Bab IV

Functions

Dalam konteks pemrograman, *function* atau fungsi didefinisikan sebagai sekumpulan baris perintah atau kode yang dikelompokkan menjadi satu kesatuan yang akan dieksekusi ketika dia dipanggil. Sebuah *function* bisa menerima parameter, bisa mengembalikan suatu nilai, dan bisa dipanggil berkali-kali secara independen. Tujuan dari *function* adalah untuk memecah program yang besar dan kompleks menjadi bagian-bagian kecil dengan tugasnya masing-masing.

A. Pembuatan Function

Dalam bahasa pemrograman Python, *function* dibuat dengan sintaks sebagai berikut:

```
def <function_name>(parameters):
  statements
```

Keterangan:

1. *def* adalah sintaks yang mendefinisikan bahwa blok kode program adalah sebuah *function*
2. <function_name> adalah nama dari *function*
3. (parameters) adalah nilai-nilai (satu atau lebih) yang dikirimkan ke dalam *function* yang akan dieksekusi di dalam *function body*
4. *statements* adalah kumpulan perintah yang akan dieksekusi ketika *function* dipanggil

Catatan:

Blok kode program di dalam Python didefinisikan dengan indentasi

Contoh Function:

```
def hello_world():
  print('Hello python! Hello World!'
```

B. Pemanggilan Function

Function dipanggil dengan menuliskan nama fungsinya diikuti tanda kurung () sebagai berikut:

```
hello_world()
```

Output:

```
Hello python! Hello World!
```

Function juga dapat digunakan atau dipanggil berkali-kali:

```
hello_world()

hello_world()

hello_world()
```

Output:

```
Hello python! Hello World!

Hello python! Hello World!

Hello python! Hello World!
```

C. Function dengan Parameter

Sebuah *function* dapat menerima satu atau lebih parameter atau argumen yang merupakan nilai yang dikirimkan ke dalam fungsi untuk dieksekusi dalam *function body*.

```
def welcome(name):

  print(f'Hello {name}, welcome!')
```

Keterangan:

name adalah sebuah parameter yang dikirimkan ke dalam fungsi untuk ikut dieksekusi pada perintah *print*()

```
welcome('Human')
```

```
welcome('Robot')

welcome('Alien')
```

Output:

```
Hello Human, welcome!

Hello Robot, welcome!

Hello Alien, welcome!
```

D. Function dengan Parameter Wajib

Parameter di dalam Python dapat bersifat wajib, artinya parameter tersebut harus diisi. Perhatikan contoh berikut:

```
def introduce(name, origin):

  print(f"I'm {name} from {origin}")
```

Parameter *name* dan *origin* adalah parameter wajib. Jika *function* tersebut dipanggil dengan kedua parameternya:

```
introduce('Human', 'Earth')
```

Output:

```
I'm Human from Earth
```

Jika *function* tersebut dipanggil hanya dengan salah satu parameternya:

```
introduce('Human')
```

Output:

```
Exception has occurred: TypeError

introduce() missing 1 required positional argument:
'origin'
```

Terjadi *exception* karena terdapat parameter tidak lengkap atau hilang.

E. Function dengan Parameter Opsional

Parameter di dalam Python juga dapat bersifat opsional, artinya parameter yang dapat tidak diisi karena memiliki nilai *default*. Perhatikan contoh berikut:

```
def introduce (name, origin='Somewhere'):
  print(f"I'm {name} from {origin}")
```

Parameter *name* adalah parameter wajib sedangkan *origin* adalah parameter opsional. Jika *function* tersebut dipanggil dengan parameter wajibnya:

```
introduce('Human')
```

Output:

```
I'm Human from Somewhere
```

Jika *function* tersebut dipanggil dengan parameter wajib dan parameter opsionalnya:

```
introduce('Alien', 'Space')
```

Output:

```
I'm Alien from Space
```

F. Function dengan Parameter Berurut dan Tidak Berurut

Sebuah *function* dapat memiliki parameter opsional yang lebih dari satu, artinya terdapat beberapa parameter yang dapat diisi ataupun tidak diisi karena memiliki nilai default. Perhatikan contoh berikut:

```
def introduce (name, origin='Somewhere', power='no
power'):
  print(f"I'm {name} from {origin}, I have {power}")
```

name adalah parameter wajib, sedangkan origin dan power adalah parameter opsional. Jika *function* tersebut dipanggil dengan parameternya secara lengkap dan berurut:

```
introduce('Human', 'Earth', 'a mind')
```

