SNS COLLEGE OF ENGINEERING

Kurumbapalayam (PO), Coimbatore – 641 107
Accredited by NAAC-UGC with 'A' Grade
Approved by AICTE, Recognized by UGC & Affiliated to Anna University, Chennai

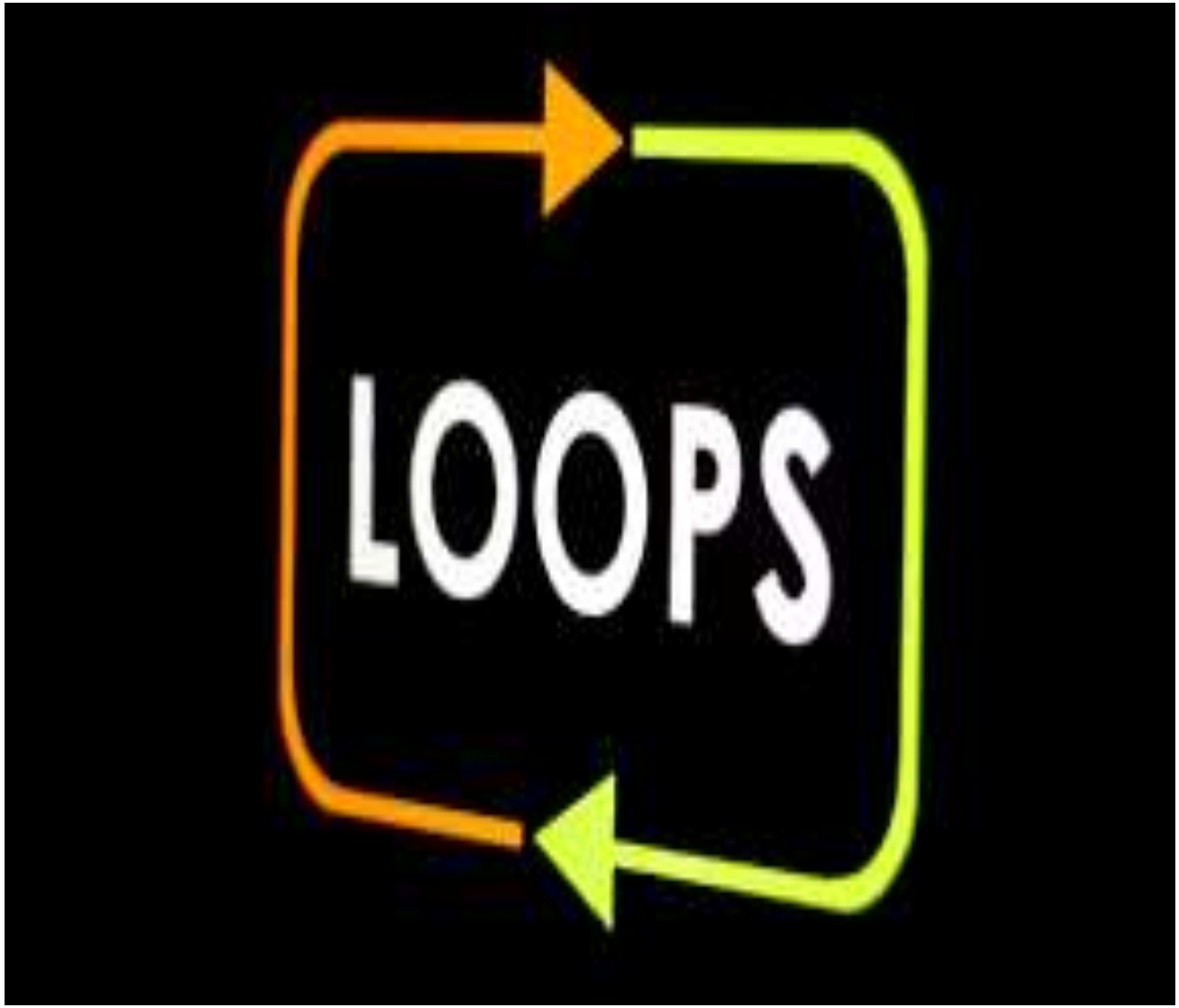
DEPARTMENT OF INFORMATION TECHNOLOGY

COURSE NAME: 23ITB202-PYTHON PROGRAMMING

II YEAR/ III SEM

Unit : Introduction

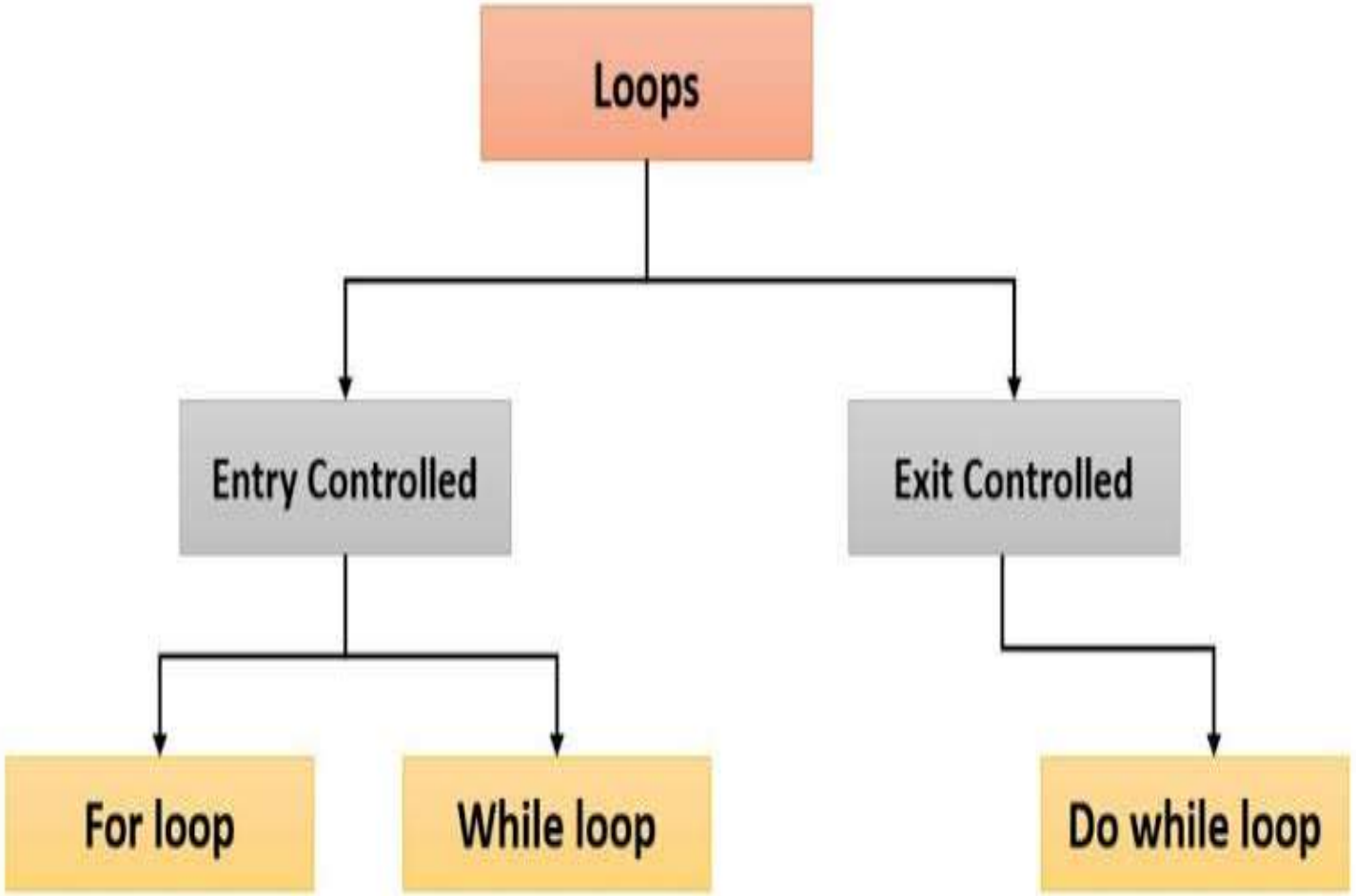
