



# FUNGSI

(Part 1)

# Fungsi

- Fungsi adalah blok program dengan tujuan/tugas khusus/tertentu yang dapat digunakan/dipanggil pada saat dibutuhkan tanpa harus menulis ulang kode-kode program yang ada di dalamnya.

# Manfaat Fungsi

- Menjadikan struktur program menjadi **modular**, sehingga logika-logika dalam sebuah program lebih mudah dipahami.
- Program yang dirancang akan lebih efisien karena **tidak ada duplikasi** penulisan suatu kode-kode program saat dibutuhkan.
- Proses **pelacakan kesalahan** di dalam program menjadi **lebih mudah**, karena sekumpulan kode program dengan tujuan khusus dikemas di dalam suatu fungsi tersendiri.

# Mendefinisikan Fungsi

```
def nama_fungsi():  
    statement-1  
    statement-2  
    statement-3  
    statement-N
```

# Aturan Pembuatan Nama Fungsi

Aturan pembuatan nama fungsi sama dengan aturan pembuatan *identifier* (*Variable*, *class*, dan lainnya)

1. Nama fungsi dapat berupa kombinasi antara huruf (a-z atau A-Z), angka (0-9), dan/atau karakter *underscore* ( \_ ).
2. Diawali dengan huruf dan **tidak boleh diawali** dengan **angka**.
3. Tidak boleh menggunakan karakter khusus seperti, !, @, #, \$, %, dll
4. Tidak boleh menggunakan kata kunci (*Keywords*) yang sudah ditetapkan (*Reserved*) dalam pemrograman Python.

# Python Keywords

**Tabel 1.** Kata kunci (keywords) pada Python

and	del	for	is	raise	as
assert	elif	from	lambda	return	nonlocal
break	else	global	not	try	yield
class	except	if	or	while	with
continue	exec	import	pass	def	finally
in	print	input	<b>True</b>	<b>False</b>	<b>None</b>

Catatan:

Kata kunci (keywords) pada pemrograman python, bersifat case-sensitive dan ditulis dengan huruf kecil semua, kecuali **True**, **False**, dan **None**

# Contoh Penamaan Fungsi

```
def Data():  
    statement-1  
    statement-2  
    statement-3  
    statement-N
```

```
def _Penjumlahan():  
    statement-1  
    statement-2  
    statement-3  
    statement-N
```

```
def s4Mp11nG():  
    statement-1  
    statement-2  
    statement-3  
    statement-N
```

Valid

```
def 077_data():  
    statement-1  
    statement-2  
    statement-N
```

```
def akar_kuadrat():  
    statement-1  
    statement-2  
    statement-N
```

```
def input():  
    statement-1  
    statement-2  
    statement-N
```

Tidak Valid

# Memanggil Fungsi

Fungsi dalam program dapat dipanggil dengan menuliskan **nama fungsi** dan diikuti dengan **tanda kurung buka** dan **kurung tutup**.

Seperti contoh berikut:

```
nama_fungsi ()          # Memanggil fungsi
```



# Contoh Program dengan Fungsi:

```
# Cara Pemanggilan fungsi

def fungsi1():
    print("Aku fungsi1")

def fungsi2():
    print("Aku fungsi2")
    fungsi1()                # Memanggil fungsi1() dari fungsi2()

def main():
    print("\nPanggil semua fungsi!")
    fungsi2()                # Memanggil fungsi2() dari fungsi main()

# Program Utama
main()                      # Memanggil fungsi utama atau main()
```

```
# Output
Panggil semua fungsi!
Aku fungsi2
Aku fungsi1
```

# NILAI BALIK FUNGSI



# Nilai Balik Fungsi

- Suatu fungsi dapat mengembalikan suatu nilai pada saat dipanggil. Hal ini tergantung pada saat kita mendefinisikan fungsi tersebut.
- Secara default, nilai balik fungsi dalam pemrograman Python adalah **None**.
- Untuk memberikan nilai balik pada fungsi digunakan kata kunci **return** pada akhir fungsi.

