

Algoritme Pemrograman

#9 - Fungsi

TIM TPB Alpro

Apa yang akan kalian pelajari?

Membuat dan menggunakan fungsi dengan tujuan agar:

- ✓ Memudahkan pengorganisasian program
- ✓ Menggunakan Kembali rangkaian program yang telah dibuat (*Coding* lebih cepat)
- ✓ Lebih mudah memelihara kode program

Fungsi

- Fungsi merupakan sekumpulan pernyataan / baris kode yang bertujuan untuk melakukan tugas tertentu
- Kita sudah sering menggunakan fungsi pada Python seperti :

`print()`

`range()`

`len()`

`max()`

...

$f(x)$

Fungsi, lanjt.

- Kebanyakan kode program utuh yang ditulis sangat besar
- Pemrogram mengorganisasi kodenya dengan membuat program tersebut menjadi beberapa sub-program yang lebih kecil sehingga mudah untuk dikelola.
- Dengan cara seperti ini, paradigma “Divide and Conquer” diterapkan untuk menyelesaikan masalah pemrograman

$f(x)$

Definisi Fungsi

Nama Fungsi

Kurung Buka dan Tutup

```
def fungsiSaya():  
    print("cetak dari dalam fungsi")  
    print("cetak diluar fungsi")  
fungsiSaya()
```

Pemanggilan
fungsi

Fungsi `dir()`



`dir()`

Fungsi bawaan `dir()` akan mencetak daftar atribut ataupun fungsi yang dapat digunakan oleh pemrogram

Jika dimasukkan argumen berupa objek maka akan mengembalikan atribut yang tersedia pada objek tersebut

Tanpa argument akan memberikan nama dan fungsi-fungsi yang dapat dipanggil oleh pemrogram

Alur Eksekusi

Kode Program



```
print(dir())
```


```
def fungsiSaya():  
    print("cetak dari dalam fungsi")
```

```
print(dir())
```

```
print("cetak diluar fungsi")  
fungsiSaya()
```

Alur Eksekusi

Keluaran



```
['__annotations__', '__builtins__', '__cached__', '__doc__', '__file__',  
 '__loader__', '__name__', '__package__', '__spec__']
```

```
['__annotations__', '__builtins__', '__cached__', '__doc__', '__file__',  
 '__loader__', '__name__', '__package__', '__spec__', 'fungsiSaya']
```

cetak diluar fungsi

cetak dari dalam fungsi

Variabel Lokal dan Global

- **Variabel lokal** di Python merupakan variabel yang didefinisikan pada blok tertentu (seperti fungsi) yang hanya dapat dipanggil pada blok tersebut.
- **Variabel global** di Python merupakan variabel yang didefinisikan di luar blok tertentu dan dapat dipanggil dibagian manapun.

$f(x)$

Variabel Lokal

Variabel lokal tidak bisa diakses diluar fungsi

```
def cetakAngka():  
    angka = int(input("Masukkan angka 0 - 9? "))  
    print(angka)
```

```
cetakAngka()  
print(angka)
```

NameError: name 'angka' is not defined

Variabel Lokal

Variabel lokal tidak akan mengganti nilai variabel lokal dengan nama yang sama pada cakupan (salah satunya fungsi) yang berbeda

Kode Program

```
def balikpapan():  
    kota = "balikpapan"  
    print(f"Kota: {kota}")
```

```
def samarinda():  
    kota = "samarinda"  
    print(f"Kota: {kota}")
```

```
balikpapan()  
samarinda()
```

Keluaran

```
Kota: balikpapan  
Kota: samarinda
```

Variabel Global

Variabel global merupakan suatu variabel yang dapat diakses oleh semua cakupan. Seperti layaknya anda membuat variabel diluar fungsi.

Kode Program

```
a = 5
def kuadratkan():
    print(a**2)

kuadratkan() #5
```

Dapat diakses didalam fungsi

Variabel Global, lanjut.