Topic : Basics of Python


LOOPS

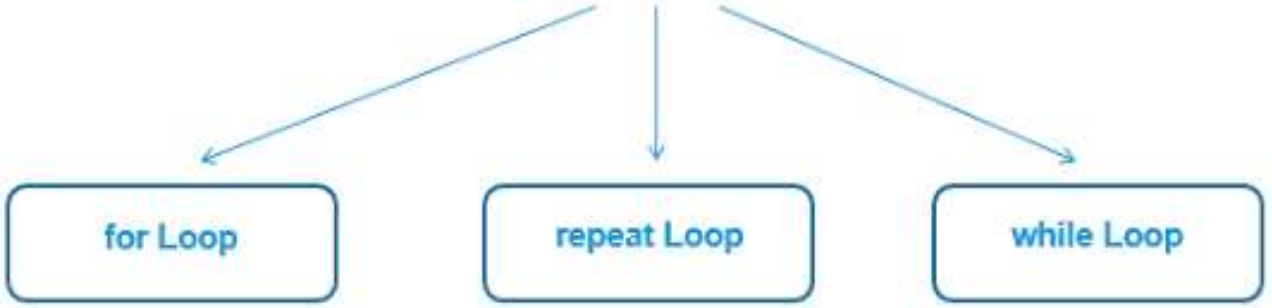
Looping statement

JAVA Types of Loops

- There are three types of loops:
  - `while` Loops
  - `for` Loops
  - `do` Loops
- Each loop requires the following steps:
  - Initialization (get ready to start looping)
  - Condition (test if we should execute loop body)
  - Update (change something each time through)


