Python:
Function and Looping

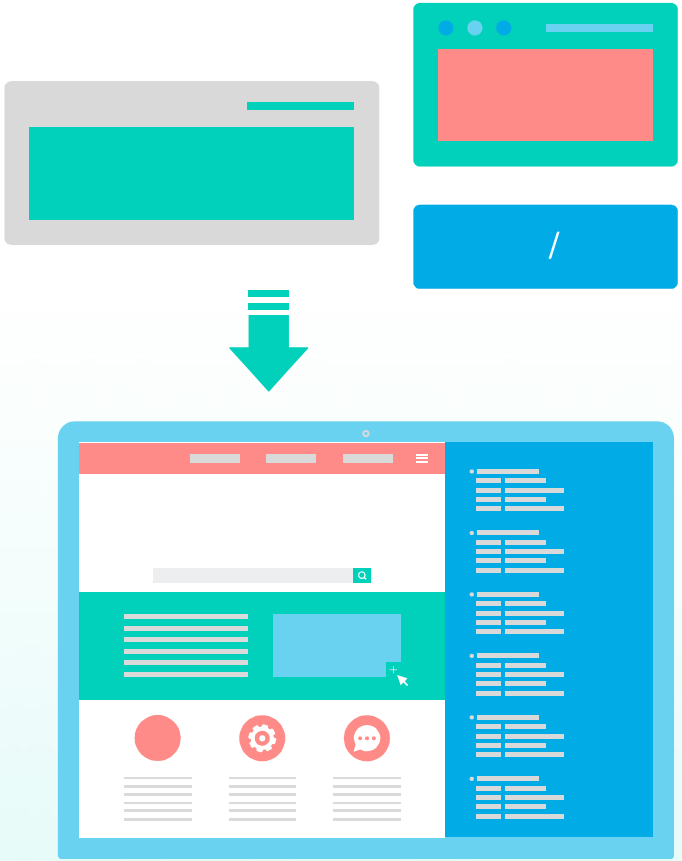
By Big Three:

Ilham Ahsanudin

Ilyas Rizky Ibrahim

Harryyanto Ishaq Agasi

Table of contents

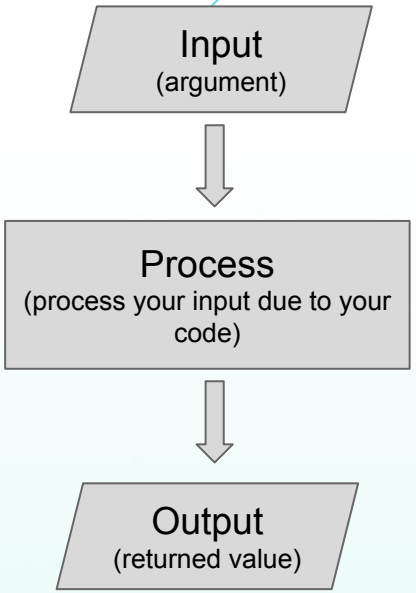


1

Introduction of Python function

What is function?

- A function is a block of code which only runs when it is called.
- You can pass data, known as parameters, into a function.
- A function can return data as a result.



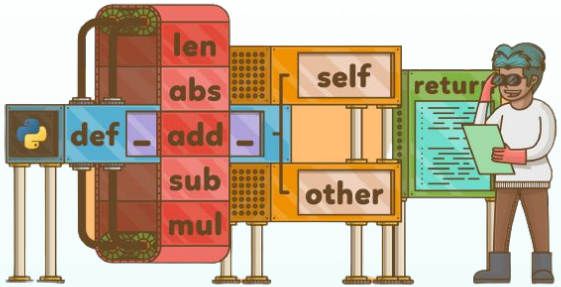
https://www.w3schools.com/python/python_functions.asp

Why do we need function?

In Python, a function is a group of related statements that performs a specific task.

Functions help break our program into smaller and modular chunks. As our program grows larger and larger, functions make it more organized and manageable.

Furthermore, it avoids repetition and makes the code reusable.



Real Python

DRY (do not repeat yourself) principle

/* Every piece of knowledge must have a single, unambiguous, authoritative representation within a system */

By Andy Hunt and Dave Thomas in their book “The Pragmatic Programmer”

2

How to make and use function?

