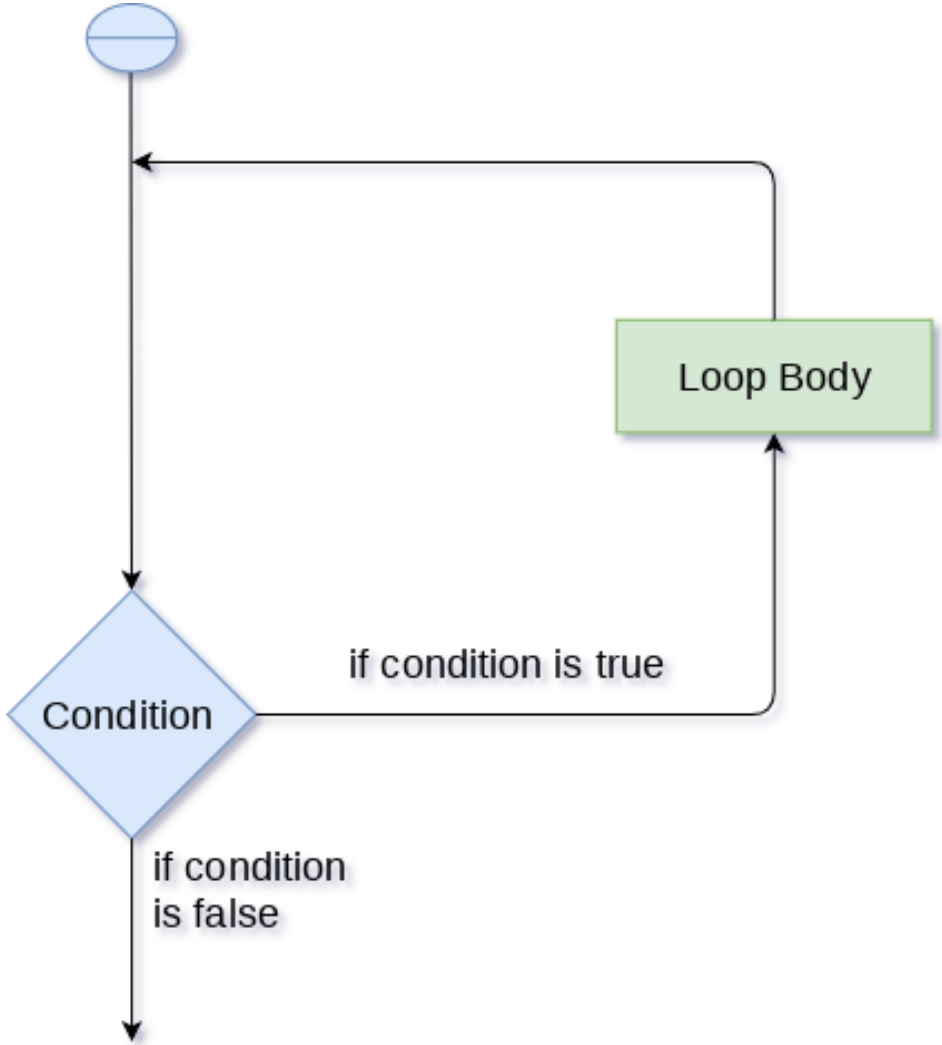
Python Loops



Advantages of loops

- It provides code re-usability.
- Using loops, we do not need to write the same code again and again.
- Using loops, we can traverse over the elements of data structures (array or linked lists).

Python programming language provides following types of loops to handle looping requirements.

