ) :  
    return a+b
```



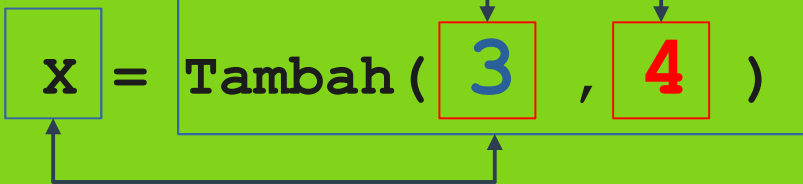
Realisasi:

a dan b adalah parameter FORMAL, yaitu **variabel yang dibuat saat REALISASI**. Boleh sembarang variabel, namun yang mudah dimengerti dan dipahami.

Aplikasi:

Saat aplikasi fungsi, parameter **FORMAL** berubah menjadi **AKTUAL**, yaitu **nilai yang sebenarnya** yang dikirim ke dalam fungsi saat dipanggil.

```
x = Tambah ( 3 , 4 )
```



Fungsi : Argumen Parameter

- Jumlah , jenis dan urutan argumen harus presisi
 - `bla(int,float,boolean,list)`
 - `bla(1,2.5,False,[1,2,3])` → BENAR
 - `bla([1,2,3],1, False,2.5)` → SALAH
- Jenis Argumen
 - **Tunggal**
 - `int,float,boolean,string,list,set,object`
 - **Unpacking operator** atau **Iterable**
 - `*args` → **tuple** atau multiargumen
 - `**kwargs` → **dictionary** atau pasangan **key:value**

Fungsi : Argumen Parameter

```
def fInt(a,b) :  
    return a+b
```

```
print(fInt(3,5))
```

```
def fFloat(a,f) :  
    return a+f
```

```
print(fFloat(1,7.5))
```

```
def fBool(b) :  
    return b
```

```
print(fBool(False))
```

```
def fStr(str) :  
    return str
```

```
print(fStr('STRING'))
```

```
def fList(lst) :  
    return lst
```

```
print(fList(list([1,2,3])))
```

```
def fSet(s) :  
    return s
```

```
print(fSet(set((1,2,3))))
```

Fungsi : Argumen Parameter

```
class MyObj:  
    def __init__(self, arg1='', arg2=0.0):  
        self.nama=arg1  
        self.ipk=arg2
```

```
def fObj(obj):  
    obj.nama=str(input())  
    obj.ipk=float(input())
```

```
o=MyObj()  
fObj(o)
```

```
def fArgs(*args):  
    return args
```

```
data=(1,2,3)  
print(fArgs(*data))
```

```
def fKwargs(**kwargs):  
    return kwargs
```

```
data={'a':1,'b':2}  
print(fKwargs(**data))
```

Latihan Fungsi

1. Buatlah 5 fungsi yang mengembalikan **rerata** dari suatu argumen yang berbeda :
 - a) `rerata1(a,b,c,d,e)→float` , versi **integer**
 - b) `rerata2(a,b,c,d,e)→float`, versi **float**
 - c) `rerata3(s)→float`, versi **string**
 - d) `rerata4(l)→ float`, versi **list**
 - e) `rerata5(*t)→float`, versi **tuple**
2. Buatlah fungsi untuk menentukan apakah terdapat digit bilangan prima dalam argumen bilangan integer `n`, **`is_exist_prime(n)`** .
`print(is_exist_prime(46389))` #True/False
3. Perhatikan model data **Titik** di bawah ini, buatlah fungsi **`createD(dT)`** dan **`createC(oT)`** dimana **`dT`** adalah representasi dari **dictionary** **Titik** dan **`oT`** representasi dari **class** **Titik**. Masing-masing **fungsi** akan **mengambil input absis** dan **ordinat** dari keyboard user lalu mengembalikan **dictionary** dan **object**.