Function Syntax

1. Keyword `def` that marks the start of the function header.
2. A function name to uniquely identify the function. Function naming follows the same rules of writing identifiers in Python.
3. Parameters (arguments) through which we pass values to a function. They are optional.
4. A colon (:) to mark the end of the function header.
5. Optional documentation string (docstring) to describe what the function does.
6. One or more valid python statements that make up the function body. Statements must have the same indentation level (usually 4 spaces).
7. An optional `return` statement to return a value from the function.

```
def function_name(parameters):
    """docstring"""
    statements
```

Function Syntax

In Python a function is defined using the `def` keyword

```
def my_function():
    print("Hello from a function")


my_function()
```

To call a function, use the function name followed by parenthesis:

output:

```
PS C:\                                  > python function_.py
Hello from a function
```

Parameter and argument

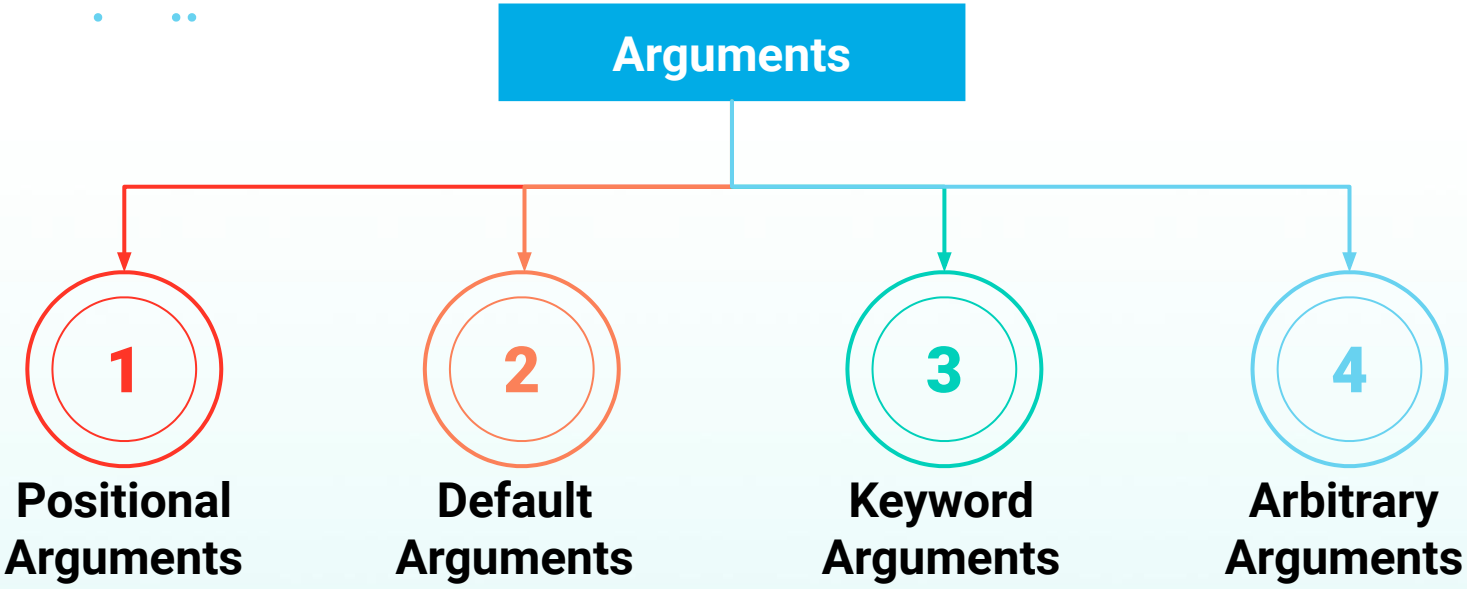
- Parameter is a variable listed inside a function for holds a value from argument.
- Argument is a value that is sent to the function when it's called.

Parameter

```
def func1(angka1, angka2):
    hasil = angka1 + angka2
    return hasil

hasil = func1(1, 3)
print(hasil)
```

Argument

Arguments Type



Positional Argument

The position argument is an argument that the values of an arguments should be in the same order with the parameters.