Loops
Entry Controlled
Exit Controlled
For loop
While loop
Do while loop

Types of Loops



- Iterates over the elements of any sequence (vector) **till the condition defined is true**
- **Number of iterations** are fixed and known in advance

- **Infinite loop** and used with **break statement** to exit the loop
- **Number of iterations** depends on the condition which is checked at the end of each iteration

- Repeats a statement or group of statements **until some condition is met**
- **Number of iterations** depends on the condition which is checked at the beginning of each iteration

While loop

- while loop is used to execute a block of statements repeatedly until a given condition is satisfied. And when the condition becomes false, the line immediately after the loop in the program is executed.

Syntax

Syntax :

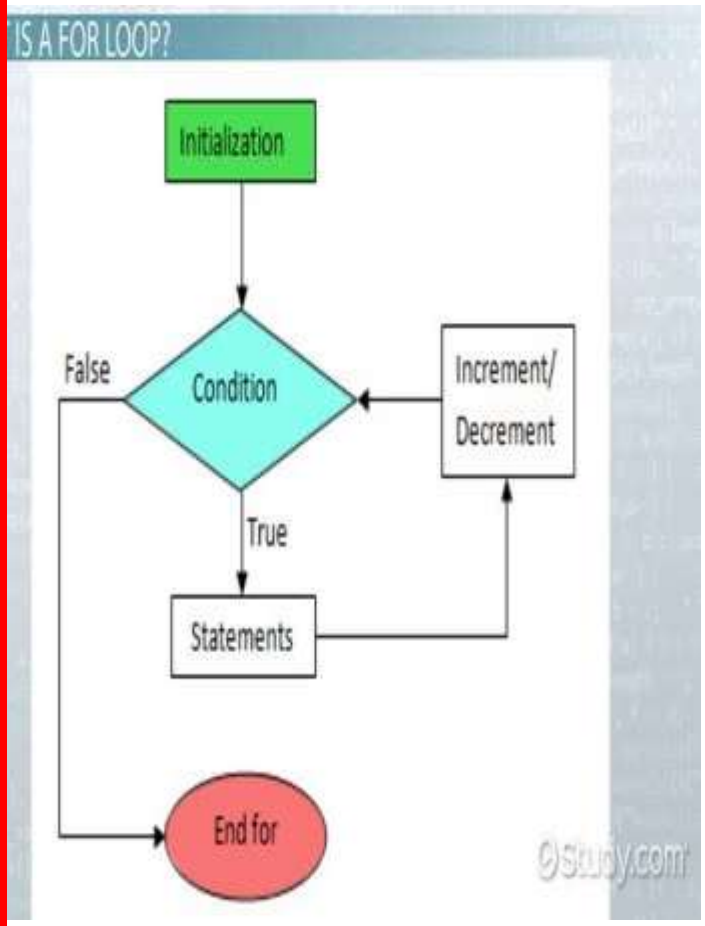
while expression:

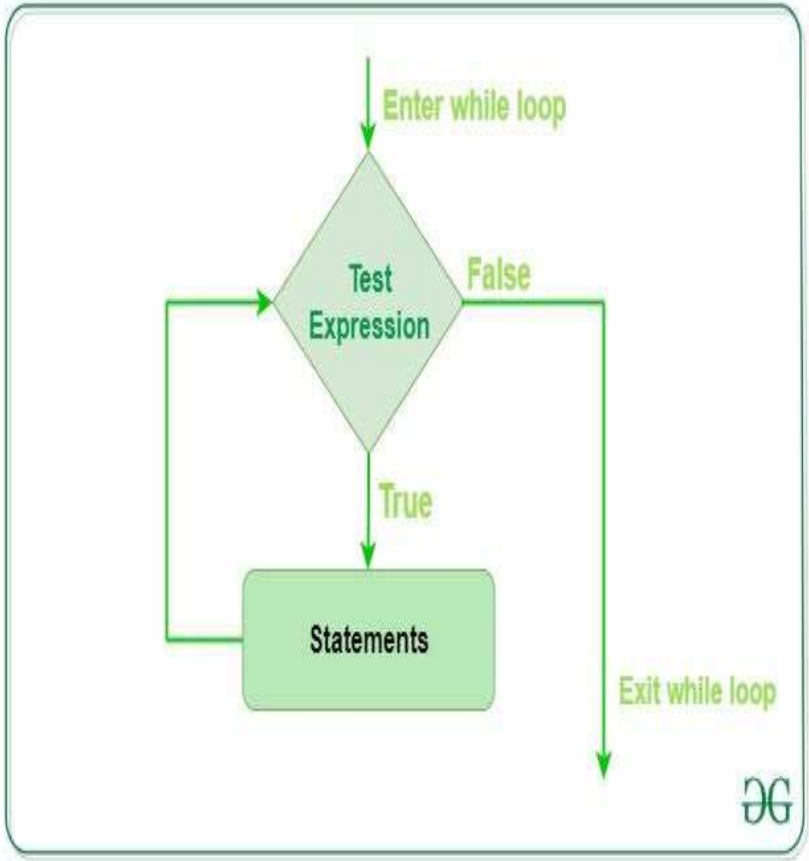
    statement(s)

