

LEARNING PROGRESS REVIEW

WEEK 4

Diaz Jubairy - Hermulia Hadie
Desi Sulistyowati - Farahul Jannah

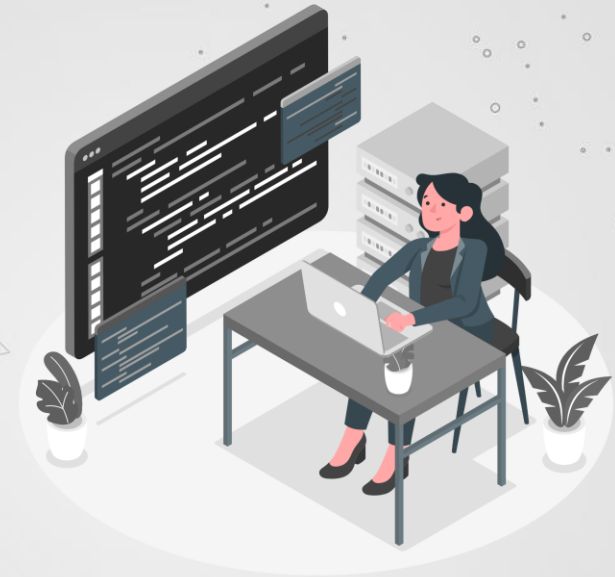


TABLE OF CONTENTS



BASIC PROGRAMMING I

Conditions, control flow,
if, switch case

01

BASIC PROGRAMMING II

While loop, for loop,
nested loop control loop,

02

BASIC PROGRAMMING III

Data type, array, list,
tuple, set, dictionary

03



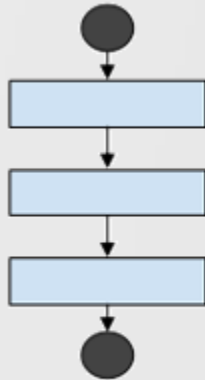


01

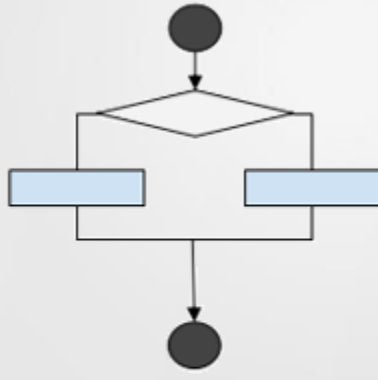
BASIC PROGRAMMING I CONDITIONS

Condition and Control Flow

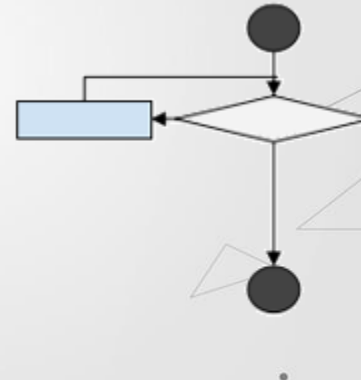
Control flow dapat didefinisikan sebagai cara untuk memberi tahu program instruksi apa yang harus dijalankan sesuai urutan yang diinginkan. *Control flow* pada sebuah program Python diatur menggunakan *conditional statements*, *loops*, dan *function calls*.