# Mendefinisikan Fungsi Dengan Nilai Balik

Berikut adalah cara mendefinisikan fungsi dengan nilai balik:

```
def nama_fungsi():  
    statement-1  
    statement-2  
    statement-N  
    return nilai_balik
```

# Contoh Fungsi Tanpa Nilai Balik (Default)

```
def tampilkan():                                # Fungsi tanpa nilai balik (default)
    print("Hello Sahabat Robonesia!")

# Program Utama
tampilkan()                                    # Panggil fungsi tampilkan()
t = tampilkan()                                # Tampilkan nilai balik fungsi default
print(t)
```

```
# Output
Hello Sahabat Robonesia!
Hello Sahabat Robonesia!
None
```

# Contoh Fungsi dengan Nilai Balik

```
def penjumlahan():          # Fungsi dengan nilai balik tunggal
    hasil = 7+8+9
    return hasil

def kota():                 # Fungsi dengan nilai balik berupa list
    a = ['Batam','Jogja','Wonogiri','Surabaya']
    return a

def sayur():               # Fungsi dengan nilai balik berupa tuple
    b = ('Bayam',"Kangkung",'Wortel',"Sawi")
    return b

# Program Utama
c = penjumlahan()          # Menugaskan nilai balik fungsi penjumlahan() pada variabel c
print(c)
print(kota())              # Menampilkan nilai balik fungsi kota() secara langsung
print(sayur())             # Menampilkan nilai balik fungsi sayur() secara langsung
```

```
# Output
24
['Batam', 'Jogja', 'Wonogiri', 'Surabaya']
('Bayam', 'Kangkung', 'Wortel', 'Sawi')
```

# FUNGSI DENGAN ARGUMENT



# Argument Vs. Parameter

- Dalam bahasa pemrograman, ada 2 istilah yang sering disebutkan ketika ingin melewatkan suatu nilai dalam suatu fungsi, yaitu **argument** dan **parameter**.

**Parameter** = variabel yang digunakan untuk melewatkan suatu nilai di dalam suatu fungsi

**Argument** = merupakan nilai/data yang dimiliki oleh suatu parameter tersebut dan **dilewatkan** di dalam suatu fungsi pada saat fungsi tersebut dipanggil.



# Mendefinisikan Fungsi Dengan Argument

```
def nama_fungsi (par1, par2, ..., par-N) :  
    statement-1  
    statement-2  
    statement-3  
    statement-N
```

## Note:

**Argument** yang disimpan di dalam **parameter** (**par1**, **par2**, dst) untuk **dilewatkan** dalam suatu fungsi dapat berupa nilai **integer**, **float**, **string**, **list**, **tuple**, dll)

# 1. Fungsi dengan Argument Integer/Float

```
def penjumlahan(a,b): # Melewatkan argument integer/float
    jum = a+b
    print(a,"+",b," = ", jum)

# Program Utama
pilih = 'y'
while(pilih=="y" or pilih=="Y"):
    c = float(input("A: "))
    d = float(input("B: "))
    penjumlahan(c,d)
    pilih=input("Anda ingin menghitung lagi (y/n): ")
```

```
# Output
A: 7
B: 19
7.0 + 19.0 = 26.0
Anda ingin menghitung lagi (y/n): y
A: 2
B: 9
2.0 + 9.0 = 11.0
Anda ingin menghitung lagi (y/n): n
```

## 2. Fungsi dengan Argument **String**

```
def programUtama():  
    data = input("Tuliskan kalimat: ")  
    jumlah = jumlah_karakter(data)  
    print('\nJumlah karakter adalah: ', jumlah)  
  
def jumlah_karakter(string):  
    hitung = 0  
    indeks = 0  
    while (string[indeks] != '.' and indeks < (len(string)-1)):  
        hitung+=1  
        indeks+=1  
    return hitung+1  
  
programUtama()          # Panggil fungsi utama (main)
```

```
# Output  
Tuliskan kalimat: Belajar robotika di ROBONESIA  
  
Jumlah karakter adalah: 29
```

## 3. Fungsi Dengan Argument **List/Tuple**

```
def tampilkan_Tuple(tuple):
    for t in range(len(tuple)):
        print(t+1, '.', tuple[t])

def tampilkan_List(list):
    for u in list:
        print(u)

def program_utama():
    list = [1, 2, 3, 4, 5]
    tuple = ('jogja', 'solo', 'klaten', 'sukoharjo', 'wonogiri')

    print("Nama-nama kota:")
    tampilkan_Tuple(tuple)
    print('-----')
    print("Ini adalah data list:")
    tampilkan_List(list)

program_utama()          # Panggil program utama
```

```
# Output
Nama-nama kota:
1 . jogja
2 . solo
3 . klaten
4 . sukoharjo
5 . wonogiri
-----
Ini adalah data list:
1
2
3
4
5
```

