

## Hello World

Perintah untuk mencetak/menampilkan data

```
# Untuk mencetak data di Python menggunakan perintah "print('')""  
print("Hello, teman-teman!");
```

```
➦ Hello, teman-teman!
```

## Variable

Variable adalah suatu entitas yang digunakan untuk menyimpan data pada sebuah memori

```
# Cara untuk mendeklarasikan Variable sebagai berikut  
a = 100  
# Perintah di atas berarti telah didefinisikan Variable "a" dengan nilai "5".
```

```
# Untuk mencetak atau melihat isi variable berikut bisa menggunakan perintah "print" seperti berikut  
print(a)
```

```
➦ 100
```

```
# Kita juga mencetaknya seperti ini:  
print("variabel a memiliki nilai =", a)
```

```
➦ variabel a memiliki nilai = 100
```

## Operators

Operators adalah suatu simbol yang digunakan untuk operasi tertentu

Operators dibagi menjadi 2:

- Arithmetic Operators
- Assignment Operators

Arithmetic Operators digunakan ketika ingin mengoperasikan operasi matematika

```
# Pertama kita mendefinisikan variabel dengan nilai tertentu  
x = 14  
y = 4
```

```
# Penjumlahan  
print(x+y) # Output: x + y = 18
```

```
➦ 18
```


```
# Pengurangan  
print('x - y =', x-y) # Output: x - y = 10
```

```
➦ x - y = 10
```

```
# Perkalian  
print('x * y =', x*y) # Output: x * y = 56
```

```
➦ x * y = 56
```


```
# Pembagian  
print('x / y =', x/y) # Output: x / y = 3.5
```

 `x / y = 3.5`


```
# Floor division (quotient) -> hasil bagi
print('x // y =', x//y) # Output: x // y = 3
```

 `x // y = 3`

```
# Modulus (sisir hasil bagi)
print('x % y =', x%y) # Output: x % y = 2
```

 `x % y = 2`

```
# Perpangkatan
print('x ** y =', x**y) # Output: x ** y = 38416
```

 `x ** y = 38416`

Assignment Operators digunakan ketika ingin menetapkan nilai dalam suatu variable

```
# Perintah "=" digunakan untuk memasukkan nilai dalam suatu variable
# Kita sudah melakukan perintah ini sebelumnya
```

```
x = 10
```

```
# Selanjutnya "+=", perintah ini untuk menambahkan nilai ke dalam variable yang sudah ada
# x += 5 ----> x = x + 5
x +=5
print(x) # Output: 10
```

 `15`

```
# Selanjutnya "/=", perintah ini untuk membagi nilai ke dalam variable yang sudah ada
# x /= 5 ----> x = x / 5
x /= 5
print(x) # Output: 2.0
```


 `3.0`

## Tipe Data

Python memiliki beberapa tipe data, diantaranya:

1. Integer (1)
2. Float (1.0)
3. String ("Satu")

```
z = 5.0
y = 5.0
print(z+y)
```

 `10.0`

## Data Structures

Untuk manajemen suatu kumpulan data, Python memiliki beberapa Data Structures

## Lists

List adalah suatu tempat penyimpanan data di dalam sebuah kurung siku (square bracket) "[]" yang dipisahkan oleh koma. Data yang disimpan dapat berbentuk Integer, Float, dan String.

```
# list kosong
my_list = []

# kumpulan list dengan tipe data integer
my_list = [1, 2, 3]

# kumpulan list dengan berbagai macam tipe data
my_list = [1, "Hello", 3.4]
```

# Untuk mengakses nilai list sebagai berikut

```
names = ["toni", #0
         "tono", #1
         "tina"] #2
```

```
my_list = [1, "Hello", 3.4]
my_list[0] = 2
```

```
print(my_list)
print(names)
print(names[-2])
```

```
# Mengakses data terakhir
# print(names[2])
```

```
⇒ [2, 'Hello', 3.4]
   ['toni', 'tono', 'tina']
   tono
```

## Tuples

Sama dengan Lists, perbedaannya dengan Tuples kita tidak dapat mengubah isi datanya ketika sudah didefinisikan. Selain itu, tuple didefinisikan menggunakan kurung biasa "()".

```
names = ("toni", "tono", "tina")
# names[0] = "aurora" #tidak bisa diubah isi data ketika sudah didefinisikan
```

```
print(names)
print(names[0])
```

```
# names = ["toni", "tono", "tina"]
# print(names)
# names[0] = "hehe"
```

```
⇒ ('toni', 'tono', 'tina')
   toni
```