Parameter 1 Parameter 2

```
def func1(angka1, angka2):
    hasil = angka1 + angka2
    return hasil

hasil = func1(1, 3)
print(hasil)
```

Argument 1 Argument 2

Output :

Console Shell

```
4
>
```

Default Argument

The default argument is an argument where if no value is passed to the parameter, the default argument will be used

Parameter 1 Parameter 2 Default value of parameter 2

```
1 def kary(nama, alamat="Bandung"):
2     print("Nama saya", nama,
3     "dan alamat saya di", alamat)
4
5 data_karyawan = kary("Jhon")
```

Argument 1

Output :

Console Shell

```
Nama saya Jhon dan alamat saya di Bandung
>
```

Keyword Argument

We can give value of argument explicitly, without concern the parameter order.

Parameter 1 Parameter 2

```
1  def kary(nama, alamat):
2      print("Nama saya", nama,
3      "dan alamat saya di", alamat)
4
5  data_karyawan = kary(alamat="Semarang"
6                       , nama="Jhon")
```

Argument 1 Argument 2

Output :

Console Shell

```
Nama saya Jhon dan alamat saya di Semarang
>
```

Arbitrary Argument

1. ***args**

when we don't know how much argument we need, we can use Arbitrary Arguments method. We can use asterisk sign (*) in parameter.

Parameter 2 with asterisk sign

Parameter 1

main.py

```
1  def kary(nama, *hobi):
2      print(nama, "mempunyai hobi", hobi)
3
4  kary("Jhon", "membaca", "naik gunung", "berenang")
```

Argument 1

Argument 2

Output :

Console Shell

```
Jhon mempunyai hobi ('membaca', 'naik gunung', 'berenang')
>
```

Arbitrary Argument

2. **kwargs

Meanwhile ***kwargs* is used to pass a keyword, variable-length argument list. We use double asterisk sign (**)

Parameter 1 with asterisk sign

main.py

```
1  def kary(**movie):
2      print(movie)
3
4  kary(comedy="Shaolin Soccer", horror="Saw")
```

Argument 1 with keyword value

Output :

Console Shell

```
{'comedy': 'Shaolin Soccer', 'horror': 'Saw'}
>
```

Return Type

With Return Statement

Without Return Statement

With Return Statement

Return statement is used when we want to return the value that we passed to function, so we can use the return value for another process in program.

main.py

```python
def func1(angka1, angka2):
    hasil = angka1 + angka2
    return hasil

hasil1 = func1(1,3)
hasil2 = func1(2,3)

print(hasil1)
print(hasil2)
```

Console Shell

```
4
5
>
```

Without Return Statement

Because if we're not using return statement, when we print *hasil1*, *hasil2* the result is **None.** So the value is cannot used again for another process in program.

main.py

```python
def func1(angka1, angka2):
    hasil = angka1 + angka2

hasil1 = func1(1,3)
hasil2 = func1(2,3)

print(hasil1)
print(hasil2)
```

Console Shell

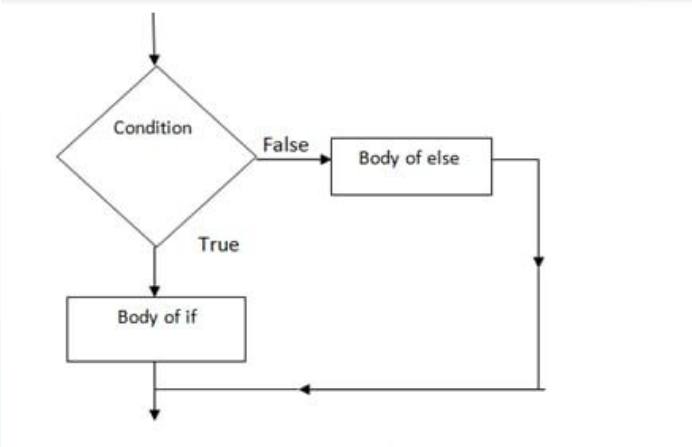
```
None
None
>
```

3

Looping

Control Flow Logic

IF Statements



When one condition is false and the justification is in the requirement for one condition IF Statement is applied.

