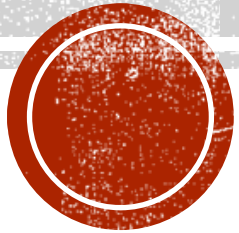
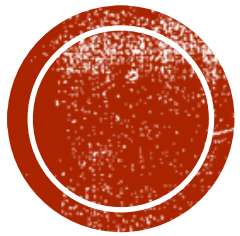
LEARNING PROGRESS REPORT

Dinda Adilfi Wirahmi





FUNGSI

APA FUNGSI?

- modul kode "mandiri" yang menyelesaikan tugas tertentu. Fungsi biasanya "mengambil" data, memprosesnya, dan "mengembalikan" hasil. Setelah fungsi ditulis, fungsi tersebut dapat digunakan berulang kali.

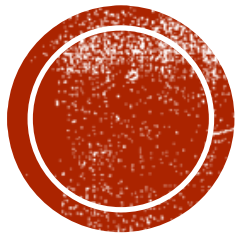
MENGAPA BUTUH FUNGSI?

Modular

- Semakin besar kode, fungsi akan membuat kode tersebut menjadi lebih rapi dan mudah untuk di*manage*.

DRY

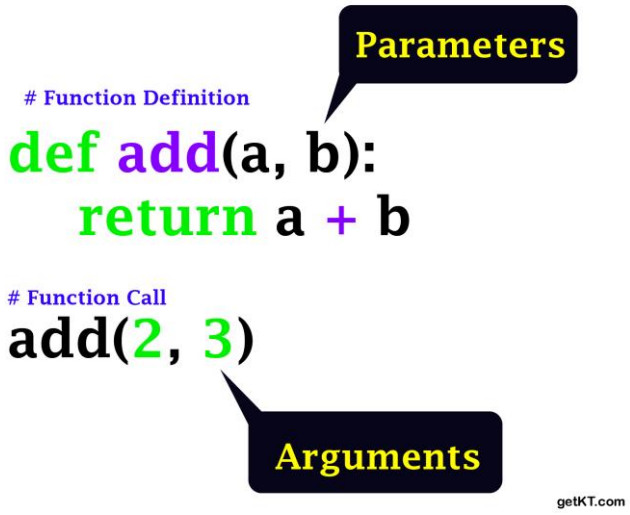
- Menghindari user dari melakukan pekerjaan berulang-ulang (*don't repeat yourself*)



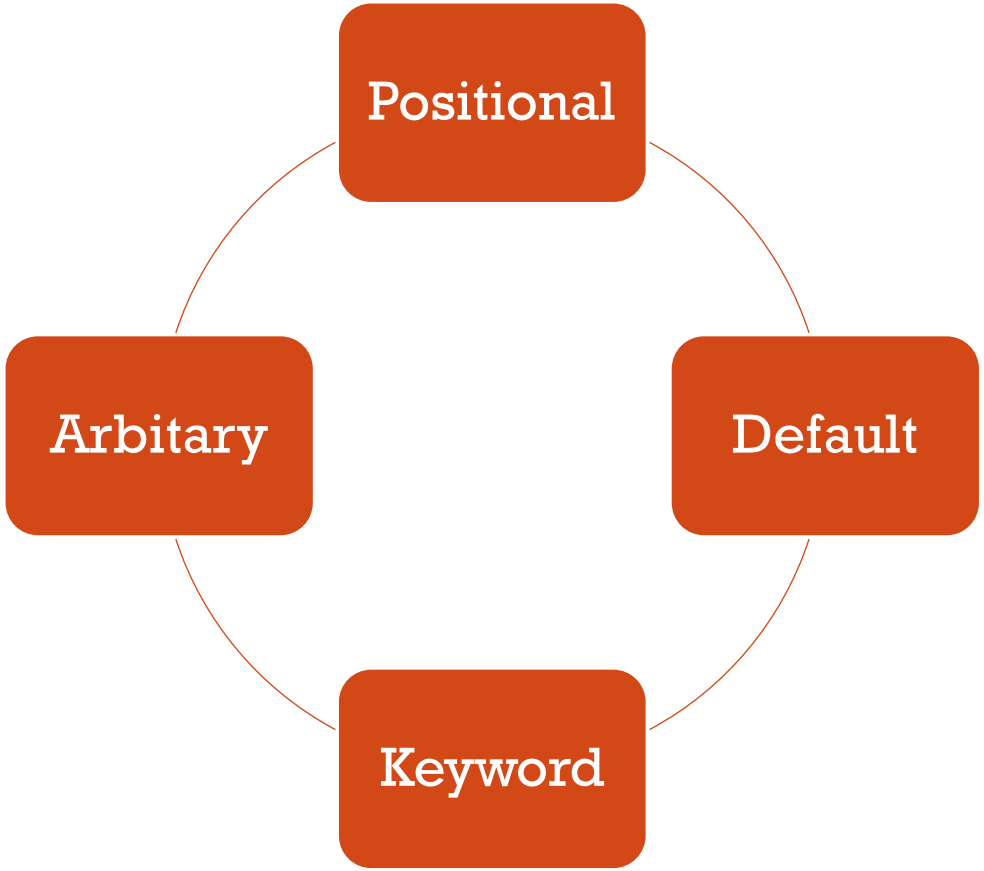
ARGUMENTS & PARAMETERS

PERBEDAAN ARGUMENTS DAN PARAMETERS

- **argumen** adalah sebutan untuk nilai inputan fungsi pada saat fungsi itu dipanggil.
- **Parameter** adalah sebutan untuk nilai input fungsi pada saat fungsi itu di definisikan.