```
d=dict();c=Titik()
d=createD(d);print(d)
c=createC(c);print(c)#obj printed
```

Titik
absis : float
ordinat: float

```
Class Titik:
    def __init__(self,absis=0.0,ord=0.0):
        self.absis=absis
        self.ordinat=ord
```

Fungsi : Tempat/Lokasi → 1 File

- Fungsi direalisasikan dan diaplikasikan dalam file program utama (1 file)
- Letak fungsi berada di atas/sebelum fungsi utama

<pre>#file: testFungsi.py def f1(n): return n**2</pre>	Fungsi f1 di definisikan dalam 1 file yang sama dengan program utama (file yang mengandung fungsi utama dan entry program).
<pre>def main(): x=f1(10)</pre>	→ Fungsi utama
<pre>if __name__ == '__main__': main()</pre>	→ Entry Program

Fungsi : Tempat/Lokasi 2 file

- Fungsi dapat direalisasikan dalam file tersendiri yang terpisah dengan program utama, disebut teknik modular (modul hanya berisi fungsi dan tidak mengandung fungsi utama serta entry program)
- Cara aplikasi fungsi, dengan meng-inklusi/meng-import modul dalam file program utama dengan instruksi :
- **from** <filemodul> **import** [<fungsi>|*]

```
#file: testModul.py
```

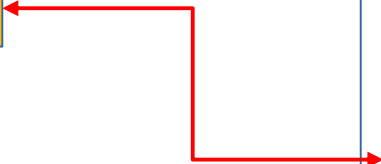
```
from Modul import *
```

```
def main():  
    x=f1(10)
```

```
if __name__ == '__main__':  
    main()
```

```
#file : Modul.py  
import os
```

```
def f1(n):  
    return n**2
```



Variabel, Konstanta, Object Global

```
#file: testModul.py
from Modul import *
import Globals
def main():
    Globals.init() #dipanggil sekali
    f1(10); print(Globals.data)
if __name__ == '__main__':
    main()
```

diinisialisasi

```
#file : Globals.py
import os
# dipanggil sekali dlm driver
def init():
    global data
    data=0
```

dipanggil

```
#file : Modul.py
import Globals
def f1(n):
    Globals.data=Globals.data+n
```

dimanipulasi

tempat menyimpan variabel,

Fungsi : Tempat/Lokasi

Mgg ke-2 dan ke-3

- Fungsi dapat direalisasikan dimanapun dalam program dengan urutan yang jelas.
- Pemisahan dan pengelompokan fungsi menjadi *gugus tugas tertentu* dinamakan modul, dan kumpulan modul disebut *pustaka/library* atau dikenal dengan API (application programming interface)

Fungsi : Tanpa return value (aka Prosedur)

Mgg ke-3

- **Defaultnya** fungsi dalam python mengembalikan/return **None**.
- Jika tidak ada keyword **return**, maka fungsi tetap akan mengembalikan **None**.
- **Prosedur** adalah fungsi yang dapat **menyimpan** nilai dalam **parameter**.

Fungsi dengan return value

```
def test1(a,b):  
    return a+b  
  
print(test1(10,20))  
def test2():  
    a=10;b=20  
    c=a+b  
    return a,b,c  
x,y,z=test2()  
print(x,y,z)
```

Fungsi tanpa return value

```
def test3(a,b):  
    print(a+b)
```

```
test3(10,20)
```

```
global data
```

```
def test4():  
    global data  
    data=data+1  
    print(data)
```

```
test4()  
print(data)
```

Prosedur : Argumen

Mgg ke-3

- **Sebagai Input**

- Mirip fungsi yaitu nilai yang akan dikomputasi untuk ditampilkan (print), bukan disimpan.

- **Sebagai Output**

- Sebagai **penampung hasil komputasi (output)**, mungkin **bagian** dari yang *dikomputasi (input)*.

Argumen input