Untuk membuat variabel lokal menjadi global dapat menggunakan keyword `global`

Kode Program

```
a = 5
b = 10

def ubahNilai():
    global a
    a = 90
    b = 90

ubahNilai()
print(a) # 90
print(b) # 10
```

Variabel lokal b
tidak merubah
variabel global b

Membuat variabel
a menjadi global

Catatan Variabel Global

- Variabel global dapat membuat penelusuran kesalahan lebih rumit
- Fungsi yang menggunakan variabel global akan bergantung pada variabel tersebut sehingga membuat fungsi kurang portable (mudah dipindah posisikan/disesuaikan)
- Dalam beberapa kasus variabel global sangat berguna

$f(x)$

Fungsi dengan Argumen

- Kadang-kadang fungsi akan lebih bermanfaat jika kita dapat memasukkan atau mengirimkan satu atau lebih data untuk bisa diproses
- Mekanisme ini sudah pernah kita gunakan dalam menggunakan fungsi yang tersedia di Python :
 - `len('Widodo')` #6
 - `max([1,2,3])` #3

Fungsi dengan Argumen

Parameter -> variabel

Kode Program

```
def pangkat2(bil):  
    """Fungsi untuk memangkatkan dua suatu bilangan"""  
    print(bil**2)  
help(pangkat2)  
pangkat2(5)
```

Docs string

Bantuan informasi
suatu fungsi

Argumen -> data

Keluaran

Help on function pangkat2 in module __main__:

```
pangkat2(bil)
```

```
    Fungsi untuk memangkatkan dua suatu bilangan
```


Fungsi dengan Lebih dari Satu Argumen

Kode Program

```
def jumlahBilangan(bil1, bil2):  
    """Fungsi untuk menjumlahkan dua buah bilangan"""  
    print(bil1 + bil2)  
help(jumlahBilangan)  
jumlahBilangan(10, 5) # 15
```

Keluaran

```
Help on function jumlahBilangan in module  
__main__:
```

```
jumlahBilangan(bil1, bil2)  
    Fungsi untuk menjumlahkan dua buah  
bilangan
```



Coding Challenge ~ Live Coding

Buatlah program untuk mencetak nilai faktorial dari suatu bilangan. Misal faktorial(4) akan menghasilkan:

$$4 \times 3 \times 2 \times 1 = 24$$

Pass by Value and Pass by Reference pada Argumen

- *Pass by Value* merupakan variabel yang dilewatkan berupa nilainya (nilai dari variabel disalin dan disematkan pada variabel baru)
- *Pass by Reference* merupakan variabel yang dilewatkan berupa referensi / alamat memorinya, sehingga apapun yang dioperasikan pada variabel tersebut akan merujuk pada variabel aslinya.

```
def contohByValue(par):  
    par = 7
```

```
angka = 9  
contohByValue(angka)  
print(angka)
```

```
def contohByReference(par):  
    par[0] = "huruf"
```

```
list = [9,10,11]  
contohByReference(list)  
print(list)
```

```
def contohLain(par):  
    par = [0,1,2]
```

```
contohLain(list)  
print(list)
```

Fungsi dengan Nilai Kembali

- **Fungsi dengan nilai Kembali** merupakan suatu fungsi yang akan mengembalikan berupa nilai / *value* sebagian program yang memanggil fungsi tersebut.
- Fungsi ini sama dengan fungsi yang telah kita buat sebelumnya, namun memiliki kemampuan untuk mengirim kembali informasi dari pemanggil fungsi.
- Contoh fungsi dengan nilai kembali yang sering digunakan:

```
nama = input("Masukkan nama anda: ")
```

```
panjangKarakter = len("Institut Teknologi Kalimantan")
```

Pendefinisian Fungsi dengan Nilai Kembali

- Menggunakan sintaksis yang sama dalam mendefinisikan fungsi
- Terdapat *keyword* “return” pada akhir pernyataan dalam blok fungsi yang akan mengembalikan nilai yang dimaksud
- Return akan membuat fungsi berhenti seketika, layaknya break pada perulangan. Statement setelahnya tidak akan dijalankan

```
def fungsiSaya(arg1, arg2):  
    #statement  
    #statement  
    #...  
    #return
```

```
def rataan3Bil(bil1, bil2, bil3):  
    rata2 = (bil1+bil2+bil3)/3  
    return rata2  
  
x = rataan3Bil(5,6,7)  
print(x) # 6.0
```

Coding Challenge ~ Live Coding

Diskon Barang.