- while loop
- for loop
- nested loops

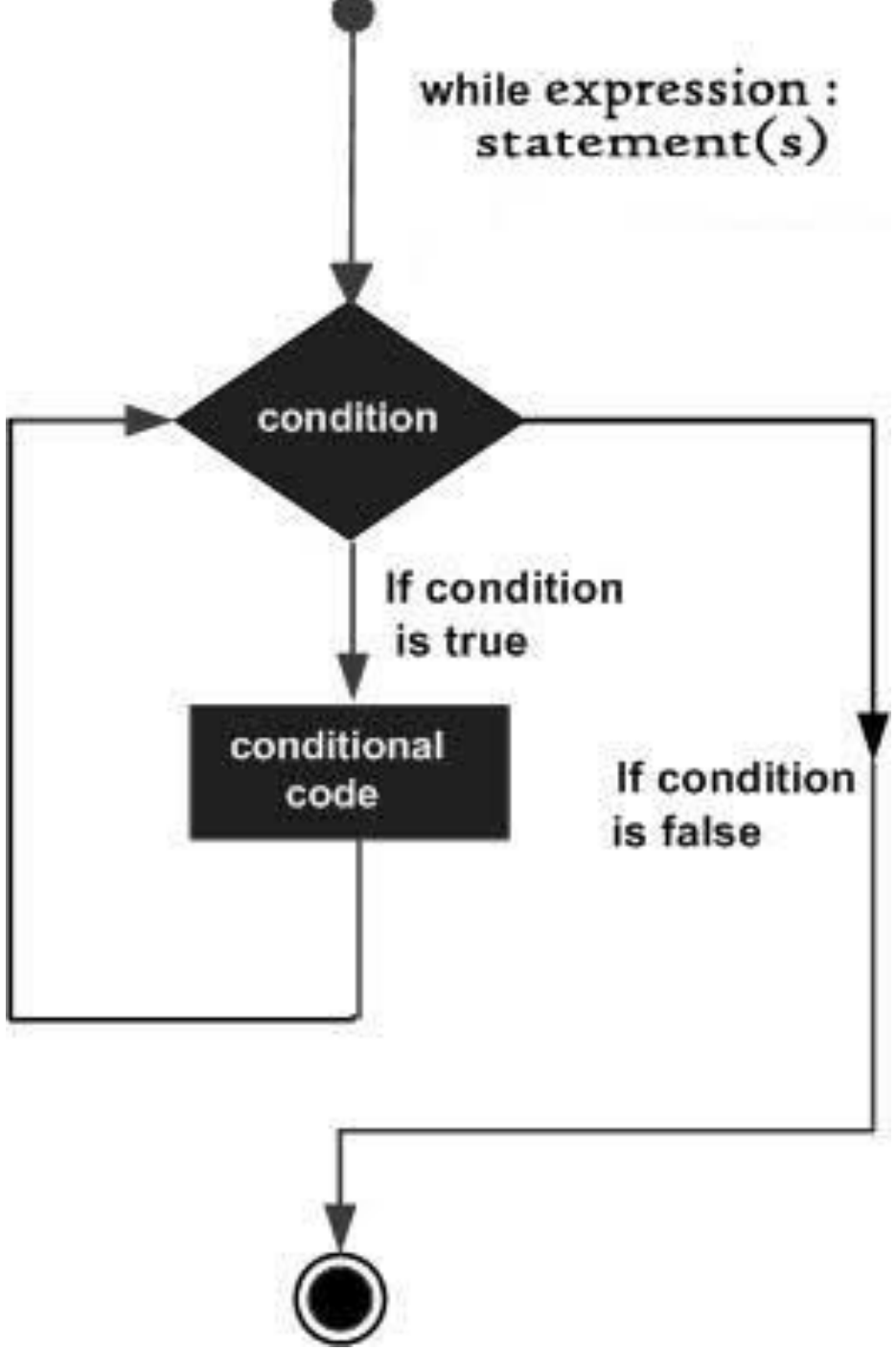
Syntax

The syntax of a while loop in

Python programming language is −

while expression:

statement(s)



The **break** Statement

With the break statement we can stop the loop even if the while condition is true:

Example

#Exit the loop when i is 3:

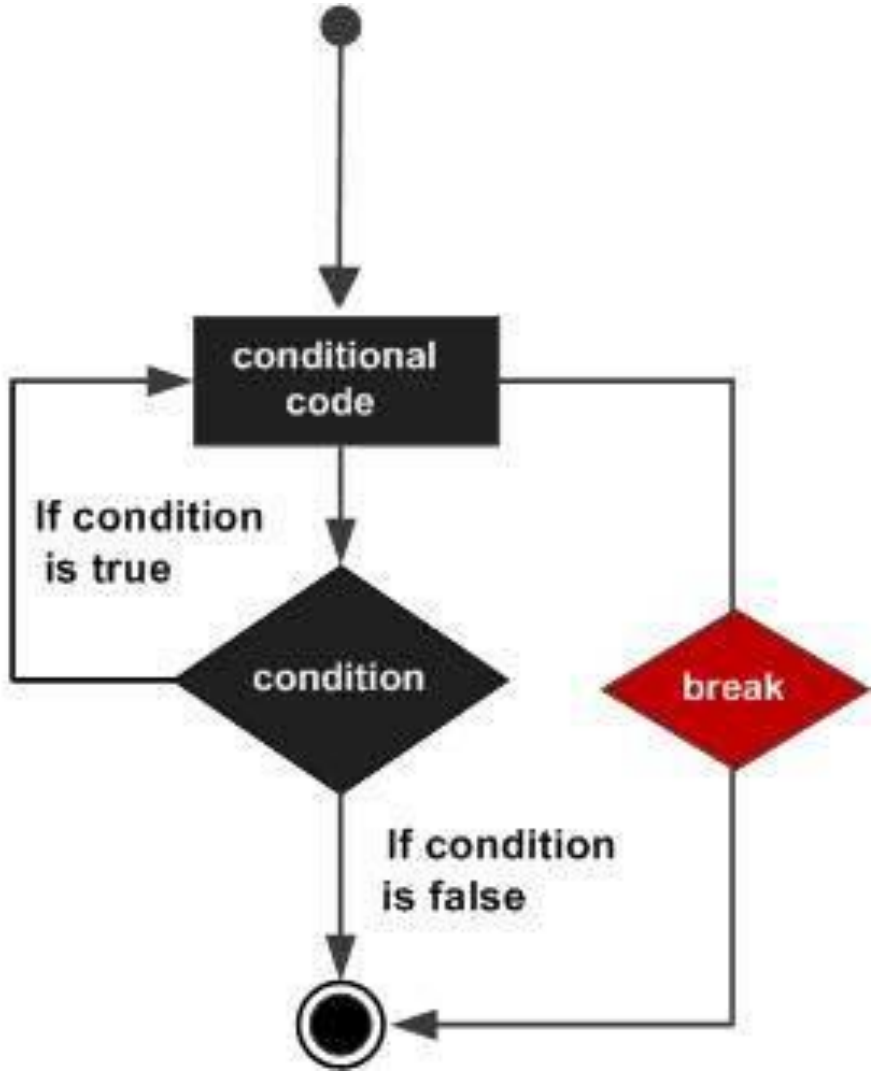
```
i=1
while(i<6):
    if(i==3):
      break
     i+=1
     print(i)

print(“hh”)
```

break statement
It terminates the current loop and resumes execution at the next statement.
The **break** statement can be used in both *while* and *for* loops.

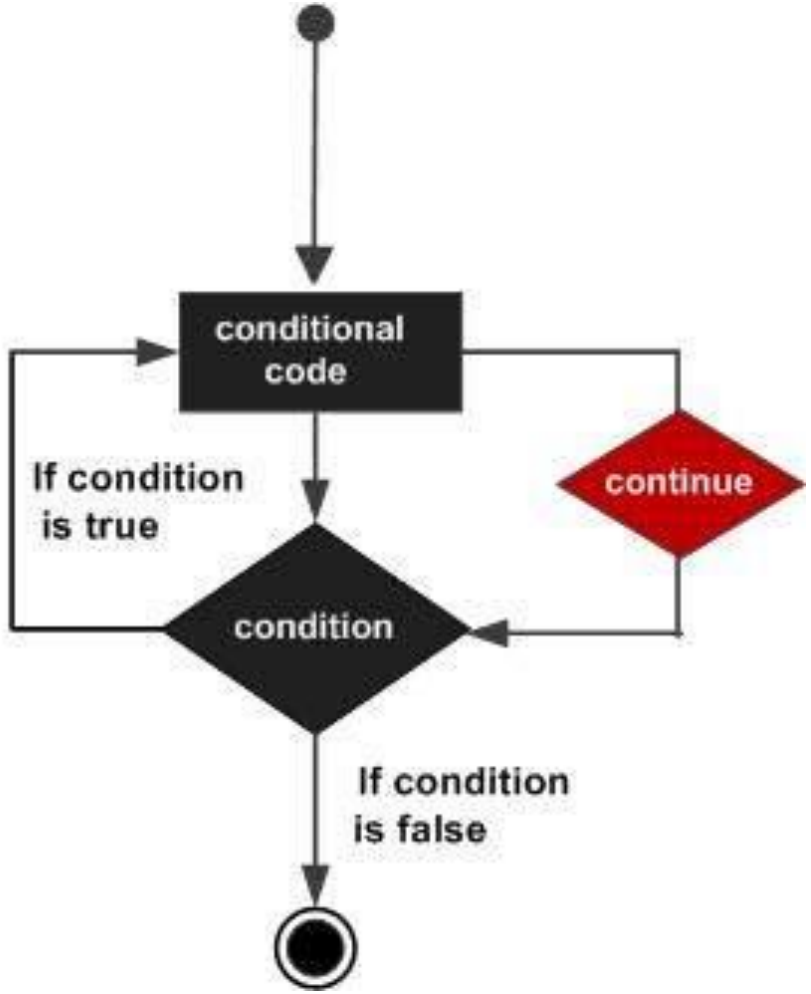
```
var = 10                    # Second Example
while var > 0:
   print ('Current variable value :', var)
   var = var -1
   if var == 5:
      break

print "Good bye!"
```