## 4. Fungsi Dengan Argument Formal - **Posisional**

```
# Fungsi argument formal - argument posisional

def fungsi1(nama):
    print("Assalamu'alaikum", nama)

def fungsi2(nama1,nama2):
    print("Assalamu'alaikum",nama1,"dan",nama2)

fungsi1('Taufiq')           # Memanggil fungsi1
fungsi2('Boby','Budi')      # Memanggil fungsi2

fungsi2('Razak')            # Memanggil fungsi2, namun kurang jumlah argument-nya
```

```
# Output
Traceback (most recent call last):
  File "E:\PROGRAMMING\Pemrograman Python - 2 - PyCharm\pycharm_workspace\fungsi.py", line 121, in
<module>
    fungsi2('Razak')          # Memanggil fungsi2, namun kurang jumlah argument-nya
TypeError: fungsi2() missing 1 required positional argument: 'nama2'
Assalamu'alaikum Taufiq
Assalamu'alaikum Boby dan Budi
```

## 5. Fungsi Dengan Argument Formal - Default

```
# Fungsi argument formal - argument default

def tampilkan(t = 70): # Fungsi 'tampilkan' dengan argument default
    print(t)

tampilkan()           # Memanggil fungsi 'tampilkan' tanpa melewati argument (default)
tampilkan(277)        # Memanggil fungsi 'tampilkan' dengan melewati argument (277)
```

```
# Output
70
277
```

# FUNGSI DENGAN **VARIABLE-LENGTH** ARGUMENT



## 1. Fungsi Dengan Variable-length Argument **Non-Keyword (\*Tuple)**

- Saat suatu fungsi dipanggil, semua argument formal ditugaskan kepada variabel lokal yang sesuai dengan definisi yang telah ditetapkan.
- *Non-keyword argument* akan mengelompokkan **argument selain argument formal** yang dipanggil saat pemanggilan fungsi ke dalam suatu **tuple**.
- *Non-keyword argument* ditandai dengan tanda bintang satu (\*).

Format/Syntax:

```
def nama_fungsi(formal_argument, *argTuple):  
    statement1  
    statement2  
    dst
```



# Contoh Fungsi **Non-Keyword Argument (\*Tuple)**

```
def data_view(a,b,*c):  
    print("Formal argument A: ", a)  
    print("Formal argument B: ", b)  
    for t in c:  
        print("Non-keyword argument : ", t)  
  
def utama():  
    data_view(100,80,60,'Batam',355,'Jogja') # Memanggil 6 data  
  
utama() # Memanggil fungsi utama
```

```
#Output  
Formal argument A: 100  
Formal argument B: 80  
Non-keyword argument : 60  
Non-keyword argument : Batam  
Non-keyword argument : 355  
Non-keyword argument : Jogja
```

## 2. Fungsi Dengan Variable-length Argument Keyword (\*\*Dictionary)

- *Keyword argument* akan memetakan atau mengemas argument-argument ke dalam suatu **dictionary**.
- *Keyword argument* ditandai dengan tanda bintang dua (\*\*).

Format/Syntax:

```
def nama_fungsi(formal_argument, *argTuple, **argDictionary):  
    statement1  
    statement2  
    dst
```

```
def nama_fungsi(formal_argument, **argDictionary):  
    statement1  
    statement2  
    dst
```

# Contoh Fungsi Keyword Argument (\*\* Dictionary)

```
# Fungsi dengan argument berdasarkan panjang variabel - Keyword argument (Dictionary)
```

```
def data_dict(u,v,**w):  
    print("Formal argument U: ", u)  
    print("Formal argument V: ", v)  
    for t in w.keys():  
        print("Keyword argument ",t," : ",w[t])  
  
def utama():  
    data_dict(u='Durian',v='777',R=357,S=425,T=757)  
  
utama()      # Memanggil fungsi utama
```

```
#Output  
Formal argument U:  Durian  
Formal argument V:  777  
Keyword argument  R   :   357  
Keyword argument  S   :   425  
Keyword argument  T   :   757
```

# Catatan untuk *variable-length argument*

- Saat mendefinisikan suatu fungsi *variable-length argument*, baik berupa *non-keyword argument* (\*) ataupun *keyword argument* (\*\*), maka urutan penulisannya **harus sesudah argument formal**.
- Penulisan *keyword argument* (\*\*) **harus setelah** *non-keyword argument* (\*).

```
def robonesia(P, Q, *R):           # valid
def kota(P, Q, *R, **S):          # valid
def olahraga(*V, W, X):          # tidak valid
def sayur(V, **W, X):             # tidak valid
def buah(V, W, **X, *Y):          # tidak valid
```

**Terima Kasih**