Mintalah masukan kepada pengguna berupa harga barang (gunakan fungsi)

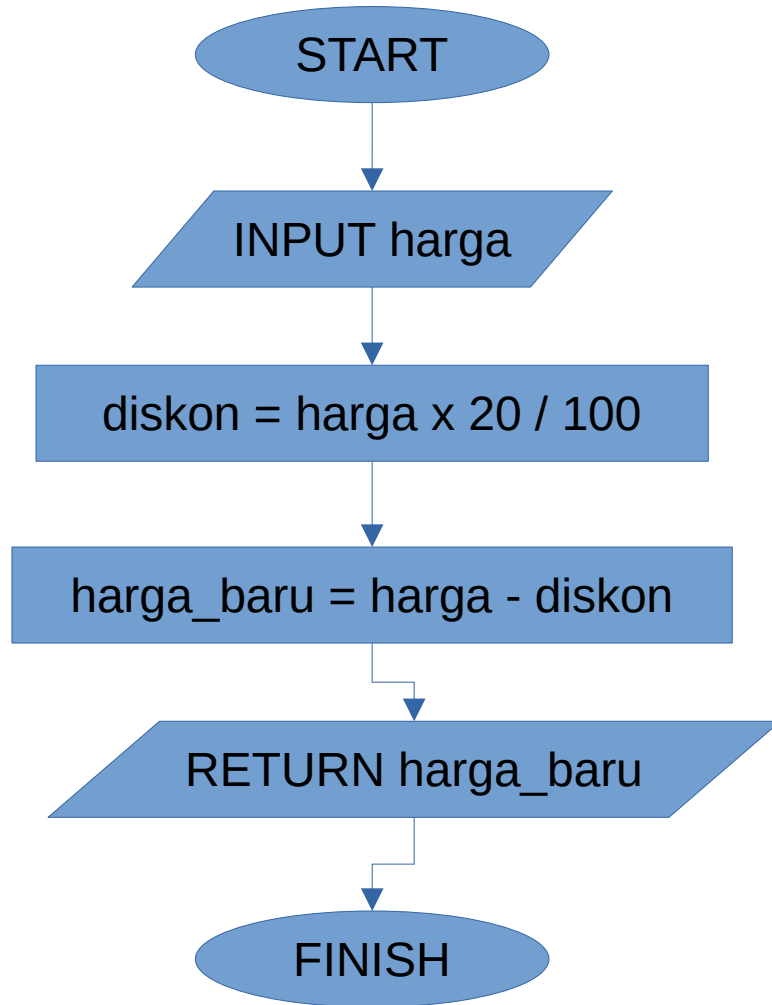
Jangan terima harga dibawah 50.000, minta kepada pengguna kembali harga diatas harga tersebut.

Diskon 20% harga tersebut (gunakan fungsi)