Sequential



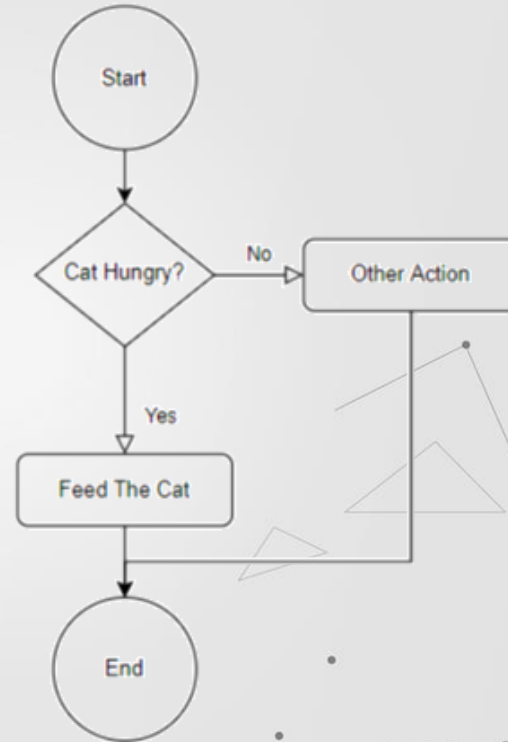
Conditional



Looping

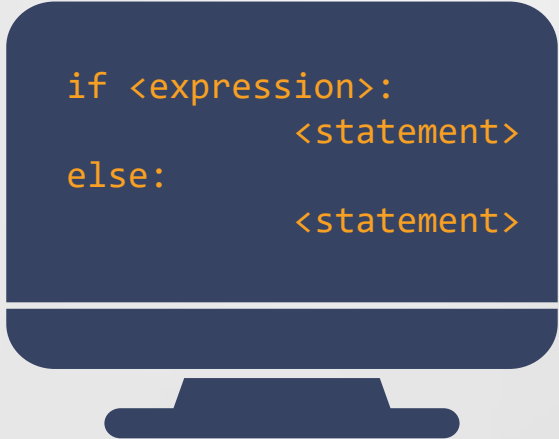
Condition and Control Flow

Kode yang dijalankan akan berhenti sampai suatu atau beberapa kondisi yang telah ditentukan tercapai. Sebagai contoh, apabila kucing lapar (kondisi), maka saya akan memberinya makan (aksi).



Condition and Control Flow

Untuk melakukan proses pengambilan keputusan pada Python, dapat digunakan if statement. If statement memuat kode-kode yang hanya akan dijalankan apabila sesuai dengan kondisi yang ditentukan (condition is true). Apabila statement tidak sesuai dengan kondisi (condition is false), maka dapat dijalankan opsi statement lainnya yang diinginkan.



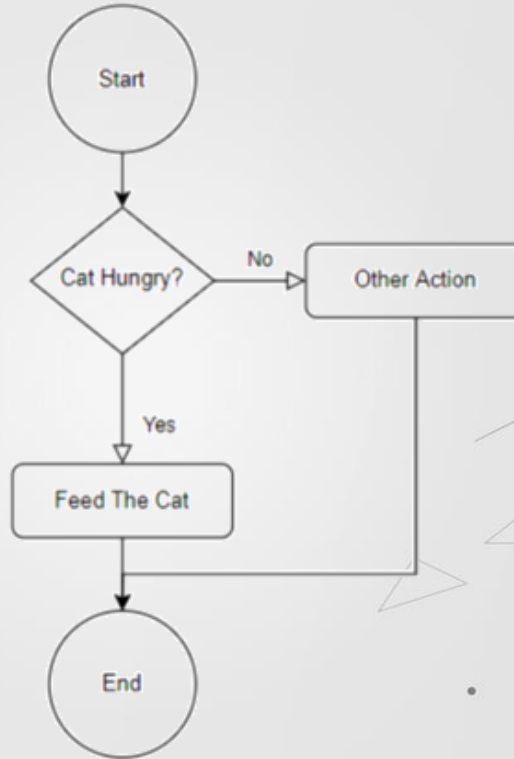
```
if <expression>:  
    <statement>  
else:  
    <statement>
```

Condition and Control Flow

if cat is hungry then
 feed the cat
else
 do other things



```
if cat == 'hungry':  
    feed_the_cat()  
else:  
    other_action()
```




Condition and Control Flow

Python dapat menggunakan kondisi yang berbasis logika matematika, dengan syntax sebagai berikut:

- Equals: $a == b$
- Not Equals: $a != b$
- Less than: $a < b$
- Less than or equal to: $a \leq b$
- Greater than: $a > b$
- Greater than or equal to: $a \geq b$

Logika umum lainnya dan syntax yang digunakan yaitu sebagai berikut:

- ◻ True if both the operands are true: and
 - ◻ True if either of the operands is true: or
 - ◻ True if operand is false (complements the operand): not
- 

"If" Statement

Fungsi IF merupakan percabangan yang digunakan untuk menentukan tindakan apa yang dilakukan sesuai dengan kondisi tertentu. Contohnya:

Single Condition (If)

```
a = 3
if a == 3:
    print("a is three")
```

Output: **a is three**

Two Condition (If → else)

```
a = 5
b = 5
if a != b:
    print("a and b is not the same")
else:
    print("a and b is the same")
```