Output:

```
I’m Human from Earth, I have a mind
```

Jika *function* tersebut dipanggil dengan parameternya secara lengkap tapi tidak berurut:

```
introduce(‘Human’, ‘a mind’, ‘Earth’)
```

Output:

```
I’m Human from a mind, I have Earth
```

Jika *function* tersebut dipanggil dengan parameternya secara tidak lengkap tapi berurut:

```
introduce(‘Human’, ‘Earth’)
```

Output:

```
I’m Human from Earth, I have no power
```

Jika *function* tersebut dipanggil dengan parameternya secara tidak lengkap dan tidak berurut:

```
def introduce(‘Human’, ‘a mind’)
```

Output:

```
I’m Human from a mind, I have no power
```

Dari keempat program diatas diketahui bahwa yang menghasilkan output yang benar adalah function yang parameternya dipanggil secara berurutan. Sedangkan, *function* yang parameternya dipanggil secara tidak berurutan menghasilkan output yang salah.

Pemanggilan *function* dengan parameter yang tidak berurutan dapat dilakukan dengan mendefinisikan nama parameter beserta nilainya untuk menghasilkan output yang benar. Perhatikan contoh berikut:

```
introduce(name='Alien', power='a strength',
origin='Space')
```

Output:

```
I'm Alien from Space, I have a strength
```

G. Function yang Mengembalikan dan Tidak Mengembalikan Nilai

Ditinjau dari segi pengembalian nilai, *function* dapat dikategorikan menjadi 2 bagian:

1. Function yang Mengembalikan Nilai

Function yang mengembalikan nilai adalah jenis *function* yang jika dipanggil akan mengembalikan nilai sebagai hasil dari pemanggilan *function* tersebut. Nilai tersebut dapat digunakan untuk perintah lebih lanjut seperti menyimpan nilai dalam variabel, melakukan perhitungan, atau mencetak nilai, dsb.

Perhatikan contoh berikut:

```
def add2values(a, b):
  result = a + b
  return result
```

Dalam sebuah *function*, untuk mengembalikan nilai digunakan sintaks *return*. Jika *statement return* telah dieksekusi pada sebuah *function*, maka semua proses yang ada di dalam blok code function tersebut akan berhenti dieksekusi.

```
#Case01 (just call the function)
add2values(1, 2)
#Case02 (store the function in a variable)
a = add2values(1, 2)
#Case3 (print the function)
print("1 + 2 =", add2values(1, 2))
```

Output:

```
#Output01


#Output02


#Output03
1 + 2 = 3
```

2. Fungsi yang Tidak Mengembalikan Nilai

Fungsi yang tidak mengembalikan nilai adalah fungsi yang ketika dipanggil hanya mengeksekusi perintah pada *body function* tanpa mengembalikan nilai yang ditandai dengan tidak adanya *statements* return. Perhatikan contoh berikut:

```
def add2values(a, b):
  result = a + b
  print(a, '+', b, '=', result)

add2values(1, 2)
```

Output:

```
1 + 2 = 3
```

H. Function Rekursif

Function rekursif adalah *function* yang memanggil dirinya sendiri dalam *function body*-nya yang menyebabkan efek perulangan. Perulangan ini bisa berhenti ketika kondisi tertentu tercapai, atau bisa juga bersifat tak terbatas, atau mungkin bahkan bisa menimbulkan error karena pemanggilan fungsi yang tak ada habisnya.

Function rekursif dengan perulangan terbatas;

```
def show_intervals(start, end):
  print(start)
  start = start + 1

  if(start <= end):
     show_intervals(start, end)

show_intervals(1, 10)
```

Output:

```
1
2
3
. . .
9
10
```

Function rekursif dengan perulangan tanpa batas;

```
def show_multiple(value):
   print(value)
   value = value + value
   show_multiple(value)
show_multiple(2)
```

Output:

```
2

4

6

8

10

. . .

. . .

. . .

RecursionError: maximum recursion depth exceeded while
calling a Python object
```