TIPE ARGUMENTS



POSITIONAL ARGUMENTS

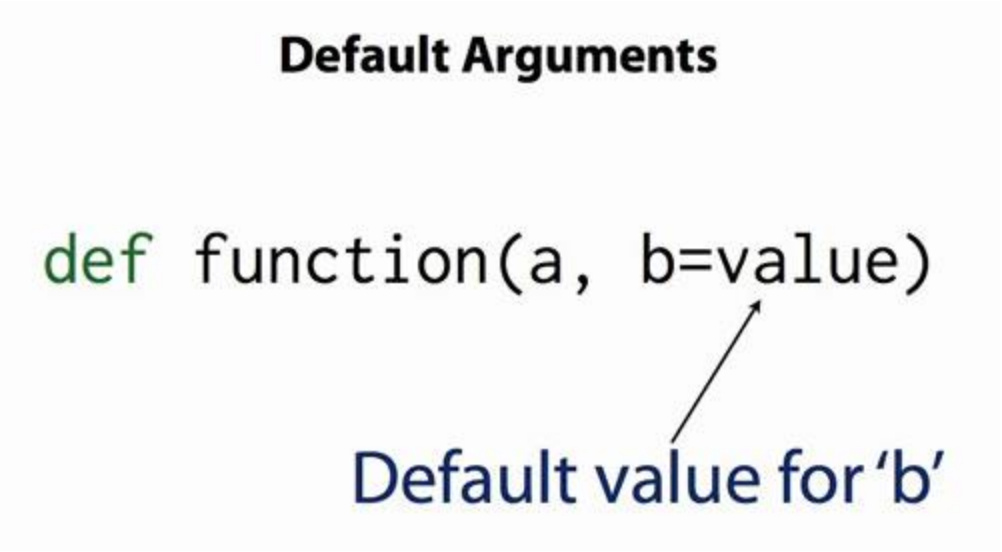
- adalah argumen yang dapat dipanggil berdasarkan posisinya dalam definisi fungsi

Positional Arguments →

```
def the_stars( x, y ):
   txt = "The stars are " + x + " and they are " + y
   return txt

sentence = the_stars( "red" , "orange" )
print sentence
```

DEFAULT ARGUMENTS

- adalah argumen yang mengambil nilai default jika tidak ada nilai eksplisit yang diteruskan ke argumen ini dari pemanggilan fungsi.



KEYWORD ARGUMENTS

- adalah istilah programming untuk menyebut cara pengiriman nilai dari argumen ke parameter function dengan menulis nama parameter, tidak sekedar nilainya saja.
- Dengan menggunakan parameter ini, user tidak harus bergantung kepada urutan parameter pada saat menjalankan sebuah fungsi. Urutan argumen bisa ditulis acak selama nama argumen sama dengan nama parameter.

```
1  def foo(var1, var2, var3):
2    ## isi fungsi disini...
3    ## isi fungsi disini...
4
5  foo(var3 = 100, var1 = 200, var2 = 300)
```

ARBITARY ARGUMENTS (*ARGS)

- **Arbitrary arguments** adalah istilah untuk menyebut jumlah argumen yang tidak bisa ditentukan atau berubah-ubah.

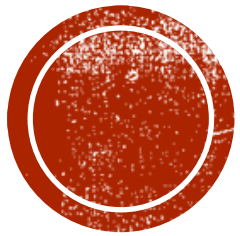
```
1  def sapa_teman(nama1, nama2, nama3):
2    print("Halo",nama1)
3    print("Halo",nama2)
4    print("Halo",nama3)
5
6  sapa_teman("Alex","Nisa","Sari")
```

Hasil kode program:

```
Halo Alex
Halo Nisa
Halo Sari
```

ARBITARY KEYWORD ARGUMENTS

- **Arbitrary keyword arguments** adalah istilah untuk menyebut jumlah named argumen fungsi yang tidak bisa ditentukan atau berubah-ubah.
- Jika dalam arbitrary arguments (***args**) argumen fungsi ditulis langsung dengan nilai saja, maka dalam arbitrary keyword arguments (****kwargs**), argumen fungsi tersebut ditulis dalam bentuk pasangan nama dan value.



RETURN

DENGAN STATEMENT RETURN

- Dalam banyak situasi, hasil sebuah function perlu disimpan ke dalam variabel terlebih dahulu, untuk kemudian di proses lebih lanjut.
- Untuk keperluan inilah perlu menambah perintah return ke dalam function. Tujuannya, agar sebuah function bisa mengembalikan nilai.