Output: **a and b is the same**

Multiple Condition (If → elif → else)

```
a = 3
b = 5
if a > b:
    print("a is bigger than b")
elif a < b:
    print("a is smaller than b")
else:
    print("a and b is the same")
```

Output: **a is smaller than b**



“If” Statement [NESTED]

Merupakan statement IF yang dijalankan pada statement IF yang lain. Contohnya:

```
x = 50
if x > 20:
    print("Above twenty, ")
    if x > 40:
        print("and also above 40")
    else:
        print("but not above 40")
else:
    print("Below twenty")
Output: Above twenty, and also above 40
```

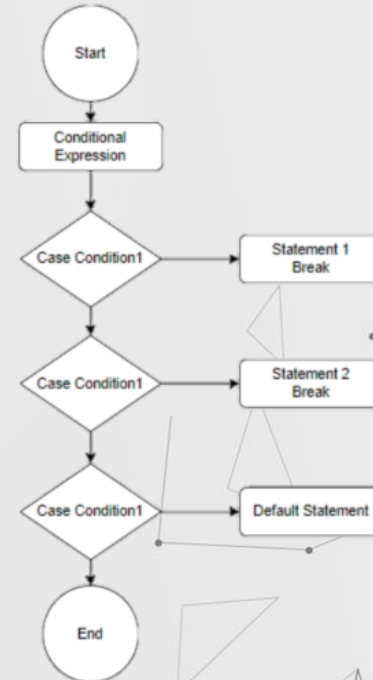


Switch Case / Match Case

Switch Case statement memungkinkan kita untuk menentukan control flow berdasarkan nilai variabel atau ekspresi. Fungsi switch case statement mirip dengan 'if' statement, hanya saja dengan metode yang berbeda.

```
match x:  
    case 'a':  
        print(1)  
    case 'b':  
        print(2)  
    case _:  
        print(0)
```

0 is the default case if x is not found





02

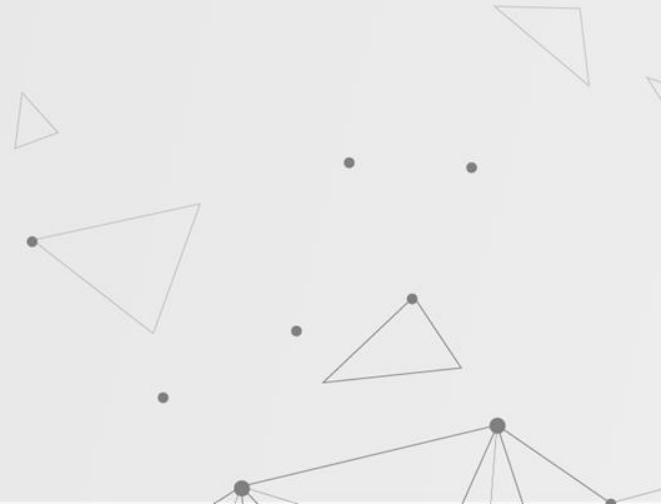
BASIC PROGRAMMING II ITERATION

Iteration / Loop

Iterasi artinya mengulangi sebagian instruksi dari program, atau dengan kata lain, kita menginstruksikan program untuk melakukan perulangan. Bahasa pemrograman akan melacak langkah iterasi apa saat ini sedang dieksekusi. Pada dasarnya, iterasi akan berulang sampai kondisi terminasi terpenuhi

Di dalam bahasa pemrograman Python pengulangan dibagi menjadi 3 bagian, yaitu :