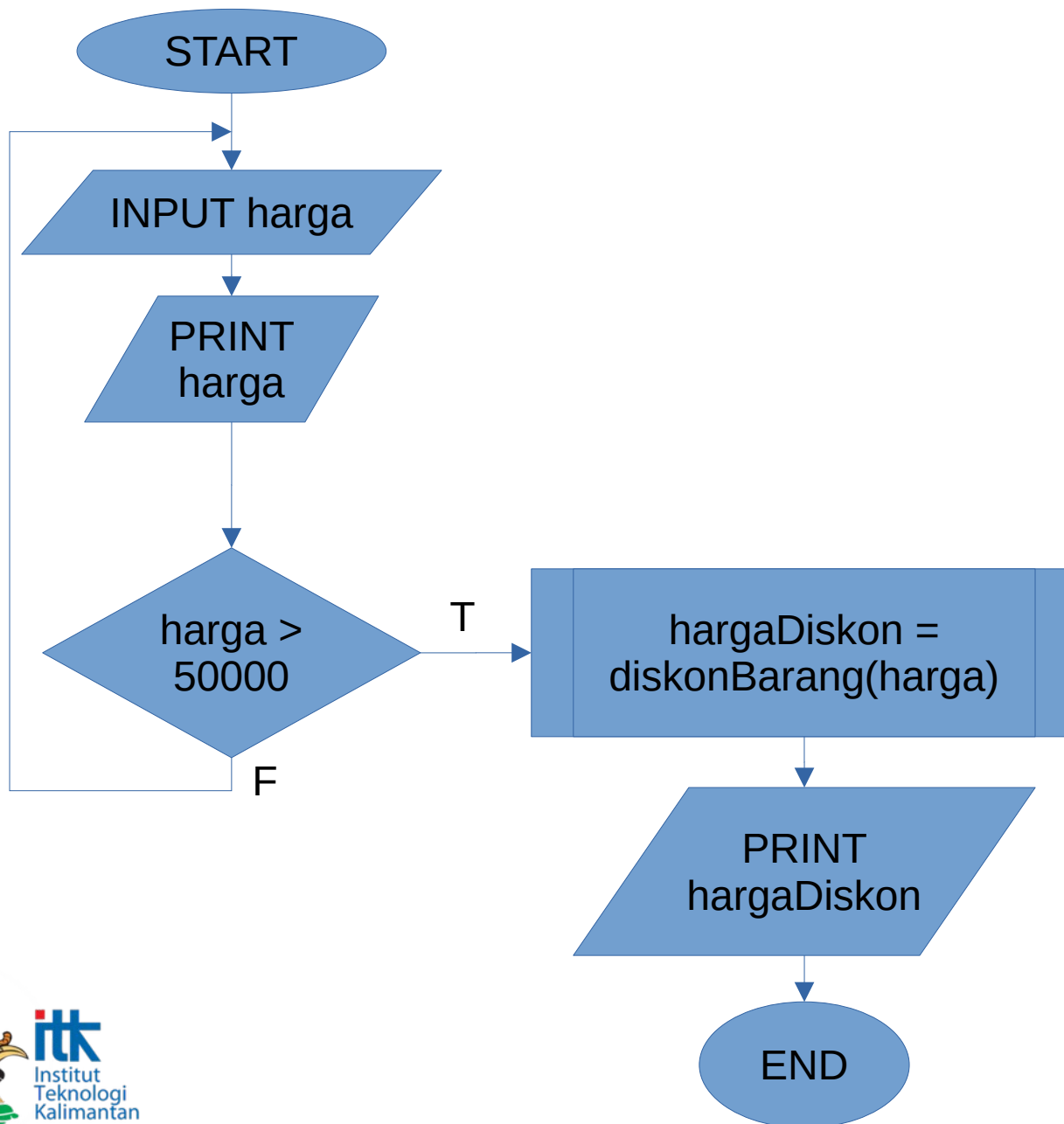
Cetak harga awal dan harga setelah diskon



flowchart1. diskonBarang()



flowchart2:




```
def diskonBarang(harga_lama):
    diskon = harga_lama * 20 / 100
    harga_baru = harga_lama - diskon

    return harga_baru

while True:
    harga = int(input('masukkan harga awal: '))

    if harga > 50000:
        print(f'harga awal = {harga}')
        hargaDiskon = diskonBarang(harga)
        print(f'harga setelah diskon = {hargaDiskon}')
        break
    else:
        continue
```

Fungsi dengan Banyak Nilai Kembali

➤ Fungsi dapat mengembalikan lebih dari satu nilai :

```
def tambahKali(bil1, bil2):  
    hasilTambah = bil1 + bil2  
    hasilKali = bil1 * bil2  
    return hasilTambah, hasilKali
```

```
a, b = tambahKali(2, 5)
```

```
print(a) # 7  
print(b) # 10
```

