Functions, Iterators, and Generators

Oleh : Agus Priyanto, M.Kom

INSTITUT TEKNOLOGI TELKOM PURWOKERTO
Smart, Trustworthy, And Teamwork

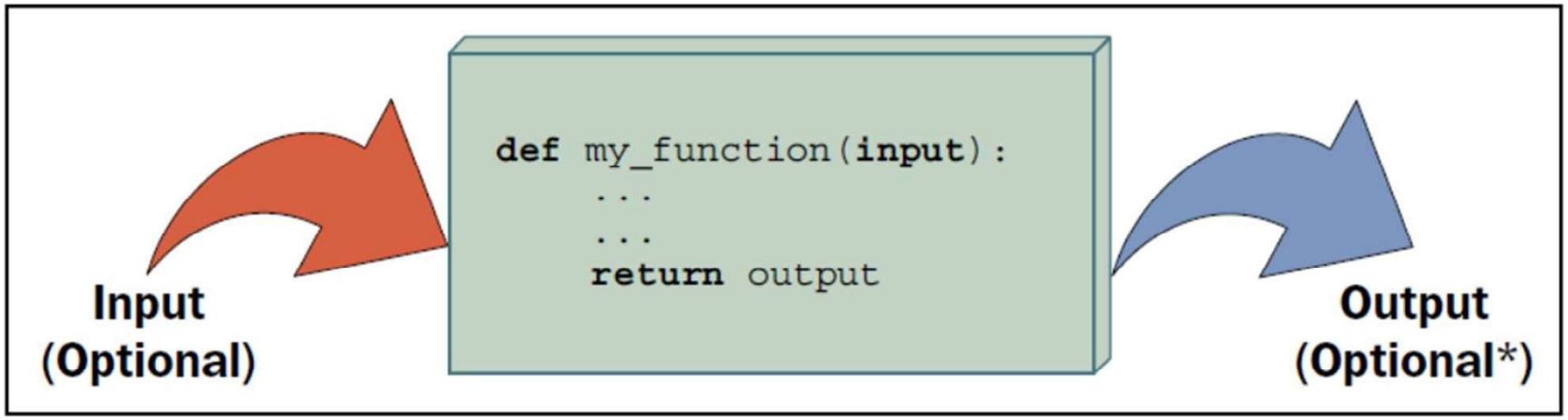
Outline Materi

- Functions
- Iterators
- Generators

Functions

- Fungsi adalah grup/blok program untuk melakukan tugas tertentu yang berulang

- Fungsi membuat kode program menjadi *reusable*, artinya hanya di definisikan sekali saja, dan kemudian bisa digunakan berulang kali dari tempat lain di dalam program



Sintak :

```
def function_name(parameters):
    statement(s)
    return [expression]
```

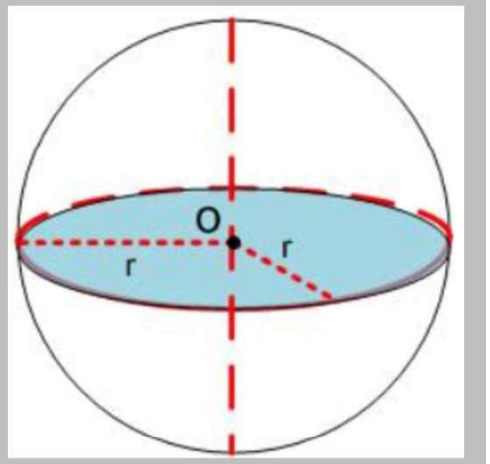
Contoh :

```
>>> x = [1, 2, 3]
>>> def fungsi_sederhana(x):
       x[1]=42
>>> fungsi_sederhana (x)
>>> print(x)
[1, 42, 3]
>>>
```