All the statements indented by the same number of character spaces after a programming construct are considered to be part of a single block of code. Python uses indentation as its method of grouping statements.

- The statements of the Python while loop are dictated by indentation.
- The code block begins when a statement is indented & ends with the very first unindented statement

Flow chart


IS A FOR LOOP?
Initialization
False
Condition
Increment/
Decrement
True
Statements
End for


Enter while loop
Test
Expression
False
True
Statements
Exit while loop

Example

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

```
1
2
3
4
5
>
```

```
The Multiplication Table of:  12
12 x 1 = 12
12 x 2 = 24
12 x 3 = 36
12 x 4 = 48
12 x 5 = 60
12 x 6 = 72
12 x 7 = 84
12 x 8 = 96
12 x 9 = 108
12 x 10 = 120
> 
```

Multiplication Table using While Loop

```
num =12
counter = 1
# we will use a while loop for iterating 10 times for the multiplication table
print("The Multiplication Table of: ", num)
while counter <= 10: # specifying the condition
    ans = num * counter
    print (num, 'x', counter, '=', ans)
    counter += 1 # expression to increment the counter
```

While Loop Multiple Conditions

```
num1 = 17
num2 = -12

while num1 > 5 and num2 < -5 : # multiple conditions in a single while loop
    num1 -= 2
    num2 += 3
    print( (num1, num2) )
```

```
(15, -9)
(13, -6)
(11, -3)
>
```

While Loops in Python

Learning Objectives:
Remember the following Programming Skills:
- FOR Loops
- WHILE Loops

'While Loops' in Python

The **x** is simply a variable. It could have any name.

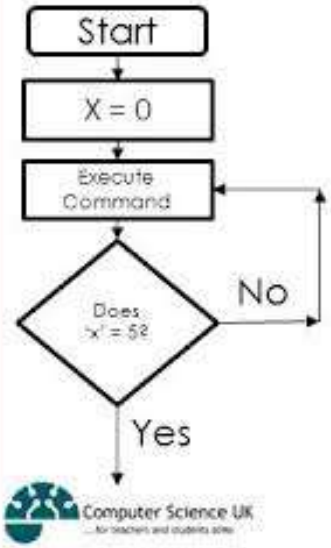
It is however a special kind of variable known as the '***most recent value***'

We must finish the statement with a colon

```
while x == n:
       != 
       >
       <
```

The **n** is represents a value that we want x to either equal, not equal, be greater than, etc. depending on the **condition** we want to use.

E.g. **n=5** and the condition **while x != 5** *(not equal to 5)* then the loop would repeat until x equals 5.