Coding Challenge ~ Live Coding

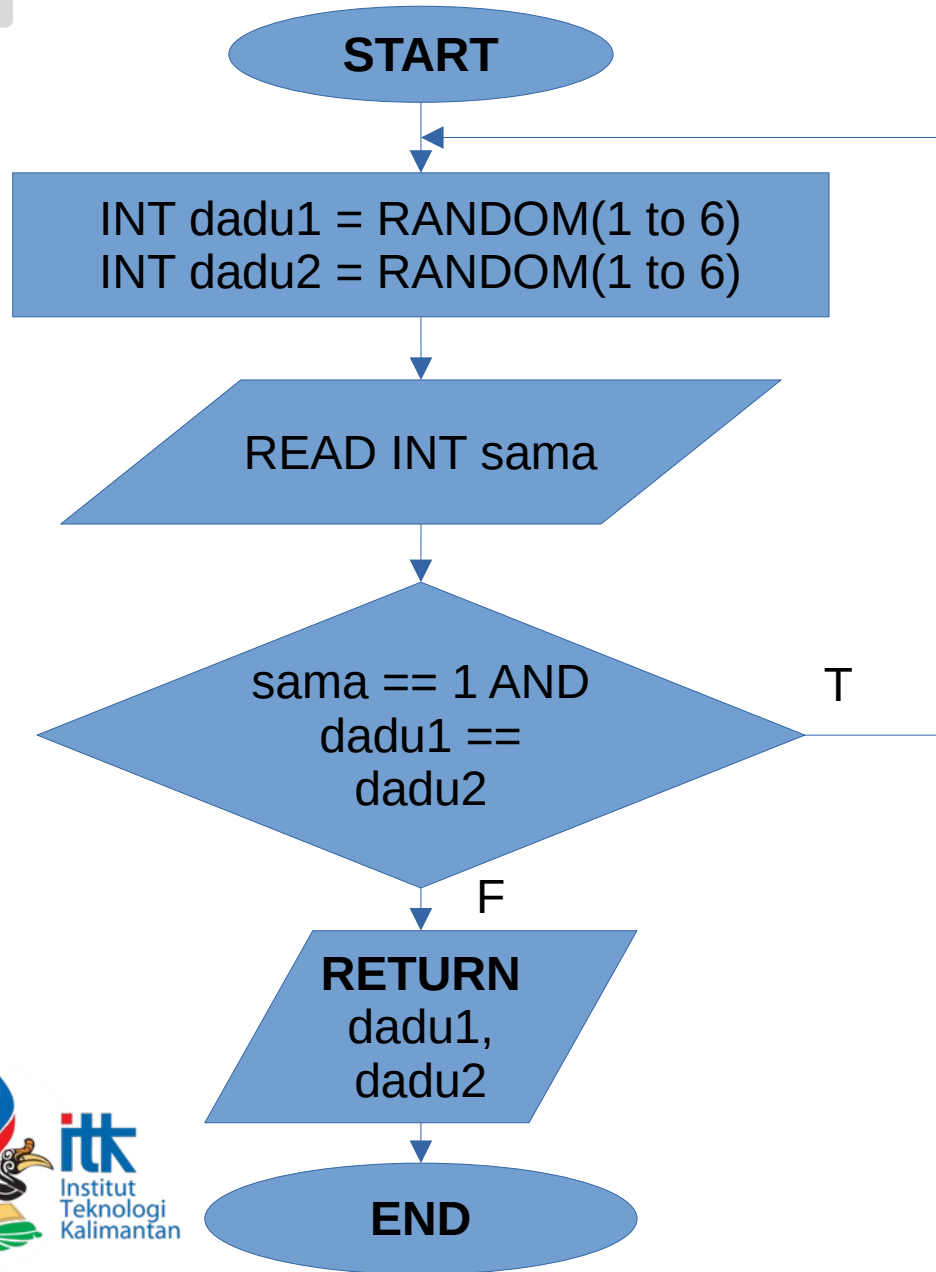
Buatlah program untuk mensimulasi dua buah dadu.

Hasilnya harus mengembalikan dua buah nilai dadu.

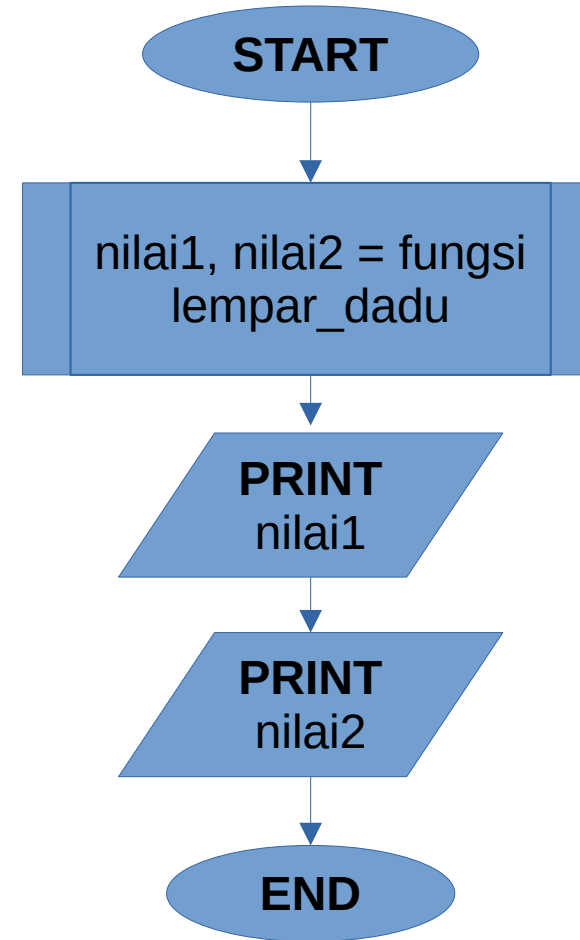
Terdapat argument tambahan apakah nilai kedua dadu
dibolehkan sama atau tidak