## Sets

Sets adalah tempat penyimpanan dari kumpulan *unique* data.

```
# Kumpulan set integers
my_set = {1, 2, 3}
print(my_set)

# set dengan campuran tipe data
my_set = {1.0, "Hello", (1, 2, 3)}
print(my_set)
```

```
⇒ {1, 2, 3}
   {1.0, (1, 2, 3), 'Hello'}
```

```
# Set juga dapat dilakukan untuk mendapat nilai unique dari suatu kumpulan data

data = ["aki", "akbar", "akbar", "nina", "lala", "lina", "ilham", "lina"]

print(set(data))
```

```
⇒ {'lala', 'ilham', 'aki', 'lina', 'akbar', 'nina'}
```

## Dictionaries

Jika tempat penyimpanan sebelumnya hanya dapat menyimpan *value* saja, ketika memakai Dictionaries ini kita juga dapat menyimpan *key* dan *value*.

```
# # Dictionary kosong
# my_dict = {}

# # Dictionary keys berbentuk Integer
# my_dict = {1: 'apple', 2: 'ball'}

# # Dictionary dengan keys Integer dan String
# my_dict = {'name': 'John', 1: [2, 4, 3]}

data = {'students':
    [{
        'name': 'Lala',
        'age': 20
    },
    {
        'name': 'Nora',
        'age': 25
    }]
}
```

```
# Kita bisa mengakses value hanya dengan mengakses key-nya saja
person = {'name': 'Jack', 'age': 26, 'salary': 4534.2}
print(person['age'])
```

```
⇒ 26
```

## If...Else Statement (Percabangan)

If...Else Statement digunakan ketika akan melakukan aksi berbeda dengan kondisi tertentu

```
alamat = 'semarang'

if alamat == 'jogja':
    print('boleh masuk!')
elif alamat == 'sleman':
    print('masih boleh masuk')
else:
    print('periksa dulu')

# num = -1

# if num > 0:
#     print("Angka positif")
# elif num == 0:
#     print("Nol")
# else:
#     print("Angka negatif")
```

➡️ periksa dulu

### While Loop Statement (Perulangan)

While Loop Statement digunakan untuk melakukan iterasi kumpulan data selama kondisinya terpenuhi.

```
n = 100

# Inisiasi sum dan counter
sum = 0
i = 2

while i <= n:
    sum = sum + i
    i = i+1    # update counter

print("Jumlah totalnya adalah", sum)
```

➡️ Jumlah totalnya adalah 5049

### For Loop Statement (Perulangan)

For Loop Statement digunakan untuk melakukan iterasi kumpulan data.

```
# numbers = [6, 5, 3, 8, 4, 2]
names = ["lala", "doni", "dino", 3]

for n in names:
    print(n)
# sum = 0

# # Iterasi kumpulan data
# for val in numbers:
#     sum = sum+val

# print("Total: ", sum)
```

➡️ lala  
doni  
dino  
3

### Break Statement

Break Statement digunakan untuk menyetop suatu proses perulangan.

```
for val in "string":
    if val == "r":
        break
    print(val)

# print("The end")
```

```
⇒ s
t
```

### Continue Statement

Continue Statement digunakan untuk melakukan skip suatu code.

```
for val in "string":
    if val == "r":
        continue
    print(val)

# print("The end")
```

```
⇒ s
t
i
n
g
```

### Pass Statement

```
sequence = ['a', 'b', 'c']
for val in sequence:
    pass
    if val == 'c':
        print("oke")
    else:
        pass
```

```
⇒ oke
```

### Function

Function atau Fungsi adalah sebuah kumpulan script untuk spesifik task

```
def ages():
    print(25)
    print(26)

def names():
    print("nana")
    print("tino")

names()
ages()

# def print_lines():
#     print("Cetak 1.")
#     print("Cetak 2.")

# menjalankan fungsi
# print_lines()
```

```
⇒ nana
tino
25
26
```

# Fungsi dapat digunakan untuk mengoperasikan proses aritmatika

```
def add_numbers(a, b):  
    sum = a + b  
    return sum  
  
result = add_numbers(4, 5)  
print(result)
```

➡ 9

## Lambda Function

Lambda: Fungsi yang tidak memiliki nama

```
data = lambda x: x - 2  
  
print(data(5))
```

➡ 3

Double-click (or enter) to edit

## File I/O

File I/O perintah dalam Python yang digunakan ketika berhubungan dengan file. Diantaranya:

1. Write File
2. Open File

### Write File

```
with open("test.txt", 'w', encoding = 'utf-8') as f:  
    f.write("Lala")  
    f.write("Dino")  
    f.write("Dono")
```

### Read File

```
f = open("test.txt", 'r', encoding = 'utf-8')  
print(f.read() )
```

➡ LalaDinoDono

## Dates and Times

```
from datetime import date  
  
today = date.today()  
print("Today's date:", today)
```

➡ Today's date: 2024-08-19

```
from datetime import date

today = date.today()

# dd/mm/YY
d1 = today.strftime("%d/%m/%Y")
print("d1 =", d1)

# Textual month, day and year
d2 = today.strftime("%B %d, %Y")
print("d2 =", d2)

# mm/dd/y
d3 = today.strftime("%m/%d/%y")
```