www.computerscienceuk.com

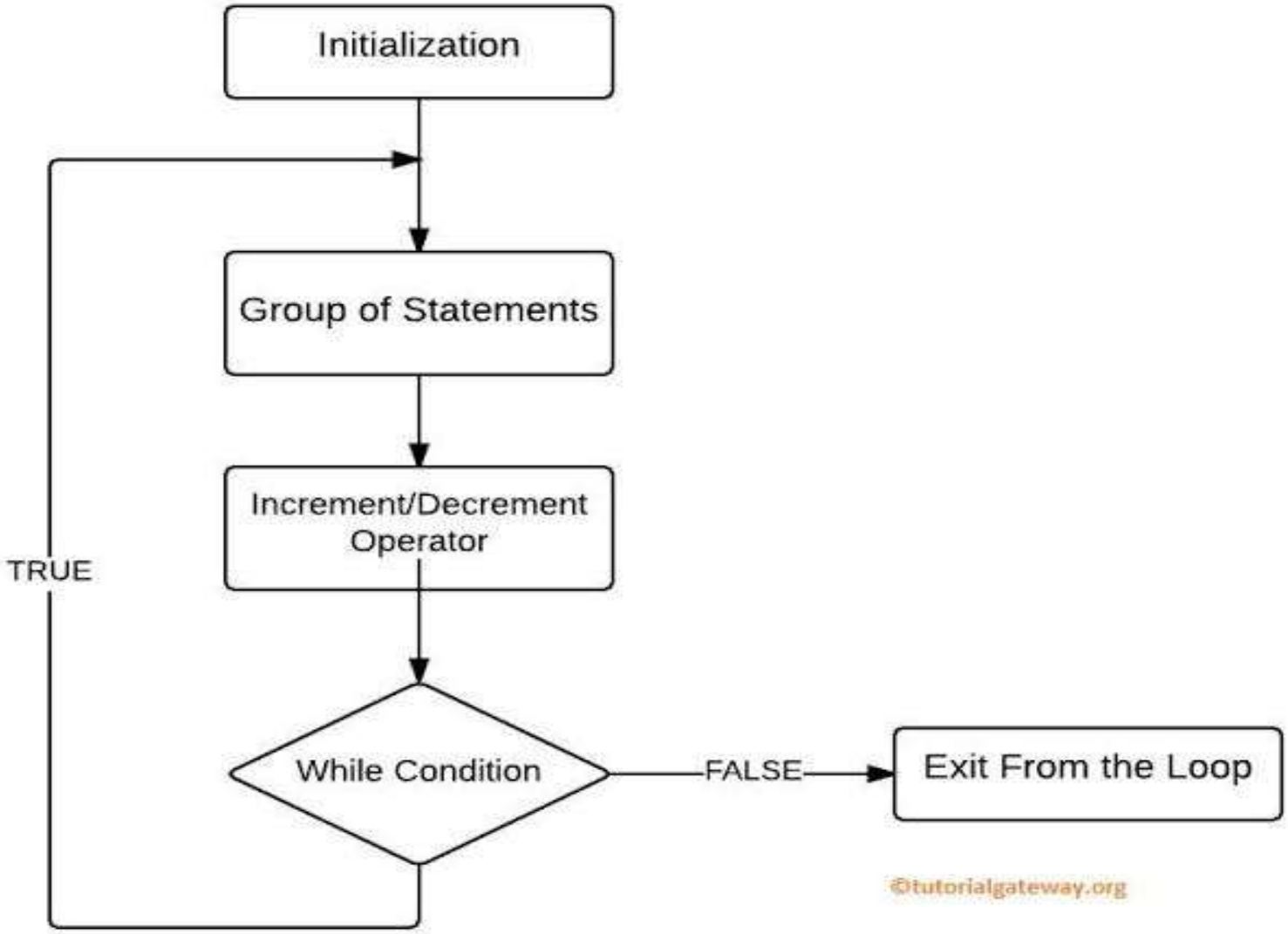
Do while loop

- The do while construct consists of a process symbol and a condition. First the code within the block is executed. Then the condition is evaluated.
- If the condition is true the code within the block is executed again. This repeats until the condition becomes false.

Syntax

```
do {
  loop block statement to be executed;
  }
while(condition);
```

Flowchart


Initialization
Group of Statements
Increment/Decrement Operator
TRUE
While Condition
FALSE
Exit From the Loop
©tutorialgateway.org

Do while

- The while loop in python first checks for condition, and then the block is executed if the condition is true. The block is executed repeatedly until the condition is evaluated to false.
- Thus, in python, we can use a while loop with if/break/continue statements that are indented, but if we use do-while, it does not fit the indentation rule.
- Therefore we cannot use the do-while loop in python.

Example

```
i = 1
while True:
    print(i)
    i = i + 1
    if(i > 5):
        break
```

```
1
2
3
4
5
>
```