algoritma 1 : fungsi lempar_dadu



algoritma 2 : jalankan fungsi lempar_dadu





```
from random import randint
```

```
def lempar_dadu(sama):  
    while True:  
        dadu1 = randint(1,6)  
        dadu2 = randint(1, 6)  
  
        if sama == 1 and dadu1 == dadu2:  
            continue  
        else:  
            break  
  
    return dadu1, dadu2
```

```
nilai1, nilai2 = lempar_dadu(0)  
print(f'Nilai dari dadu 1 = {nilai1}')  
print(f'Nilai dari dadu 2 = {nilai2}')
```

Argumen Katakunci (*Keyword Arguments*)

- Kita dapat memberikan kata kunci pada setiap argument yang ada pada fungsi

```
def rataan3bil(bil1, bil2, bil3):  
    print(f"bil 1: {bil1} | bil 2: {bil2} | bil 3: {bil3}")  
    rata2 = (bil1+bil2+bil3)/3  
    return rata2
```

```
x = rataan3bil(bil3=5, bil2=4, bil1=3)  
print(x)
```

Argumen Katakunci (*Keyword Arguments*)

- Selain kata kunci, nilai awal (*default*) juga dapat ditentukan walaupun pada saat pemanggilan fungsi, argumen tidak diberikan.

```
def rataan3bil(bil1 = 1, bil2 = 1, bil3 = 1):  
    print(f"bil 1: {bil1} | bil 2: {bil2} | bil 3: {bil3}")  
    rata2 = (bil1+bil2+bil3)/3  
    return rata2
```

```
x = rataan3bil()  
y = rataan3bil(bil2 = 5)  
print(x)  
print(y)
```

Penataan Posisi Argumen

- Argumen biasa dikenali berdasarkan posisinya atau dapat diakses melalui nama paremeternya, argumen ini sifatnya adalah wajib untuk diisi, sedangkan parameter dengan nilai awal (*default*) sifatnya adalah pilihan (*opsional*)
- Argumen dituliskan terlebih dahulu baru diikuti dengan argument pilhan (*opsional*)

```
def jumlahBil(bil1, bil2, bil3 = 0):  
    jumlah = bil1+bil2+bil3  
    return jumlah
```

```
x = jumlahBil(2, 5)  
y = jumlahBil(2, 5, 10)  
z = jumlahBil(2, 5, bil3 = 5)  
print(x)  
print(y)
```

```
z = jumlahBil(bil3 = 4, 5, 7)  
SyntaxError: positional argument  
follows keyword argument
```


*args dan **kwargs

- *args dan **kwargs merupakan sintaksis khusus untuk menunjukan argument yang tidak terbatas
- *args menunjukan argumen standar
- **kwargs menunjukan argumen kata kunci (*keyword arguments*)
- Sintaksis * menunjukan pemrogram dapat mengambil lebih banyak argumen dari argument yang telah didefinisikan sebelumnya

```
def jumlahBil(*bil):  
    jumlah = 0  
    for b in bil:  
        jumlah+=b  
    return jumlah
```

```
x = jumlahBil(7,6,5,3,2,9,10)  
print(x) #42
```