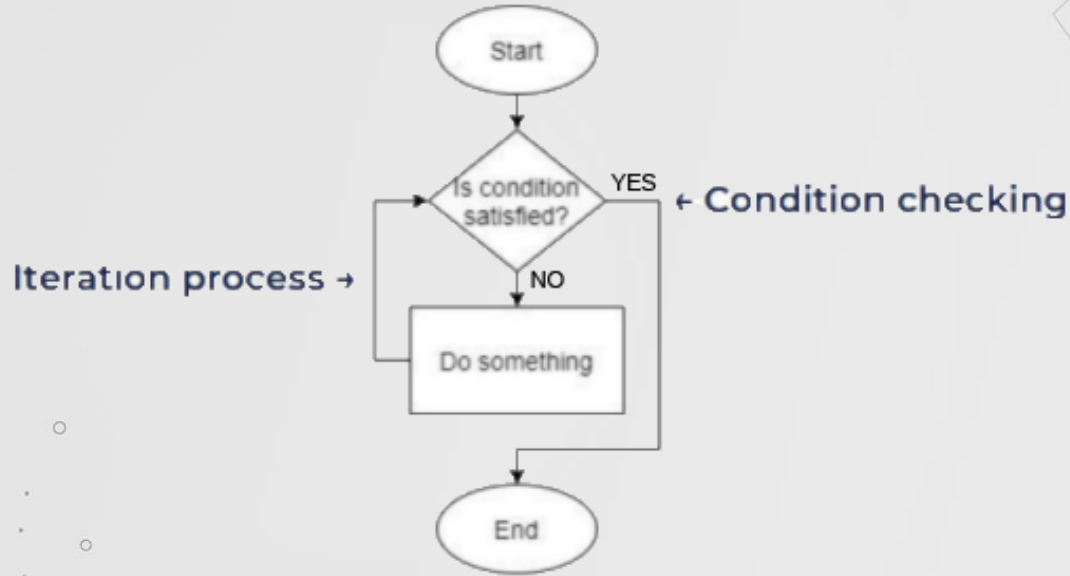
- ❑ While Loop
- ❑ For Loop
- ❑ Nested Loop



Flowchart

Every iteration must have these two parts:

1. Condition checking
2. Iteration process

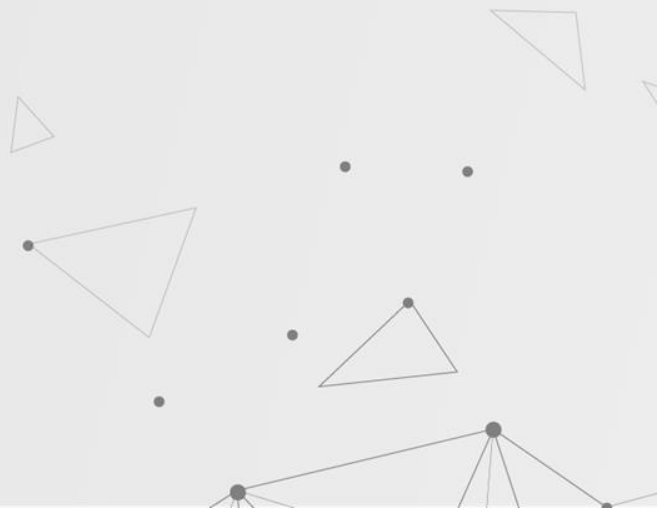


While Loop

While Loop adalah perulangan uncountable atau perulangan yang jumlah proses pengulangannya tidak ditentukan. Ia akan menjalankan baris kode di dalam blok kodenya secara terus menerus selama masih memenuhi ekspresi yang sudah ditentukan sebelumnya, yang berarti ia akan terus mengulang selama kondisi bernilai True.

```
i= 1  
While i < 6:  
    print(i)  
    i += 1
```

Output: **1**
 2
 3

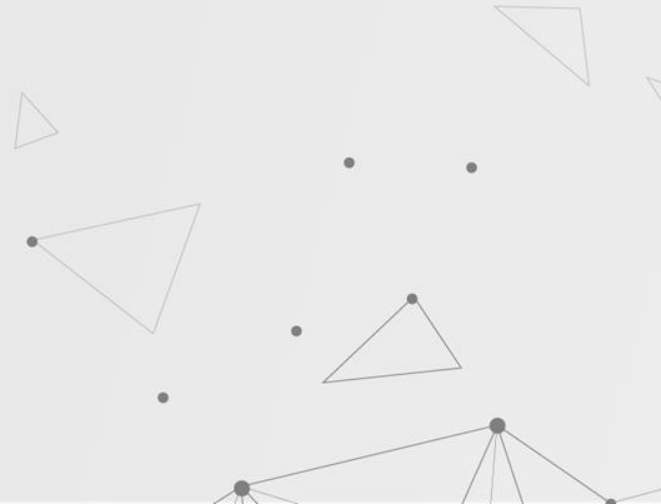


For Loop

For Loop pada python adalah perintah yang digunakan untuk melakukan iterasi dari sebuah nilai sequence atau data koleksi pada python seperti List, Tuple, String dan lain-lain. Pada iterasi For Loop jumlah pengulangan ditentukan secara eksplisit sebelumnya.

```
fruits = ["apple", "banana", "cherry"]  
for x in fruits :  
    print(x)
```

Output: **apple**
banana
cherry



Nested Loop

Nested loop atau loop bersarang, bahasa pemrograman python mengizinkan penggunaan loop didalam loop. Tentu hal ini akan berimbas pada penggunaan memori, sehingga pastikan looping yang digunakan tidak mubazir.