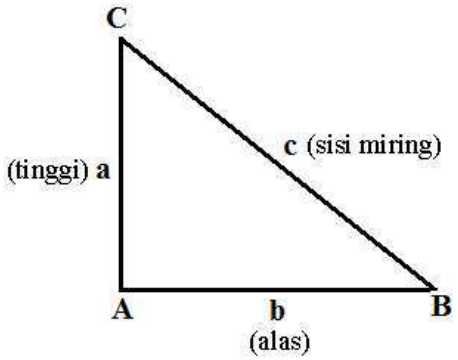
```
>>> import math
>>> dir (math)
['__doc__', '__loader__', '__name__', '__package__', '__spec__', 'acos',
   'acosh', 'asin', 'asinh', 'atan', 'atan2', 'atanh', 'ceil',
   'copysign', 'cos', 'cosh', 'degrees', 'e', 'erf', 'erfc', 'exp',
   'expm1', 'fabs', 'factorial', 'floor', 'fmod', 'frexp', 'fsum',
   'gamma', 'gcd', 'hypot', 'inf', 'isclose', 'isfinite', 'isinf',
   'isnan', 'ldexp', 'lgamma', 'log', 'log10', 'log1p', 'log2', 'modf',
   'nan', 'pi', 'pow', 'radians', 'remainder', 'sin', 'sinh', 'sqrt',
   'tan', 'tanh', 'tau', 'trunc']
>>> math.pi
3.141592653589793
>>> def volume (r):
   v = (4.0/3.0) * math.pi * r**3



>>> volume(2)
>>> 33.510321638291124
```



```
>>> def triangle_area (b, h):
...     return 0.5 * b * h
...
>>> triangle_area (3, 6)
9.0
```



```
>>> def connect(**options):
        conn_params = {
                'host': options.get('host', '127.0.0.1'),
                'port': options.get('port', 5432),
                'user': options.get('user', ''),
                'pwd': options.get('pwd', ''),
                }
        print(conn_params)


>>> connect()
{'host': '127.0.0.1', 'port': 5432, 'user': '', 'pwd': ''}

>>> connect(host='127.0.0.42', port=5433)
{'host': '127.0.0.42', 'port': 5433, 'user': '', 'pwd': ''}

>>> connect(port=5431, user='ftti', pwd='nimda')
{'host': '127.0.0.1', 'port': 5431, 'user': 'ftti', 'pwd': 'nimda'}
>>>
```

Konversi Fungsi Sederhana

```
>>> price = 100
>>> final_price1 = price * 1.2
>>> final_price2 = price + price / 5.0
>>> final_price3 = price * (100 + 20) / 100.0
>>> final_price4 = price + price * 0.2
>>> print (final_price1)
120.0
>>>
```

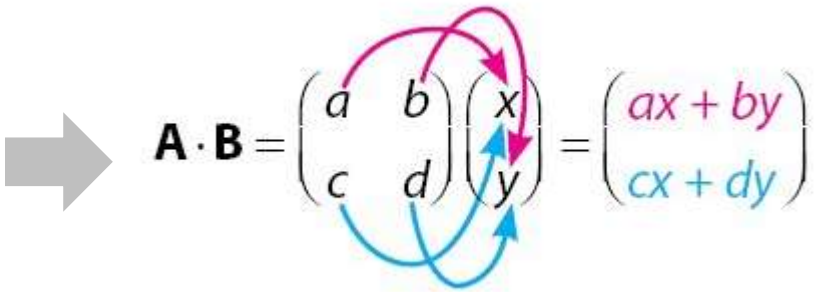
```
>>> def calculate_price_with_vat(price, vat):
        return price * (100 + vat) / 100
>>>
>>> calculate_price_with_vat (200, 10)
220.0
>>>
```

Nilai Minimal

```
>>> def minimum(*n):
        if n:
                mn = n[0]
                for value in n[1:]:
                        if value < mn:
                                mn = value
                print(mn)
>>>
>>>>>> minimum(1, 3, -7, 9)
-7
>>> minimum(2, 3, 4, 0, 10)
0
>>>
```