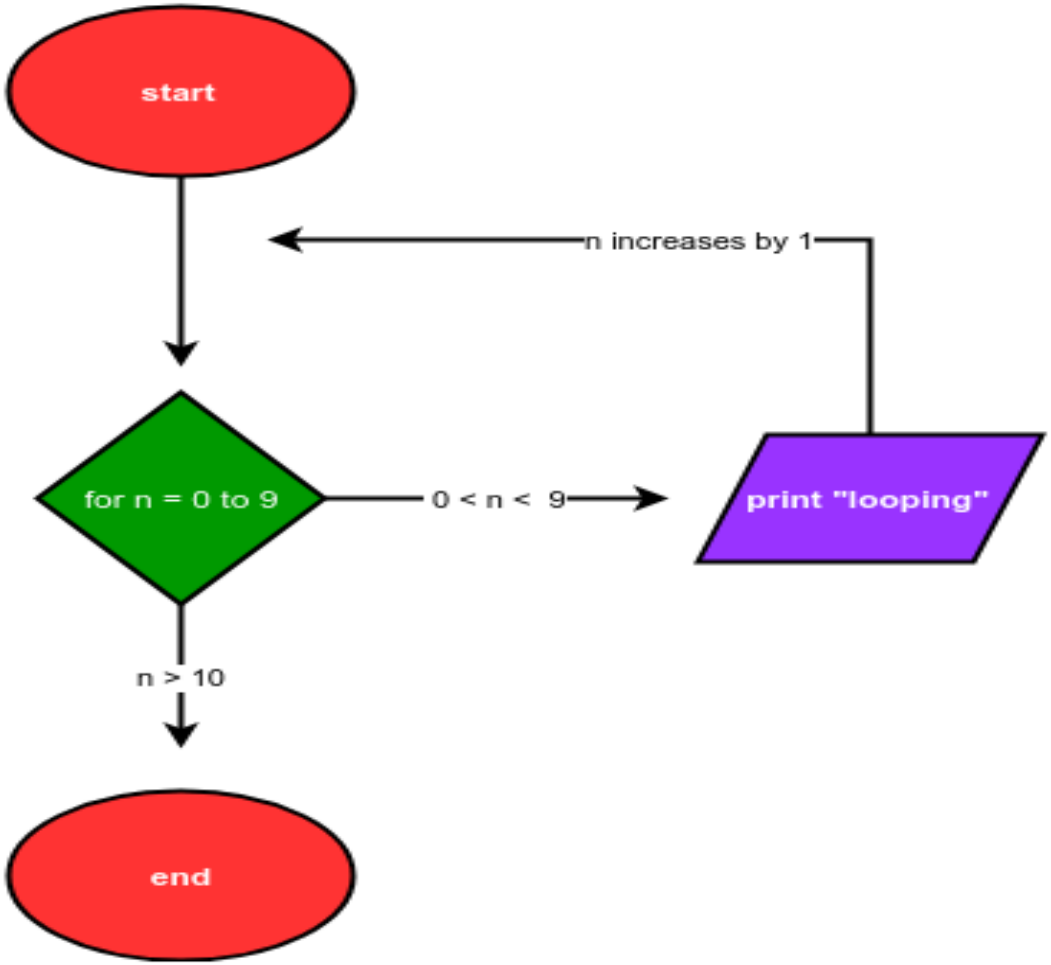
For loop

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

Syntax

```
for val in sequence:
    # statement(s)
```

Flow chart



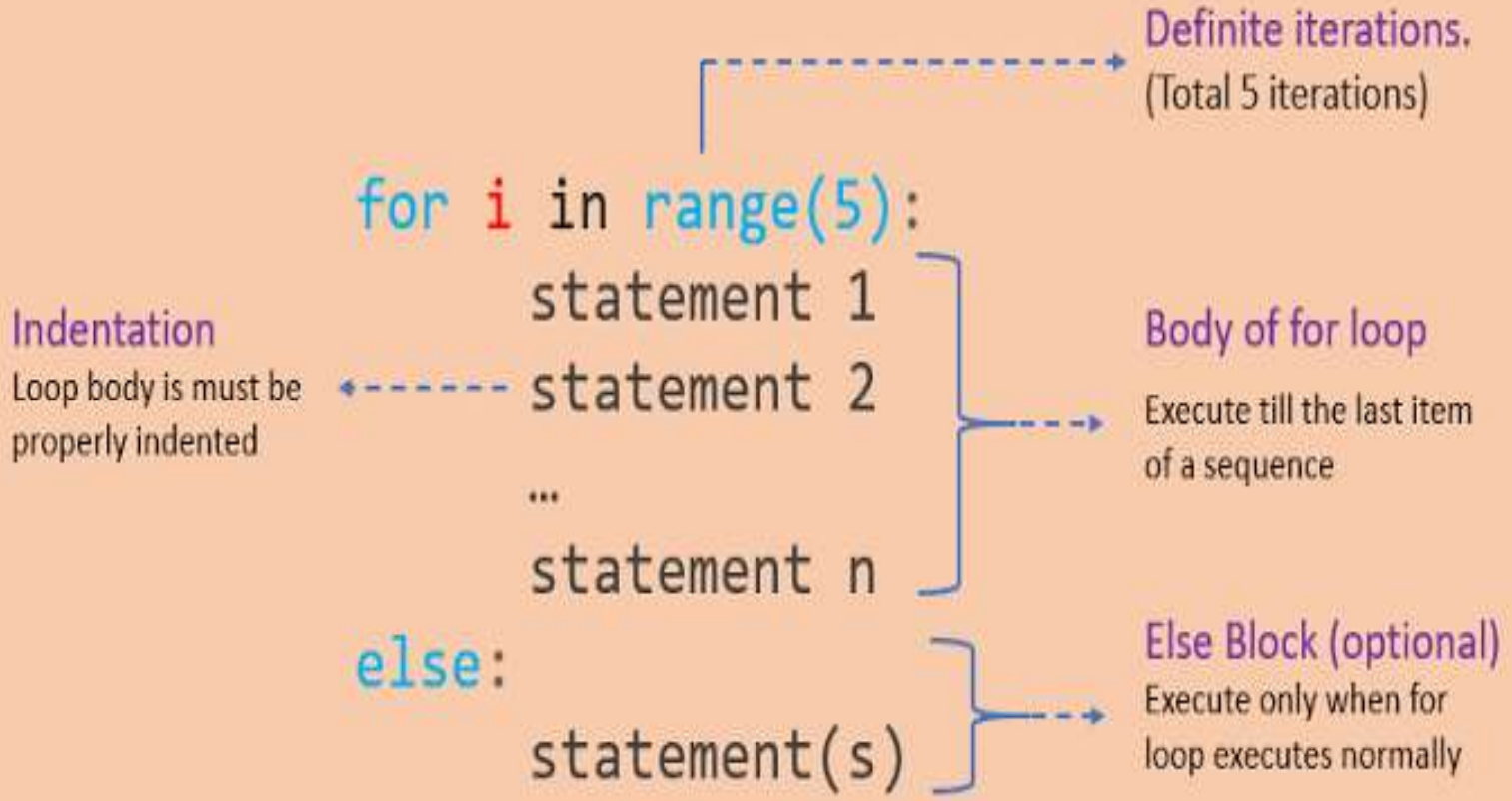
Example

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

```
apple
banana
cherry
>
```

Python for loop

A for loop is **used for iterating over a sequence and iterables** (like range, list, a tuple, a dictionary, a set, or a string).