****kwargs**

Kode Program

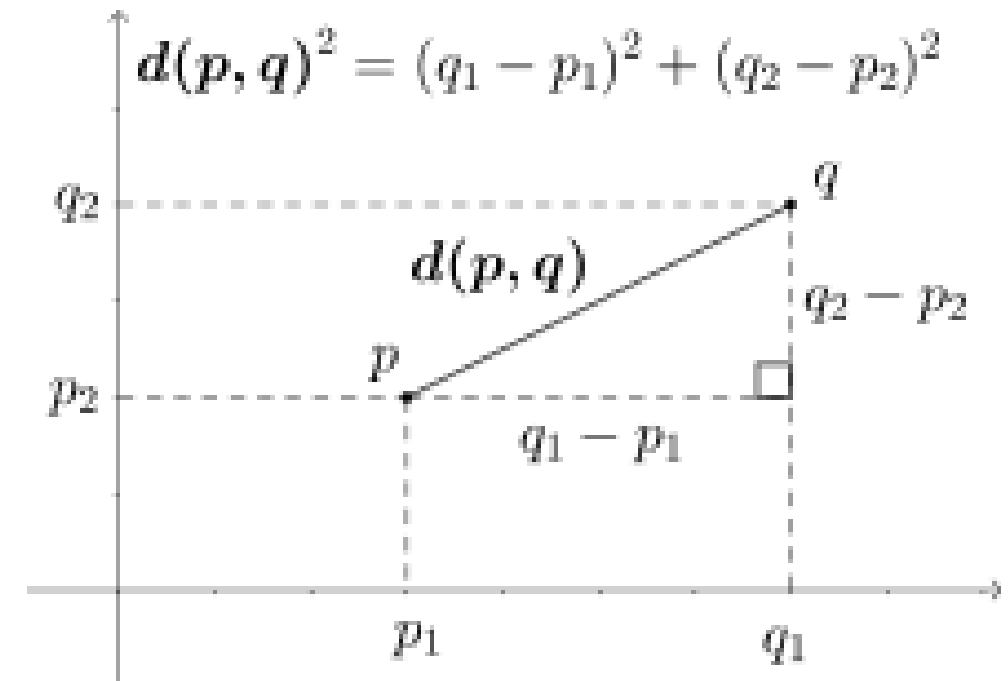
```
def cetakData(**data):  
    print(data)  
    for key, value in data.items():  
        print(f"{key}: {value}")  
  
cetakData(nama = "Anggi", umur = 18, prodi = "Matematika")
```

Keluaran

```
{'nama': 'Anggi', 'umur': 18, 'prodi': 'Matematika'}  
nama: Anggi  
umur: 18  
prodi: Matematika
```

Coding Challenge ~ Live Coding

Buatlah program untuk bisa menghitung *Euclidian Distance* multi dimensi. Manfaatkan *args dalam membuat fungsi.



$$d(\mathbf{p}, \mathbf{q}) = d(\mathbf{q}, \mathbf{p}) = \sqrt{(q_1 - p_1)^2 + (q_2 - p_2)^2 + \cdots + (q_n - p_n)^2}$$
$$= \sqrt{\sum_{i=1}^n (q_i - p_i)^2}.$$



```
import math
```

```
def hitung_euclidian_distance(*titik):  
    jumlah = 0
```

```
    for i in range(int(len(titik)/2)):  
        jumlah += (titik[i] - titik[int(len(titik)/2)]) ** 2
```

```
    jarak = math.sqrt(jumlah)
```

```
    return jarak
```

```
jarak = hitung_euclidian_distance(0,0,0,0,1,1,1,1)
```

```
print(f'Euclidian distance-nya = {jarak}')
```

Ekspresi Lambda

- Lambda digunakan untuk membuat fungsi anonim
- Hanya satu ekspresi dalam tubuh lambda, nilainya akan selalu dikembalikan
- Contoh pembuatan lambda:

```
f = lambda x, y : x**2 + y**2  
print(f(5,6)) # 61
```

Fungsi Rekursif

- Rekursif merupakan fungsi yang memanggil dirinya sendiri
- Tujuan dari rekursif adalah untuk menyederhanakan pemrograman dan sebagai alternatif iterasi

```
def faktorial(n):  
    if n < 2 :  
        return 1  
    else:  
        return n * faktorial(n-1)  
  
print(faktorial(5)) #120
```

Kondisi berhenti,
tidak memanggil
dirinya kembali

Memanggil dirinya
sendiri

Coding Challenge ~ Live Coding

Buatlah program untuk bisa
mencetak pola seperti gambar
disamping menggunakan fungsi!



Coding Challenge ~ Live Coding

Modifikasilah agar dapat menerima masukan pengguna berupa tinggi pola tersebut!




```
def cetak_bintang(tinggi):
    for i in range(tinggi):
        if i == 0 or i == tinggi-1:
            print('* ' * 10)
        else:
            for j in range(10):
                if j == 0:
                    print('*', end='')
                elif j == 9:
                    print(' *')
                else:
                    print(' ', end='')
```

```
cetak_bintang(int(input('masukkan tinggi: ')))
```

Ada Pertanyaan ???



To Be Continue

Thanks