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

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

```
Output: red apple  
        red banana  
        red cherry  
        big apple  
        big banana  
        big cherry  
        tasty apple  
        tastybanana  
        tastycherry
```

Loop Control

Python juga mendukung penggunaan kontrol di dalam looping. Ada 3 kontrol, yaitu :

- **break** → untuk menghentikan looping ketika terjadi kondisi tertentu.
- **continue** → untuk melanjutkan operasi, ketika pada blok statement menghasilkan nilai yang diharapkan atau yang dicari.
- **pass** → kontrol ini tidak menghasilkan apa-apa, pass akan berguna untuk mengecek apakah statement berjalan apa tidak






03

BASIC PROGRAMMING III

Array and Other Data Types



Type Data Array

- Tipe data array adalah variabel yang bisa menyimpan lebih dari satu data.
 - Dapat menyimpan berbagai macam tipe data.
 - Bisa diakses menggunakan indeks
 - Penomoran indeks dimulai dari 0 (nol) sampai dengan N-1 (dengan N adalah jumlah elemen data)
 - Proses inisialisasinya menggunakan kurung siku []
 - Setiap elemen dipisahkan dengan tanda koma (,)
- 

Type Data List

- Koleksi data yang bersifat ordered (berurutan) dan changeable (bisa diubah)
- Proses inisialisasinya menggunakan kurung siku []
- Tipe data list bisa menyimpan berbagai macam tipe data

- Contoh

- Kumpulan data string

```
nama_kota = ['Jakarta', 'Bogor', 'Depok', 'Tangerang', 'Bekasi']
```

- Kumpulan data integer

```
bilangan_bulat = [1,2,3,4,5,6,7]
```

- Kumpulan multiple tipe data

```
list_jawaban = [150, 33.33, 'Presiden Sukarno', False]
```



Type Data List

Built-in Method List

Append

Berfungsi untuk menambahkan elemen baru pada akhir list

```
▶ nama_kota = ['Jakarta', 'Bogor', 'Depok', 'Tangerang', 'Bekasi']
```

```
[24] nama_kota.append('Surabaya')
```

```
[25] print (nama_kota)
```

```
['Jakarta', 'Bogor', 'Depok', 'Tangerang', 'Bekasi', 'Surabaya']
```

Type Data List

Built-in Method List

Clear

Menghapus semua item pada list

```
[26] nama_kota = ['Jakarta', 'Bogor', 'Depok', 'Tangerang', 'Bekasi']
```

```
[28] nama_kota.clear()  
      print (nama_kota)
```

```
[]
```



Type Data List

Built-in Method List

Copy

Menduplikat elemen pada suatu list

```
[33] nama_kota = ['Jakarta', 'Bogor', 'Depok', 'Tangerang', 'Bekasi']
      jabodetabek = nama_kota.copy()

      print ('nama kota ', jabodetabek)
      print ('jabodetabek: ', jabodetabek)

nama kota  ['Jakarta', 'Bogor', 'Depok', 'Tangerang', 'Bekasi']
jabodetabek:  ['Jakarta', 'Bogor', 'Depok', 'Tangerang', 'Bekasi']
```


Type Data List

Built-in Method List

Count

Menghitung suatu value yang diinginkan

```
[36] nama_kota = ['Jakarta', 'Bogor', 'Depok', 'Tangerang', 'Bekasi', 'Bogor', 'Bekasi', 'Depok', 'Bogor']
```

```
nama_kota.count('Bogor')
```

```
3
```

```
[37] nama_kota.count('Depok')
```

```
2
```

```
[38] nama_kota.count('Jakarta')
```

```
1
```

Type Data List

Built-in Method List

Index

Mengetahui value yang didefinisikan berada di indeks ke berapa

```
[44] nama_kota = ['Jakarta', 'Bogor', 'Depok', 'Tangerang', 'Bekasi']
```

```
[47] nama_kota.index('Bekasi')
```

```
4
```

