



**SNS COLLEGE OF ENGINEERING**  
Kurumbapalayam (Po), Coimbatore – 641 107

**DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING**



# **19IT103 – COMPUTATIONAL THINKING AND PYTHON PROGRAMMING**

- ❖ A readable, dynamic, pleasant, flexible, fast and powerful language

## Recap

- A Boolean expression is an expression that is evaluated as either true or false.
- Two boolean operators are and and or.
- If statement executes its body only when it is true.
- To execute alternative statements when a condition fails, if-else is useful
- If-elif-else is used to check multiple conditions
- Conditionals inside conditional is said to be nested conditional

## 3.2 Iterations

- Iterations execute a set of instructions repeatedly until some limiting criteria is met.
- Iterations are **performed through 'for' and 'while' loops.**

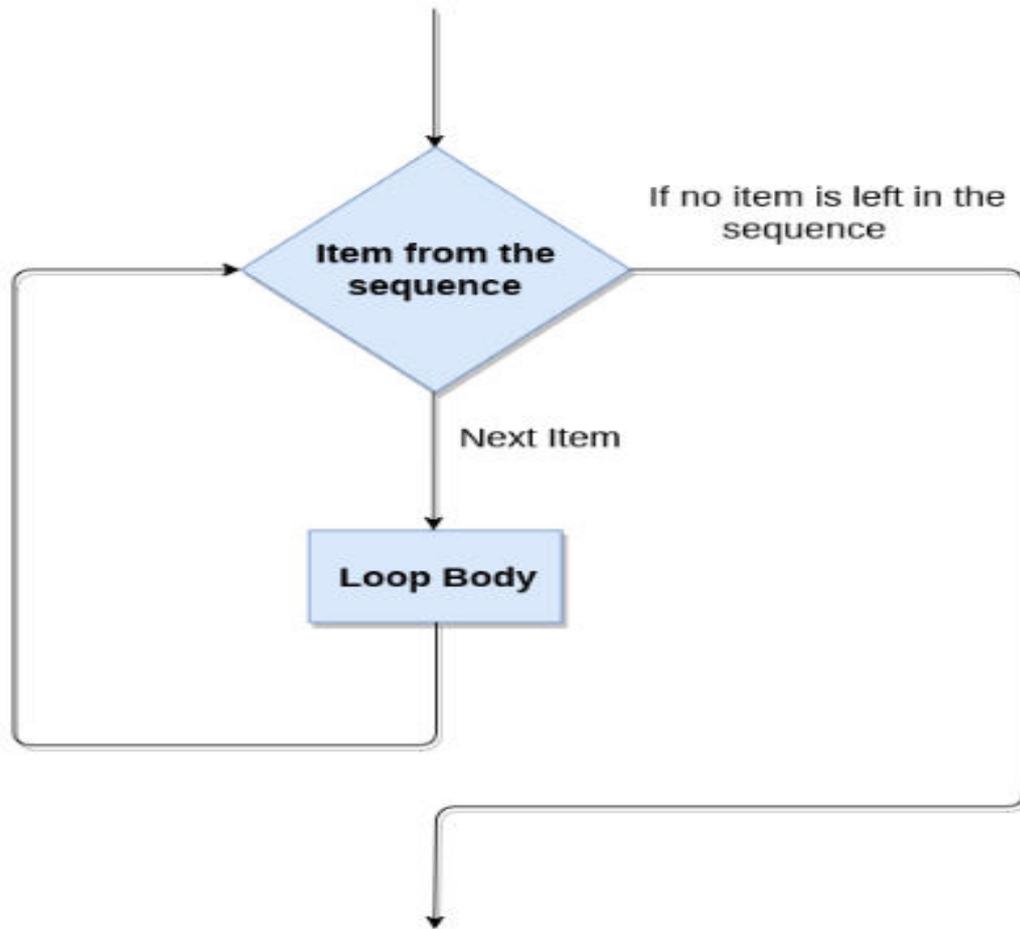
# 3.2 Iterations

## 3.2.1 'for' LOOP

- The for loop in Python is used to iterate over a sequence (list, tuple, string) or other iterable objects.
- Iterating over a sequence is called traversal.
- Iteration control variable that takes the value of the item inside the sequence on each iteration.
- Loop continues until we reach the last item in the sequence.
- The body of for loop is separated from the rest of the code using indentation.

# 3.2 Iterations

The for loop flowchart



# 3.2 Iterations

## 3.2.1 'for' LOOP

```
[*]: from time import sleep
      from random import randint

      for _ in range(0,5):
          print('Blah')
          sleep(randint(1,4))
```

Blah

---



# 3.2 Iterations

## 3.2.1 'for' LOOP

### Syntax:

```
for val in sequence:  
    Body of for
```

### Example 1:

```
#number list  
numbers = [6, 5, 3, 8, 4, 2, 5, 4, 11]  
sum = 0  
for val in numbers:  
    sum = sum+val  
print("The sum is", sum)
```

### Output:

```
The sum is 48  
>>> |
```

# 3.2 Iterations

## 3.2.1 'for' LOOP

Example 2:

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

Output:

```
P  
Y  
t  
h  
o  
n  
>>> |
```

# 3.2 Iterations

## 3.2.1 'for' LOOP using 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.

### 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.

# 3.2 Iterations

## 3.2.1 'for' LOOP using range() function

Example 3:

```
for i in range(10):  
    print(i,end = ' ')
```

Output:

```
0 1 2 3 4 5 6 7 8 9  
>>> |
```

---

# 3.2 Iterations

## 3.2.1 'for' LOOP using range() function

Example 4:

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

Output:

```
Enter the number 5
5 * 1 = 5
5 * 2 = 10
5 * 3 = 15
5 * 4 = 20
5 * 5 = 25
5 * 6 = 30
5 * 7 = 35
5 * 8 = 40
5 * 9 = 45
5 * 10 = 50
>>> |
```

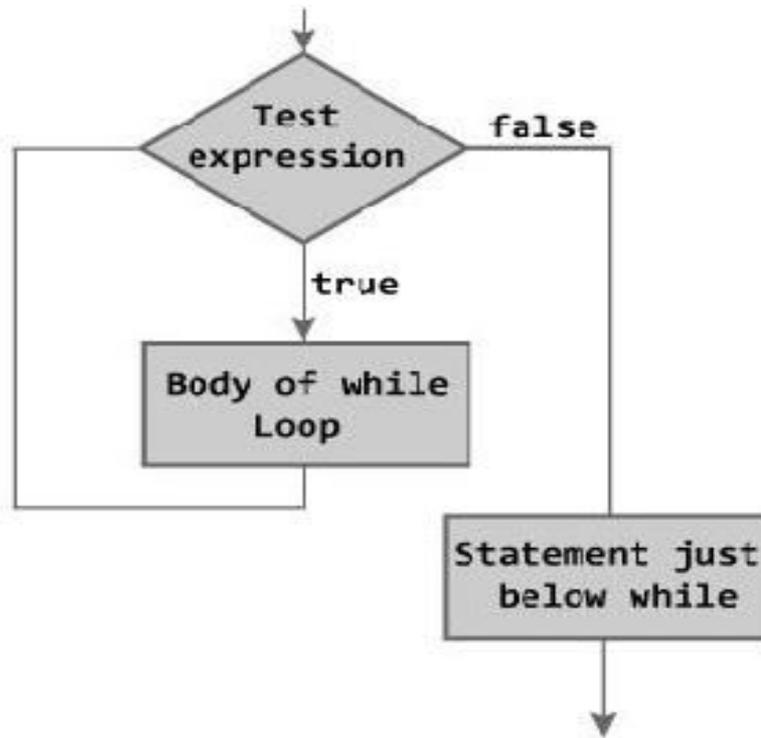
# 3.2 Iterations

## 3.2.2 'while' LOOP

- In while loop, test expression is checked first.
- The body of the loop is entered only if the test expression evaluates to True. After one iteration, the test expression is checked again. This process continues until the test\_expression evaluates to False.
- In Python, the body of the while loop is determined through indentation.
- Body starts with indentation and the first unintended line marks the end.
- Python interprets any non-zero value as True. None and 0 are interpreted as False.

# 3.2 Iterations

## 3.2.2 'while' LOOP



# 3.2 Iterations

## 3.2.2 'while' LOOP

<pre>1 a = 1 2 while a &lt; 7 : 3     if(a % 2 == 0): 4         print(a, "is even") 5     else: 6         print(a, "is odd") 7     a += 1</pre>	<i>code</i>	<i>output</i>
<i>variables</i>		
<a href="http://www.penjee.com">www.penjee.com</a>		

# 3.2 Iterations

- 3.2.2 'while' LOOP

```
1 numbers = [12, 37, 5, 42, 8, 3]
2 even = []
3 odd = []
4 while len(numbers) > 0 :
5     number = numbers.pop()
6     if(number % 2 == 0):
7         even.append(number)
8     else:
9         odd.append(number)
```

---

# 3.2 Iterations

## 3.2.2 'while' LOOP

### Syntax

```
while test_expression:  
    Body of while
```

### Example:

```
n = int(input("Enter a number: "))  
sum = 0  
i = 1  
while i <= n:  
    sum = sum + i  
    i = i+1  
print("The sum is", sum)
```

### Output:

```
Enter a number: 10  
The sum is 55  
>>> |
```

## 3.2 Iterations

### 3.2.2 'while' LOOP with else

- An optional else block with while loop can also be used.
- The else part is executed if the condition in the while loop evaluates to False.
- The while loop can be terminated with a break statement.

# 3.2 Iterations

## 3.2.2 'while' LOOP with else

**Example:**

```
counter = 0
while counter < 3:
    print("Inside loop")
    counter = counter + 1
else:
    print("Inside else")
```

**Output:**

```
Inside loop
Inside loop
Inside loop
Inside else
>>> |
```

## 3.2 Iterations

### Difference between while and for loop:

<b>while loop</b>	<b>for loop</b>
Indefinite loop	Definite loop
The exit condition will be evaluated again and execution resumes from the top(repeatedly executes a set of code)	The for is to iterate over a sequence (List, Tuple and dictionary etc)

# 3.2 Iterations

## 3.2.3 State

- State is a behavioral design pattern that allows an object to change the behavior when its internal state changes.
- The pattern extracts state-related behaviors into separate state classes and forces the original object to delegate the work to an instance of these classes, instead of acting on its own.

# Summary

- Iterative statements are used for repeated execution
- 'for' and 'while' are two looping statements used in python
- 'for loop' is definite loop whereas 'while loop' is indefinite loop
- State is the change in the behaviour of the objects

**THANK YOU**

A yellow speech bubble with a pointed tail pointing towards the bottom right. The words "THANK YOU" are cut out of the bubble in a bold, blue, sans-serif font. The background is a solid, textured blue color.