Only depending on whether the text expression is true, the statements are executed.

Indentation indicates the body of the IF Statement in python.

Python often interprets non-zero values as True whereas None and 0 are interpreted as False.

Control Flow Logic

How to add logic to our program?

```python
a = 60
b = 100

if a > b:
    print("a is greater than b")

else:
    print("b is greater than a")
```

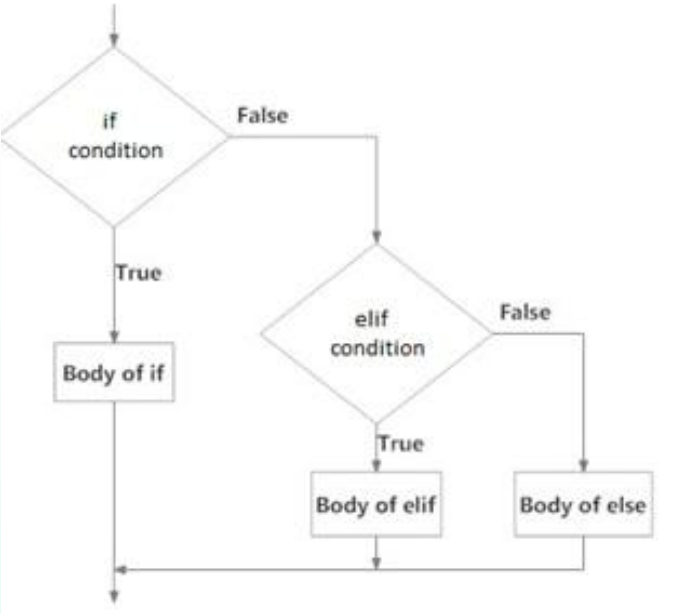
Case 1	Case 2
a = 60	a = 100
b = 100	b = 60

Output:

```
e c:/[redacted]/if.py
b is greater than a
```

Control Flow Logic

Case 1	Case 2	Case 3
a = 60	a = 100	a = 100
b = 100	b = 60	b = 100




```python
a = 100
b = 100

if a > b:
    print("a is greater than b")

elif a == b:
    print("a and b are equal")

else:
    print("b is greater than a")
```

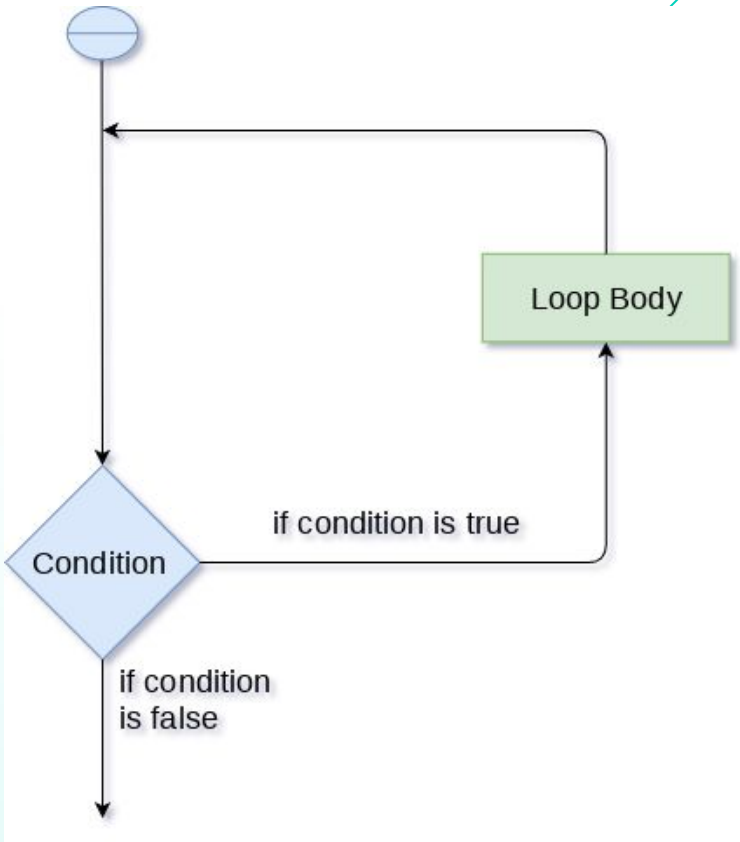
Output:

```
e c:/[redacted]/if.py
a and b are equal
```

For Loops

A `for` loop is used for iterating over a sequence (that is either a list, a tuple, a dictionary, a set, or a string).