Type Data List

Built-in Method List

Insert

Menambahkan value ke dalam list berdasarkan indeks ke berapa yang diinginkan

```
[51] nama_kota = ['Jakarta', 'Bogor', 'Depok', 'Tangerang', 'Bekasi']  
  
      nama_kota.insert(3, 'Malang')  
  
      print (nama_kota)  
  
['Jakarta', 'Bogor', 'Depok', 'Malang', 'Tangerang', 'Bekasi']
```

Type Data List

Built-in Method List

Pop

Menghapus value terakhir pada list atau juga bisa menghapus value pada posisi yang didefinisikan.

```
[52] nama_kota = ['Jakarta', 'Bogor', 'Depok', 'Tangerang', 'Bekasi']

      nama_kota.pop()

      print (nama_kota)

['Jakarta', 'Bogor', 'Depok', 'Tangerang']

[54] nama_kota = ['Jakarta', 'Bogor', 'Depok', 'Tangerang', 'Bekasi']

      nama_kota.pop(1)

      print (nama_kota)

['Jakarta', 'Depok', 'Tangerang', 'Bekasi']
```

Type Data List

Built-in Method List

Remove

Menghapus value pada list sesuai dengan value yang didefinisikan

```
[57] nama_kota = ['Jakarta', 'Bogor', 'Depok', 'Tangerang', 'Bekasi']  
  
      nama_kota.remove('Tangerang')  
  
      print(nama_kota)  
  
      ['Jakarta', 'Bogor', 'Depok', 'Bekasi']
```

Type Data List

Built-in Method List

Reverse

Membalikkan urutan value yang ada di dalam suatu list

```
[58] nama_kota = ['Jakarta', 'Bogor', 'Depok', 'Tangerang', 'Bekasi']  
  
      nama_kota.reverse()  
  
      print(nama_kota)  
  
['Bekasi', 'Tangerang', 'Depok', 'Bogor', 'Jakarta']
```

Type Data List

Built-in Method List

Sort

- Mengurutkan value secara ascending di dalam suatu list.
- Apabila tipe datanya integer maka akan diurutkan dari yang terkecil.
- Apabilan tipe datanya string diurutkan berdasarkan A sampai Z

```
[59] nama_kota = ['Jakarta', 'Bogor', 'Depok', 'Tangerang', 'Bekasi']  
      nilai_siswa = [97, 78, 90, 65, 45, 92, 100, 69]  
  
      nama_kota.sort()  
      nilai_siswa.sort()  
  
      print(nama_kota)  
      print(nilai_siswa)  
  
      ['Bekasi', 'Bogor', 'Depok', 'Jakarta', 'Tangerang']  
      [45, 65, 69, 78, 90, 92, 97, 100]
```

Type Data Tuple

- Koleksi data yang bersifat ordered (berurutan) dan unchangeable (tidak bisa diubah)
- Proses inisialisasinya menggunakan tanda kurung ()
- Tipe data list bisa menyimpan berbagai macam tipe data
 - Contoh
 - Kumpulan data string

```
[65] nama_kota = ('Jakarta', 'Bogor', 'Depok', 'Tangerang', 'Bekasi')
```

- Kumpulan data

```
[63] nilai_siswa = (97, 78, 90, 65, 45, 92, 100, 69)
```

- Kumpulan multiple tipe

```
[64] list_jawaban = (150, 33.33, 'Presiden Sukarno', False)
```


Type Data Tuple

Built-in Method Tuple

Sequence Unpacking

Mengekstrak isi dari tuple ke dalam variabel-variabel tunggal secara berurutan

```
[62] siswa = ('Nurul Huda', 'Bangkalan', 24)
```

```
nama, asal, usia = siswa
```

```
print('Asal:', asal)
```

```
print('Usia:', usia)
```

```
print('Nama:', nama)
```

```
Asal: Bangkalan
```

```
Usia: 24
```

```
Nama: Nurul Huda
```

Type Data Tuple

Built-in Method Tuple

Len

Menghitung jumlah value yang ada di dalam tuple

```
[71] nama_kota = ('Jakarta', 'Bogor', 'Depok', 'Tangerang', 'Bekasi')
```

```
len(nama_kota)
```

```
5
```