continue Statement

```
var = 10
    # Second Example
while var > 0:
   var = var -1
   if var == 5:
      continue
   print('Current variable value :', var)
print("Good bye!“)
```



- b15

The continue Statement

With the continue statement we can stop the current iteration, and continue with the next:

Example

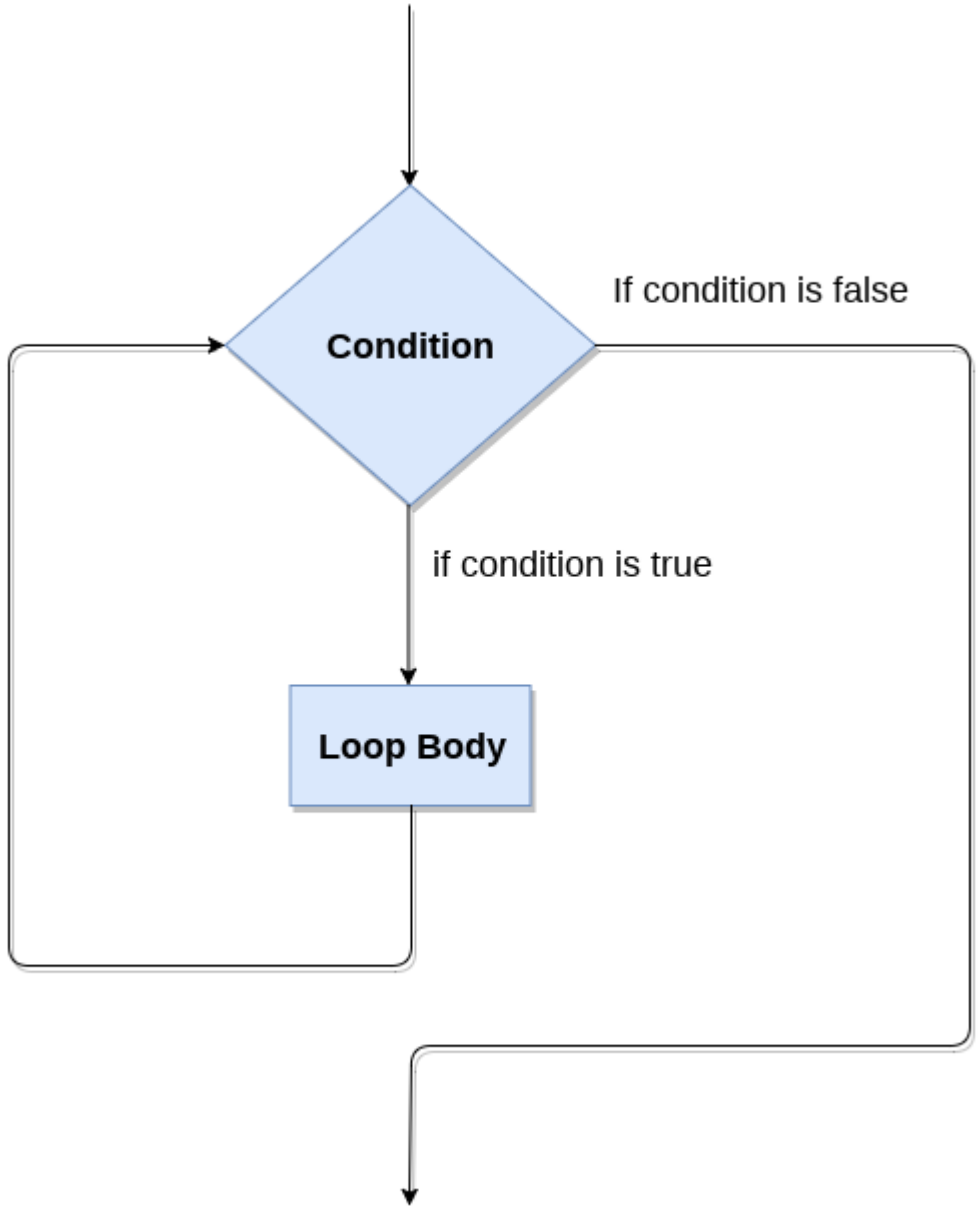
Continue to the next iteration if i is 3:

```
i=1
while(i<6):
 i+=1
 if(i==3):
  continue
 print(i)
```