This is less like the `for` keyword in other programming languages, and works more like an iterator method as found in other object-orientated programming languages.



For Loops

In Python a for looping is defined using the `for` keyword

```
for x in range(10):
    print(str(x+1) + ".", "I studying Python in Digital Skola")
```

Output:

```
PS C:\[redacted]> python forLoop.py
1. I studying Python in Digital Skola
2. I studying Python in Digital Skola
3. I studying Python in Digital Skola
4. I studying Python in Digital Skola
5. I studying Python in Digital Skola
6. I studying Python in Digital Skola
7. I studying Python in Digital Skola
8. I studying Python in Digital Skola
9. I studying Python in Digital Skola
10. I studying Python in Digital Skola
```

For Loops

In another case, if you're asked by the company's supervisor to calculate the amount of incoming money from all transactions on that day by using a program, what you gonna do then?

This brief explanation might help you

```python
total = 0
n_transaksi = 10

for x in range(n_transaksi):
    nilai_kuitansi = int(input())
    total = total + nilai_kuitansi


print("Total uang yang masuk pada hari ini adalah Rp." + str(total))
```

Output:

```
PS C:\[redacted]> python forLoop.py
17000
25000
20000
5000
7500
31000
90000
152000
7500
12000
Total uang yang masuk pada hari ini adalah Rp.367000
```

While Loops

The while loop in Python is used to iterate over a block of code as long as the test expression (condition) is true.

While loop is different to for loop because in for loop we will iterate over all sequence of object, while in while loop we will exit the loop when the test expression is false.

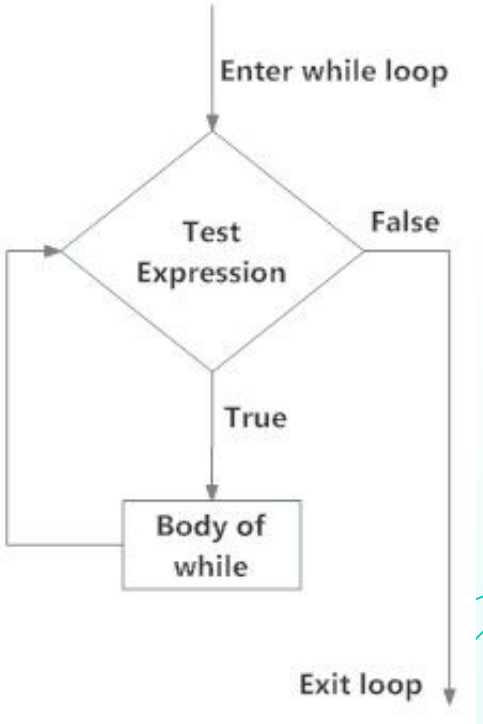


Fig: operation of while loop

https://www.programiz.com/python-programming/while-loop

While Loops

while loop keyword

```
## while loop syntax ##
avenger = input("are you an Avenger? ")
while avenger == "yes":
    print("I can do this all day")
    avenger = input("are you an Avenger? ")
```

Declaration of a variable, input value

Test expression

The while loop body

Case update

output

```
(base) ishaq@ishaq:~/[redacted]de$ python test.py
are you an Avenger? yes
I can do this all day
are you an Avenger? yes
I can do this all day
are you an Avenger? yes
I can do this all day
are you an Avenger? yes
I can do this all day
are you an Avenger? No, I'm Batman!
(base) ishaq@ishaq:~/[redacted]de$
```

Run the python script in it's directory

Inputting initial value of avenger

As long as we input "yes" string, it will print "I can do this all day", but when we input any string except "yes" it will exit the loops.

