



# Fungsi - Argumen dan Parameter

## Argumen yang dapat dikirimkan pada fungsi

Sebagai nilai yang akan dimasukkan pada saat fungsi dipanggil, ada dua jenis argumen:

- Keyword Argument, yakni argumen yang disertai identifier atau nama parameter yang secara eksplisit disebutkan. Hal ini termasuk jika kita mengirimkan nilai melalui dictionary yang diawali dua tanda \* (\*\*).
- Positional Argument, yakni argumen selain keyword argument. Jika kita mengirimkan variabel bersifat iterable, maka harus diawali tanda \*

Contoh:

### Keyword Argument

```
1. daftar(tanggal=1, bulan='Januari', tahun=2020)
2. daftar(**{'tanggal': 1, 'bulan': 'Januari', 'tahun': 2020})
```

### Positional Argument

```
1. daftar(1, 'Januari', 2020)
2. daftar(*(1, 'Januari', 2020))
```

Sintaksis prefix \* digunakan sebagai penanda iterable di Python, sedangkan prefix \*\* digunakan sebagai penanda kontainer/dictionary.

Kontainer (Dictionary) ini bisa bersifat opsional, artinya tidak wajib diisi (boleh kosong), jika memang tidak ada argumen yang perlu ditambahkan. Pada saat diisi, seperti layaknya Dictionary dapat memiliki jumlah/panjang yang dinamis, dengan pasangan kunci-nilai (key-value) yang bervariasi.

## Susunan/Urutan Parameter Fungsi

Terdapat [5 kemungkinan susunan/urutan parameter fungsi](#) menurut dokumentasi Python:

- *positional-or-keyword*: Anda bisa menuliskan argumen sebagai keyword argument atau positional argument.

```
1. def kali(nilai1, nilai2=None): ...
```



DIBANTU



nilai1 dan nilai2 merupakan positional only (harus diletakkan pada posisi tersebut):

```
1. def (nilai1, nilai2, /, nilai3): ...
```

- *keyword-only*: Anda menentukan bahwa argumen tertentu harus disupply dalam bentuk keyword argument. Hal ini dilakukan dengan cara mendeklarasi satu buah var-positional argument diikuti tanda \*. Seperti pada contoh berikut, nilai2 dan nilai3 merupakan keyword-only (harus dikirim dengan keyword):

```
1. def func(arg, *, kw_only1, kw_only2): ...
```

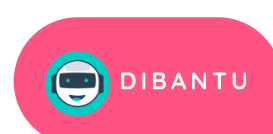
- *var-positional dan var-keyword*: Anda menentukan bahwa ada beberapa positional argument dan ada beberapa keyword argument yang akan Anda proses. var-positional ditandai dengan awalan \* (iterable) dan var-keyword ditandai dengan awalan \*\* (dictionary). Contohnya dengan \*args yang bersifat var-positional dan \*\*kwargs yang berupa var-keyword.

```
1. def func(*args, **kwargs): ...
```

Jika ada argumen posisi dinamis dan argumen kata kunci (keyword) dinamis, maka urutannya adalah argumen posisi dahulu, baru argumen kata kunci. Contoh penggunaan var-positional dan var-keyword pada sebuah berkas python adalah sebagai berikut:

```
1. def printinfo(*args, **kwargs):
2.     for a in args:
3.         print('argumen posisi {}'.format(a))
4.     for key, value in kwargs.items():
5.         print('argumen kata kunci {}:{}'.format(key, value))
6.
7.
8. # Panggil printinfo
9. printinfo()
10. printinfo(1, 2, 3)
11. printinfo(i=7, j=8, k=9)
12. printinfo(1, 2, j=8, k=9)
13. printinfo(*(2, 3), **{'i':7, 'j':8})
```

Output:





argumen posisi 1

argumen posisi 2

argumen posisi 3

argumen kata kunci i:7

argumen kata kunci j:8

argumen kata kunci k:9

argumen posisi 1

argumen posisi 2

argumen kata kunci j:8

argumen kata kunci k:9

argumen posisi 2

argumen posisi 3

argumen kata kunci i:7

argumen kata kunci j:8

## Fungsi Anonim

Fungsi Anonim (anonymous) tidak dideklarasikan seperti halnya fungsi pada umumnya dengan kata kunci `def`, melainkan menggunakan kata kunci (keyword) `lambda`. Sebuah fungsi `lambda` dapat menerima argumen dalam jumlah berapa pun, namun hanya mengembalikan satu nilai `expression`. Fungsi `Lambda` tidak dapat memuat perintah atau ekspresi lainnya, misalnya tidak bisa melakukan `print`.

Fungsi `lambda` bersifat mandiri, memiliki namespace-nya sendiri, dan tidak dapat mengakses nilai apapun selain yang berada dalam parameter list dan variabel global. Meskipun mirip, `Lambda` tidak dapat disamakan dengan `inline statement` pada bahasa `C/C++`.

Sintaks:

```
1. lambda [arg1 [,arg2,.....argn]]:expression
```

Contoh penggunaannya jika kita bandingkan dengan fungsi `kali` yang berada di modul sebelumnya

```
1. kali = lambda nilai1, nilai2: nilai1 * nilai2
2. print ("Hasil : ", kali( 11, 21 ))
3. print ("Hasil : ", kali( 2, 2 ))
```

Hasilnya



DIBANTU



masih . 4

< [Sebelumnya](#)

[Selanjutnya](#) >



Dicoding Space  
Jl. Batik Kumeli No.50, Sukaluyu,  
Kec. Cibeunying Kaler, Kota Bandung  
Jawa Barat 40123



Decode Ideas  
Discover Potential

[Tentang Kami](#)

- [Blog](#)
- [Reward](#)
- [Showcase](#)
- [Hubungi Kami](#)
- [FAQ](#)

Penghargaan



© 2022 Dicoding | Dicoding adalah merek milik PT Presentologics, perusahaan induk dari PT Dicoding Akademi Indonesia.

[Terms](#) • [Privacy](#)



DIBANTU