Python While loop

while expression:

    statements



Infinite while loop

```
while (1):
    print(" infinite while loop")
```

For Loops

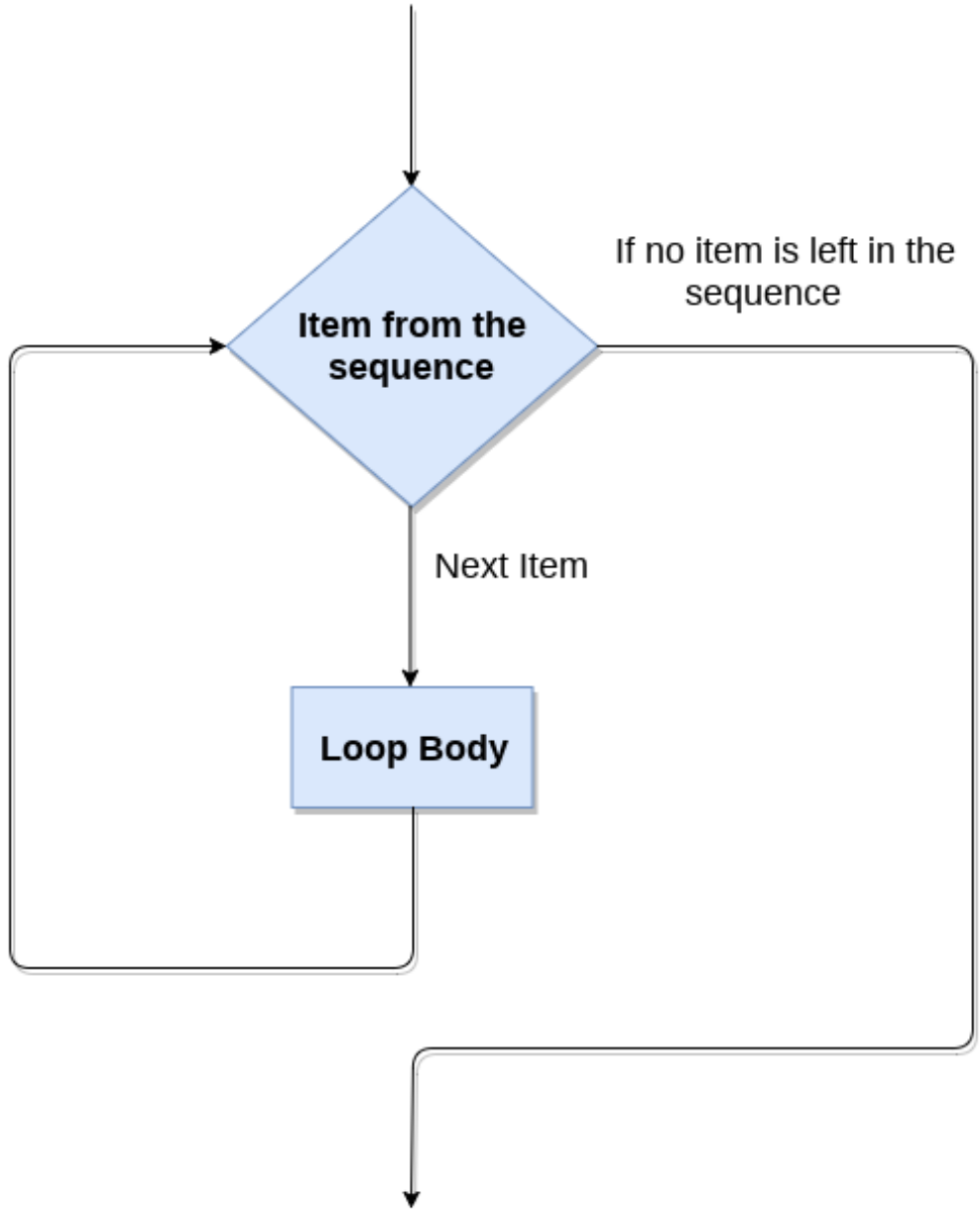
The for **loop in Python** is used to iterate the statements or a part of the program several times. It is frequently used to traverse the data structures like list, tuple, or dictionary.