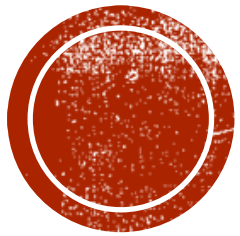
```
1  def hitung_luas_segitiga(alas, tinggi):
2    luas = (alas * tinggi) / 2
3    return luas
4
5  var1 = hitung_luas_segitiga(5, 7)
6  print("Luas segitiga adalah:",var1)
```

Hasil kode program:

```
Luas segitiga adalah: 17.5
```

FUNGSI SEBAGAI *FIRST CLASS OBJECT*

- fungsi dapat digunakan sebagai parameter pada fungsi lain, sama seperti *first-class object* lain di Python, seperti *string*, *int*, *float*, *list*, dan sebagainya.



LOGIC FLOW

- Operator ini dibutuhkan ketika proses logic

Comparison Operators

==	value sama, dan data type sama
>	lebih dari
<	kurang dari
>=	lebih dari sama dengan
<=	kurang dari sama dengan

Logical Operators

and	(keduanya benar, maka TRUE)
or	(salah satu benar, maka TRUE)
not	(membalik logika TRUE/FALSE)

CONTOH COMPARISON DAN LOGICAL OPERATOR

1. Comparison

```
x = 5   dan   y = "5"
print(x==y)
```

Outputnya : False, karena int tidak sama dengan string

```
print(x > int(y))
```

Outputnya : False, karena int 5 tidak lebih besar dari int 5

```
print(x >= int(y))
```

Outputnya : True, karena int 5 tidak lebih besar namun sama dengan int 5 sehingga salah satu kondisi terpenuhi

```
print(x < int(y))
```

Outputnya : False, karena int 5 tidak lebih kecil daripada int 5

```
print(x <= int(y))
```

Outputnya : True, karena int 5 tidak lebih kecil namun sama dengan int 5 sehingga salah satu kondisi terpenuhi

CONTOH COMPARISON DAN LOGICAL OPERATOR

2. Logical

```
x = 5
y = '5'
z = 6
print(x==int(y) and int(y)<z)
      (5 == 5  and 5 < 6)
        true          false          : true and false menghasilkan false

print(not(x==int(y) or int(y)<z))
      (not (5 == 5 or   5 < 6))
      (not (true     or   false ))
      (not   (true ))                 : not true menghasilkan false
```

IF, ELSE IF, ELSE

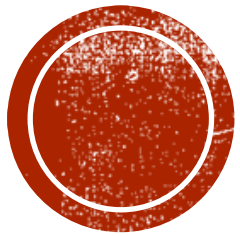
```
nilai = 40;

if (nilai > 80) :
    print('Excellent!');
elif (nilai >= 60 and nilai <= 80) :
    print('Good job!');
else :
    print("Don't give up!");
```

Konsepnya adalah up to down

Jika nilai yang sudah diinisialisasi bernilai lebih dari 80, maka akan di menghasilkan **Excellent**!

Jika bernilai pada rentang 60 – 80 maka akan menghasilkan **Good job!**

Namun, jika tidak memenuhi kedua kondisi tersebut, maka akan menghasilkan **Don’t give up!**



LOOPING

LOOPS

Secara konsep loops, terdapat syntax `range(1,5)`

Jika fungsi range memiliki dua input dimana input pertama lebih besar dari input kedua, maka outputnya adalah sebuah urutan yang dimulai pada masukan pertama. Kemudian urutan iterasi sampai angka sebelum angka kedua.

Misal :

range(1,5) : 1,2,3,4

LOOPS - FOR

Misal terdapat list :

```
squares = [“red”, “yellow”, “green”, “blue”]
```

Untuk mengambil semua atau beberapa element di dalam list, perlu dilakukan looping. Salah satunya menggunakan loops for

```
for item in squares :
      print(item)
```

Lopping tersebut akan memberikan output yaitu semua element di dalam list squares

LOOPS - WHILE

```
angka = 1
while(angka <= 10):
        print(angka)
        angka += 1
```

Variabel angka dengan value 1 akan masuk ke kondisi while terlebih dahulu.
Jika value 1 tersebut kurang atau sama dengan dari 10, maka akan ditampilkan output 1.

Setelah itu output tersebut akan dilakukan plus 1 untuk melanjutkan iterasi ke value selanjutnya sampai variabel angka memiliki value yang lebih besar dari 10 yang berarti tidak memenuhi kondisi.

Output dari loops while diatas adalah angka 1 - 10

LOOPS - NESTED

```
adj = ["red", "big", "tasty"]
fruits = ["apple", "banana", "cherry"]

for x in adj:
  for y in fruits:
    print(x, y)
```

Terdapat list adj dan fruits

Dilakukan berdasarkan index awal ke index akhir

x = “red”
  y = “apple”

Menghasilkan output : red apple

Dan seterusnya

thank you!