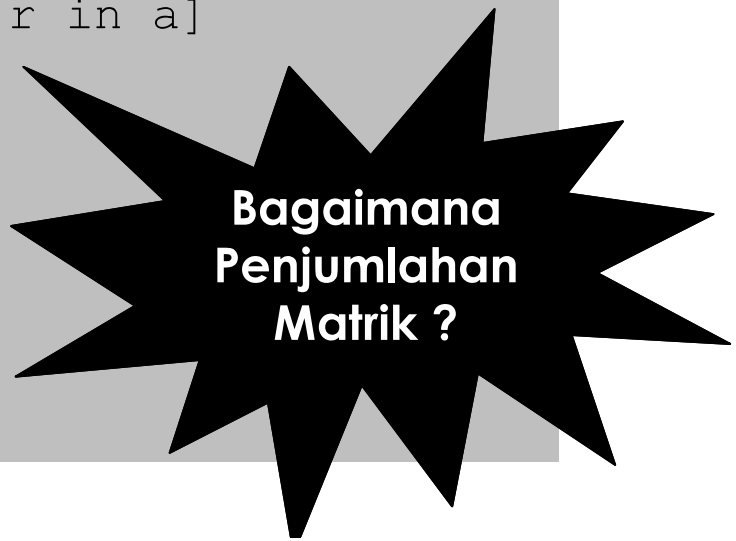
Matrik

$$\begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix} \cdot \begin{pmatrix} 5 & 1 \\ 2 & 1 \end{pmatrix} = \begin{pmatrix} 9 & 3 \\ 23 & 7 \end{pmatrix}$$



Fungsi zip() berfungsi untuk membuat iterator berisi item dari dua buah iterable atau lebih dalam bentuk tuple.

```
>>> def matrix_mul(a, b):
        return [[sum(i * j for i, j in zip(r, c))
                for c in zip(*b)]for r in a]
>>>
>>> a = [[1, 2], [3, 4]]
>>> b = [[5, 1], [2, 1]]
>>> c = matrix_mul(a, b)
>>> print (c)
[[9, 3], [23, 7]]
>>>
```



Unguided

- Buatlah fungsi untuk menghitung rataan dari dua buah bilangan ?

  Output :

  >>> rataan (200, 350)

  275.0 ---> (200 +350)/2

- Buatlah fungsi untuk menjumlahkan range dari dua buah bilangan ?

  Output :

  >>> hasil =summ(1, 5)

  >>> hasil

  15 ---> (1+2+3+4+5)

```
>>> def rataan(bil1, bil2):
      rataan= (bil1+ bil2)/2
      print (rataan)
```

```
>>> def summ(bil1, bil2):
      total = 0
      for bil_hasil in list (range(bil1, bil2+1)):
            total = total + bil_hasil
      return total
```

Iterators

- Iterator didalam python sebenarnya sudah diterapkan di dalam looping for, list comprehension, generator, dan lain – lain, hanya saja tidak tampak secara langsung

- Iterator hanyalah suatu objek yang dapat dilakukan iterasi atau looping.

- Objek akan mengembalikan data, satu data per satu waktu

- Iterator Python adalah kelas yang mendefinisikan sebuah fungsi __iter__()

- Sebagian besar objek Python bersifat iterable, artinya kita dapat melakukan loop terhadap setiap elemen dalam objek tersebut

```
>>> list_ku = [4, 10, 3, 0, 7]
>>> test_iter = iter (list_ku)
>>> print (next(test_iter))
4
>>> print (next(test_iter))
10
>>> print (next(test_iter))
3
>>> print (next(test_iter))
0
>>> print (next(test_iter))
7
>>>
```

```
>>> for angka in list_ku :
        print (angka)


4
10
3
0
7
>>>
```

Generators

- Generator adalah fungsi yang mengembalikan objek generator yang bisa kita panggil method next(), sedemikian rupa sehingga setiap pemanggilan mengembalikan nilai berikutnya

- Fungsi normal pada Python menggunakan keyword return untuk mengembalikan suatu nilai, tapi generator menggunakan keyword yield untuk mengembalikan nilai

```
>>> def cth_generator(l):
     total = 0
     for n in l:
       yield total
       total += n
>>>
>>> test_generator = cth_generator ([10,20,30,40,50])
>>> print(next(test_generator ))
0
>>> print(next(test_generator ))
10
>>> print(next(test_generator ))
30
>>>
```

Menggunakan lists, dicts, dan sets

- Didalam Python, sekuen objek seperti list adalah iterable

- Kita dapat menggunakan tuple () untuk dalam menampilkan output yang berasal dari generate fungsi maupun operator

```
>>> range (10)
range(0, 10)

>>> [range(10)]
[range(0, 10)]

>>> [x for x in range (10)]
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]

>>> list (range (10))
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
>>>
```


Backspace
Thank You
Page Up
Home
Page Down
Insert
Shift