Nested Loops

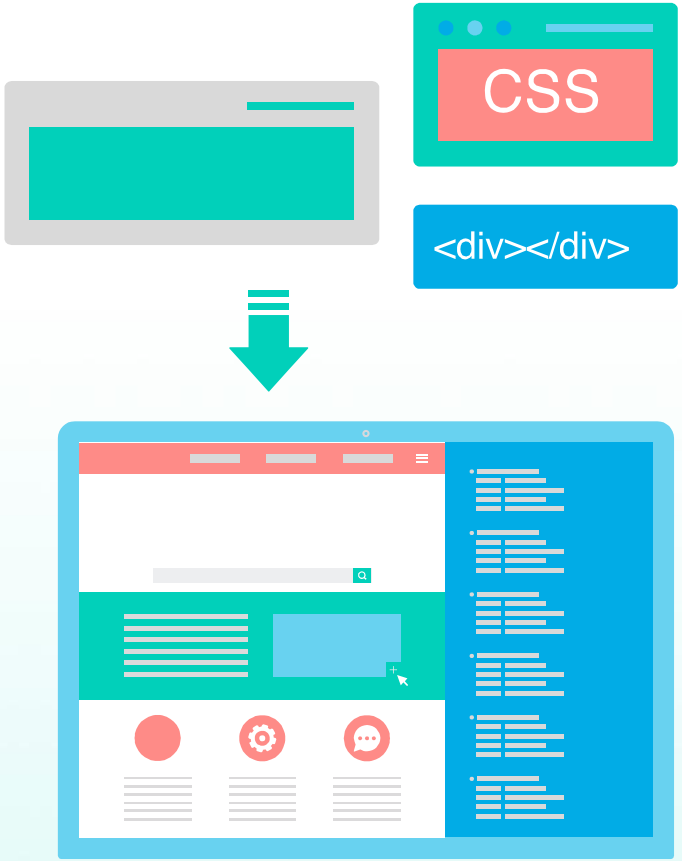
```python
## Nested loop ##
def squarePattern(n):
    for i in range (0,n):
        for j in range (0,n):
            print(" # ",end="")
        print('')

squarePattern(5)
```

for nested loop

output

```
(base) ishaq@ishaq:~/Doc[redacted]ode$ python test.py
 #  #  #  #  #
 #  #  #  #  #
 #  #  #  #  #
 #  #  #  #  #
 #  #  #  #  #
```

Running the script



4

Extras

Random password generator function with looping concept

```python
## random password generator ##
from random import randint

def password_generator(length=8):
    password = ''
    letters = 'abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ'
    numbers = '1234567890'
    spec_chr = "!@#$%^&*()_+-=,./;'[]\<>?:}{|"

    for i in range(length):
        randomChr = [letters[randint(0,len(letters)-1)],numbers[randint(0,len(numbers)-1)],spec_chr[randint(0,len(spec_chr)-1)]]
        password += randomChr[randint(0,len(randomChr)-1)]
    print("your password: ", password)

password_generator(12)
```

Importing the Random module

output

```
(base) ishaq@ishaq:~/Documents/DATA ENGINEER - CLASS/code$ python test.py
your password:  676=1F4445.+
(base) ishaq@ishaq:~/Documents/DATA ENGINEER - CLASS/code$ python test.py
your password:  j53F>Dpl=.>6
(base) ishaq@ishaq:~/Documents/DATA ENGINEER - CLASS/code$ python test.py
your password:  9;bj8&B7ut6j
(base) ishaq@ishaq:~/Documents/DATA ENGINEER - CLASS/code$ python test.py
your password:  EeTy5$}!'{b5
```

Running the script

Triangle pattern

```python

def triangle(n):

    # number of spaces
    k = 2*n - 2

    # outer loop to handle number of rows
    for i in range(0, n):

        # inner loop to handle number spaces
        # values changing acc. to requirement
        for j in range(0, k):
            print(end=" ")

        # decrementing k after each loop
        k = k - 2

        # inner loop to handle number of columns
        # values changing acc. to outer loop
        for j in range(0, i+1):

            # printing stars
            print("* ", end="")

        # ending line after each row
        print("\r")

# Driver Code
n = 5
triangle(n)
```

output

```
(base) ishaq@ishaq:~/Do
        *
      * *
    * * *
  * * * *
* * * * *
```

https://www.geeksforgeeks.org/programs-printing-pyramid-patterns-python/

THANK YOU!

Any Questions?