```
def test1(a,b):  
    print(a+b)
```

```
test1(10,20)
```

Argumen input/output

```
global d;d=0  
  
def test3(a,b,o):  
    global d  
    d=o  
    d=a+b
```

```
test3(10,20, d)
```

```
print(d)
```

Argumen output

```
global d;d=0  
  
def test3(o):  
    global d  
    d=o  
    d=20*d
```

```
test3(d)
```

```
print(d)
```



Fungsi : Passing Argumen object (sbg output)

- Argumen **iterable** (list,set,class)
 - `Bla(list,set,class)`
- Argumen **unpacking operator** (tuple dan dictionary)
 - `Bla(*tuple)`
 - `Bla(**dict)`

Versi Parameter list,set,class

```
L=[1,2,3]
S={1,2,3}
C=Point(2.0,4.0)
def test5(data):
    if isinstance(data,list):
        print('list:',data)
    elif isinstance(data,set):
        print('set:',data)
    elif isinstance(data,class):
        print('class:',data)
```

Parameter tuple dan dictionary

```
T=(1,2,3)
def test6(*data):
    print(data)
test6(*T)

D={'a':19,'b':23,'c':67}
def test7(**data):
    print(data)
test7(**D)
```

Fungsi : Contoh lain

```
class Siswa:
    def __init__(self, arg1, arg2):
        self.nama = arg1
        self.usia = arg2
def test6(objSiswa):
    objSiswa.nama='Dewa'
    objSiswa.usia=21
data=Siswa('Adi',19)
print(data)
test6(data); print(data)
```

```
T={1,2,3}
def test5(data):
    if isinstance(data,tuple):
        for t in data:
            print(data.Index(t),t)

for t in range(len(data)):
    print(t,data[t])
for t in enumerate(data):
    print(t)
```

```
L=[1,2,3]
def test5(data):
    if isinstance(data,list):
        for x in data:
            print(x)
```

```
S={1,2,3}
def test5(data):
    if isinstance(data,set):
        for s in data:
            print(s)
```

```
D={'a':1,'b':2,'c':3}
def test5(data):
    if isinstance(data,dict):
        for key in data:
            print(key)
for key in data:
    print(key,data[key])
for k,v in enumerate(data):
    print(k,v)
for item in data.items():
    print(item)#(key,value)
```

Latihan Procedure : list, set

1. Jika telah terdefinisi variabel **global data**, buatlah **prosedur rerata (lst)** yang akan merubah menghitung rerata *lst* dan menyimpan reratanya dalam variabel **data**.
2. Modifikasi soal no 1 menjadi prosedur **rerata (lst, data)** , dimana **data** adalah argumen output yang akan menyimpan **rerata lst**.
3. Jika terdefinisi variabel **global data_set**, yaitu suatu **set**, buatlah **procedure createGanjil (lst, data_set)** , yang akan menyaring bilangan ganjil yang **unik** dari *lst* dan menyimpannya kedalam **data_set** . Perhatikan **set**, tidak terurut, tidak terindex, unik, sementara **list**, terurut , terindex. Jika *lst*= [1,1,2,3,3,4,5,6,7,7,8,9,9], apakah isi dari **data_set** ?

Latihan Procedure : tuple, dictionary

4. Jika terdefinisi variabel **global data_dict**, yaitu suatu **dictionary**, buatlah prosedur yang menerima input suatu **tuple** lalu **mengolahnya** menjadi suatu pasangan **key:value** yang disimpan dalam variabel **global data_dict**, dengan nama prosedur **makeDict(t)** . Data tuple yang akan diolah berisi nilai ipk mahasiswa yaitu **ipk = (2.75,3.01,4.00,3.56,3.74,3.88)** menjadi pasangan **{ 'nama1': 2.75, 'nama2': 3.01, 'nama3': 4.0, 'nama4': 3.56, 'nama5': 3.74, 'nama6': 3.88 }**, perhatikan **key** adalah **'nama'** diikuti oleh *index tuple* yang dimulai dari angka **1**, sehingga menjadi **'nama1'** .