Type Data Tuple

Built-in Method Tuple

Max

Menampilkan nilai paling besar dari value yang ada di dalam suatu tuple

```
[73] nilai_siswa = (97, 78, 90, 65, 45, 92, 100, 69)
```

```
max(nilai_siswa)
```

```
100
```

Type Data Tuple

Built-in Method Tuple

Min

Menampilkan nilai paling kecil dari value yang ada di dalam suatu tuple

```
[75] nilai_siswa = (97, 78, 90, 65, 45, 92, 100, 69)
```

```
min(nilai_siswa)
```

```
45
```

Type Data Set

- Tipe data yang menyimpan banyak nilai dalam satu variabel dengan beberapa ketentuan:
 - nilai anggota yang disimpan harus unik (tidak duplikat)
 - nilai anggota yang sudah dimasukkan tidak bisa diubah lagi
 - set bersifat unordered alias tidak berurut (tidak bisa diakses dengan indeks)
- Tipe data set juga dapat terdiri dari multiple data
- Proses inisialisasinya menggunakan tanda kurung kurawal { }



Type Data Dictionary

- Dictionary adalah tipe data pada python yang berfungsi untuk menyimpan kumpulan data/nilai dengan pendekatan “key-value”.
- Dictionary sendiri memiliki dua buah komponen inti:
 - Key merupakan nama atribut suatu item pada dictionary.
 - Value adalah nilai yang disimpan pada suatu atribut.
- Dictionary items memiliki 3 sifat, yaitu:
 - Unordered - tidak berurutan
 - Changeable - bisa diubah
 - Unique - tidak bisa menerima dua keys yang sama
- Berikutnya adalah bagaimana cara membuat dictionary pada python. Untuk membuatnya, terdapat 2 cara, yaitu:
 - Menggunakan tanda kurung kurawal {}.
 - Menggunakan fungsi atau konstruktor dict().

Tipe Data Dictionary

Cara Penulisan

```
# cara pertama
buku = {
    "judul": "Daun Yang Jatuh Tidak Pernah Membenci Angin",
    "penulis": "Tere Liye"
}

# cara kedua
buku = dict(
    judul="Daun Yang Jatuh Tidak Pernah Membenci Angin",
    penulis="Tere Liye"
)
```

Tipe Data Dictionary

Mengakses item pada dictionary

Mengakses item pada dictionary dapat dilakukan dengan 2 cara, yaitu:

- Menggunakan kurung siku []
- Menggunakan fungsi get ()

```
✓ [5] print('Judul:', pertemuan_hari_ini.get('judul'))  
0s # atau  
print('Tanggal:', pertemuan_hari_ini['tanggal'])
```

```
Judul: Belajar Dictionary Pada Python 3  
Tanggal: 01 Februari 2021
```


Tipe Data Dictionary

Mengubah nilai item pada dictionary

```
✓ [8] mahasiswa = {  
0s   'nama': 'Lendis Fabri',  
      'asal': 'Indonesia'  
}  
  
# mengubah data  
print('Nama awal:', mahasiswa.get('nama'))  
mahasiswa['nama'] = 'Andi Mukhlis'  
print('Setelah diubah:', mahasiswa.get('nama'))
```

```
Nama awal: Lendis Fabri  
Setelah diubah: Andi Mukhlis
```

Tipe Data Dictionary

Menambahkan item pada dictionary

```
09 ▶ mahasiswa = {  
    'nama': 'Lendis Fabri',  
    'asal': 'Indonesia',  
}  
# tambah data  
mahasiswa['hobi'] = 'Memancing'  
# print ulang  
print('Hobi dari {} adalah {}'.format(  
    mahasiswa.get('nama'),  
    mahasiswa.get('hobi')  
))
```

📄 Hobi dari Lendis Fabri adalah Memancing

Tipe Data Dictionary

Menghitung jumlah key

```
✓ [17] sekolah = {  
0s     'nama': 'Sekolah Dasar Negeri Surabaya 1',  
       'jenjang': 'Sekolah Dasar',  
       'akreditasi': 'A'  
}  
  
print(  
    "Jumlah atribut variabel sekolah adalah:",  
    len(sekolah)  
)
```

➡ Jumlah atribut variabel sekolah adalah: 3

**THANK
YOU**