The syntax of for loop in python is given below.

for iterating_var **in** sequence:

    statement(s)

```
str="Python"
for i in str:
 print(i)
```


Item from the sequence
If no item is left in the sequence
Next Item
Loop Body

The range() function

range(start,stop,step size)

- Table of number

```
n = int(input("Enter the number "))
for i in range(1,11):
    c = n*i
    print(n,"*",i,"=",c)
```

- **The range() function**
- The **range()** function is used to generate the sequence of the numbers. If we pass the range(10), it will generate the numbers from 0 to 9. The syntax of the range() function is given below.
- **Syntax:**
- range(start,stop,step size)
- The start represents the beginning of the iteration.
- The stop represents that the loop will iterate till stop-1.
  The **range(1,5)** will generate numbers 1 to 4 iterations. It is optional.
- The step size is used to skip the specific numbers from the iteration. It is optional to use. By default, the step size is 1. It is optional.

Example-3: Program to print even number using step size in range().

```
n = int(input("Enter the number "))
for i in range(2,n,2):
    print(i)
```

Output:

```
Enter the number 20
2
4
6
8
10
12
14
16
18
```

- b13

We can also use the **range()** function with sequence of numbers.
The **len()** function is combined with range() function which iterate through a sequence using indexing. Consider the following example.

```
list = ['Peter','Joseph','Ricky','Devansh']
for i in range(len(list)):
    print("Hello",list[i])
```

Output:

```
Hello Peter
Hello Joseph
Hello Ricky
Hello Devansh
```

Example-2: Program to number pyramid.

```
rows = int(input("Enter the rows"))
for i in range(0,rows+1):
    for j in range(i):
        print(i,end = '')
    print()
```

Output:

```
1
22
333
4444
55555
```

Nested for loop in python

Python allows us to nest any number of for loops inside a **for** loop. The inner loop is executed n number of times for every iteration of the outer loop

```
for iterating_var1 in sequence:  #outer loop
    for iterating_var2 in sequence:  #inner loop
        #block of statements
#Other statements
```

example

```
# User input for number of rows
rows = int(input("Enter the rows:"))
# Outer loop will print number of rows
for i in range(0,rows+1):
# Inner loop will print number of Astrisk
    for j in range(i):
        print("*",end = '')
    print()
```

Unlike other languages like C, C++, or Java, Python allows us to use the else statement with the for loop which can be executed only when all the iterations are exhausted. Here, we must notice that if the loop contains any of the break statement then the else statement will not be executed

Example 1

```
for i in range(0,5):
    print(i)
else:
    print("for loop completely exhausted, since there is no break.")
```

Example 2

```
for i in range(0,5):
    print(i)
    break;
else:print("for loop is exhausted");
print("The loop is broken due to break statement...came out of the loop")
```

- The Python while loop allows a part of the code to be executed until the given condition returns false. It is also known as a pre-tested loop.
- It can be viewed as a repeating if statement. When we don't know the number of iterations then the while loop is most effective to use.

```
for letter in 'Python':     # First Example
   if letter == 'h':
      break
   print 'Current Letter :', letter

var = 10                    # Second Example
while var > 0:
   print 'Current variable value :', var
   var = var -1
   if var == 5:
      break

print "Good bye!"
```

```
for letter in 'Python':     # First Example
   if letter == 'h':
      continue
   print 'Current Letter :', letter

var = 10                    # Second Example
while var > 0:
   var = var -1
   if var == 5:
      continue
   print 'Current variable value :', var
print "Good bye!"
```

output

- When the above code is executed, it produces the following result −
- Current Letter : P
- Current Letter : y
- Current Letter : t
- Current Letter : o
- Current Letter : n
- Current variable value : 9
- Current variable value : 8
- Current variable value : 7
- Current variable value : 6
- Current variable value : 4
- Current variable value : 3
- Current variable value : 2
- Current variable value : 1
- Current variable value : 0
- Good bye!

Homework: Python Pass

- In Python, the pass keyword is used to execute nothing; it means, when we don't want to execute code, the pass can be used to execute empty. It is the same as the name refers to. It just makes the control to pass by without executing any code. If we want to bypass any code pass statement can be used.
- It is beneficial when a statement is required syntactically, but we want we don't want to execute or execute it later. The difference between the comments and pass is that, comments are entirely ignored by the Python interpreter, where the pass statement is not ignored.