Looping Through a String

```
for x in "banana":
  print(x)
```

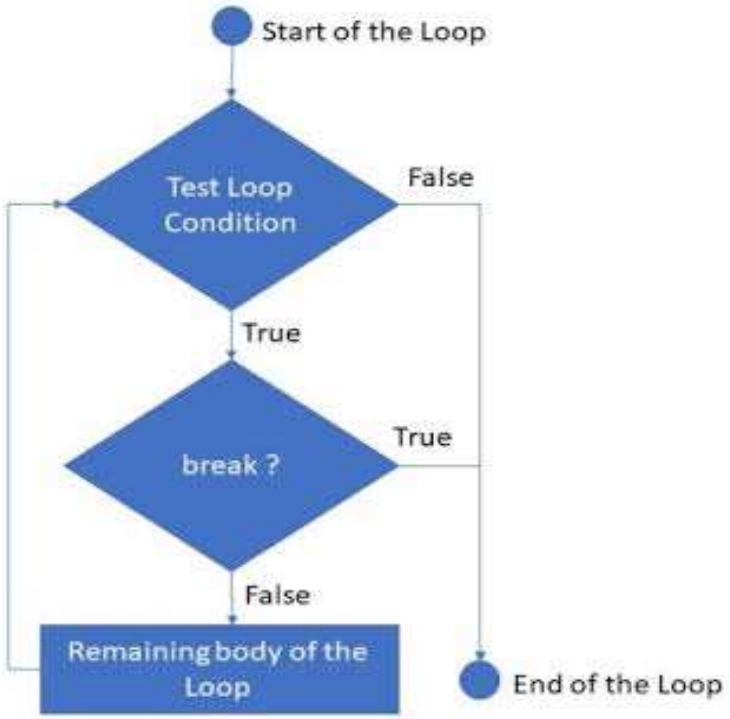
```
b
a
n
a
n
a
```

The break Statement

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

```
apple
banana
>
```

Flow chart


Start of the Loop
Test Loop Condition
False
True
break ?
True
False
Remaining body of the Loop
End of the Loop

Exit the loop when x is "banana", but this time the break comes before the print:

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

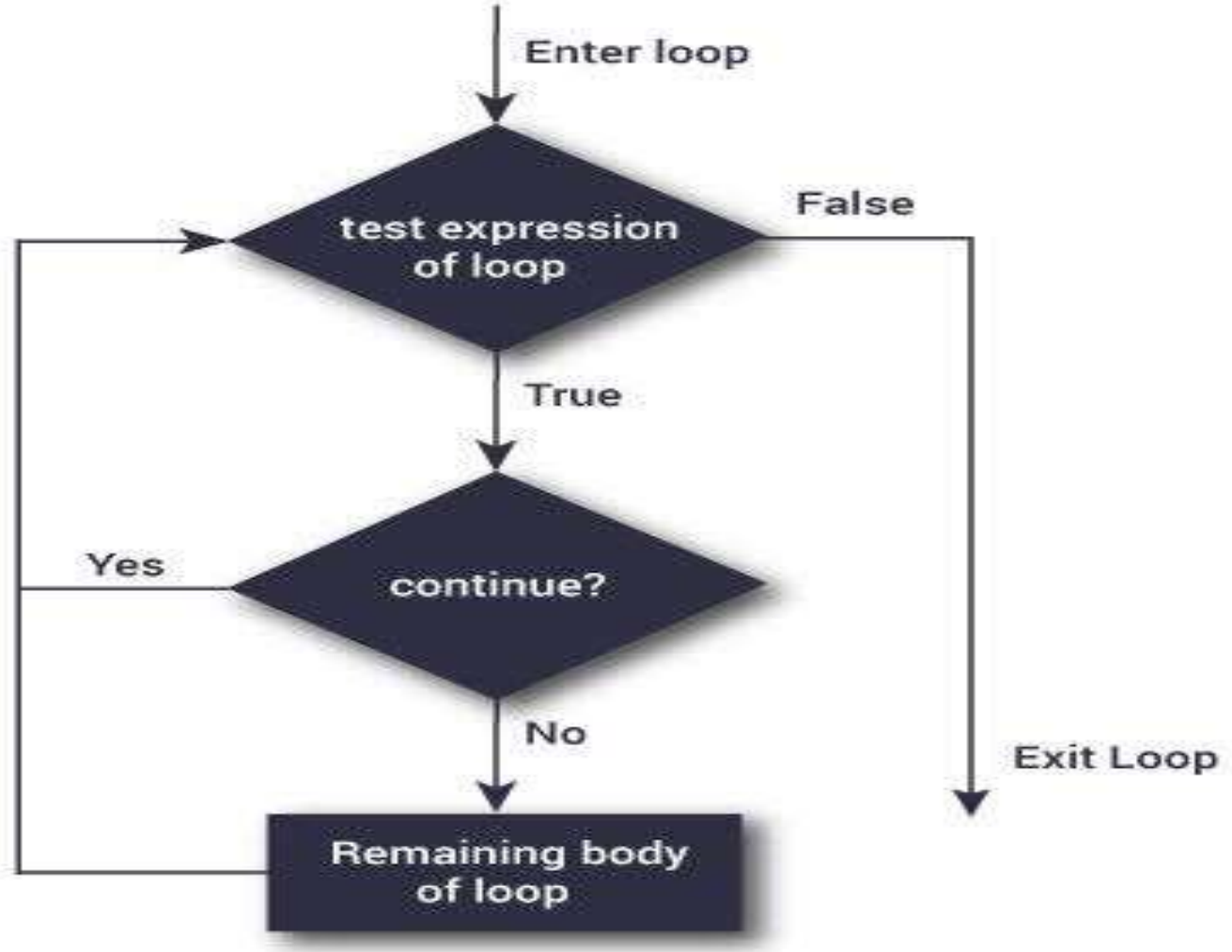
```
apple
>
```

The continue Statement

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

```
apple
cherry
>
```

Flow chart


Enter loop
test expression of loop
False
True
Yes
continue?
No
Exit Loop
Remaining body of loop

The range() Function

The range() function returns a sequence of numbers, starting from 0 by default, and increments by 1 (by default), and ends at a specified number.

```
for x in range(6):
  print(x)
```

```
0
1
2
3
4
5
>
```

Using the start parameter:

```
for x in range(2, 6):
  print(x)
```

```
2
3
4
5
>
```

ncrement the sequence with 3 (default is 1):

```
for x in range(2, 30, 3):
  print(x)
```

```
2
5
8
11
14
17
20
23
26
29
> |
```

Else in For Loop

- The else keyword in a for loop specifies a block of code to be executed when the loop is finished:

```
for x in range(6):
  print(x)
else:
  print("Finally finished!")
```

If the loop breaks, the else block is not executed.

```
for x in range(6):
  if x == 3: break
  print(x)
else:
  print("Finally finished!")
```

```
0
1
2
>
```

Pass statement

- When the pass statement is executed, nothing happens, but you avoid getting an error when empty code is not allowed.
- Empty code is not allowed in loops, function definitions, class definitions, or in if statements.

```
for x in [0, 1, 2]:
  pass
```

- # having an empty for loop like this, would raise an error without the pass statement

Nested Loop in Python

A nested loop is a loop inside the body of the outer loop. The inner or outer loop can be any type, such as a while loop or for loop.

The outer loop can contain more than one inner loop. There is no limitation on the chaining of loops

In the nested loop, the number of iterations will be equal to the number of iterations in the outer loop multiplied by the iterations in the inner loop.

In each iteration of the outer loop inner loop execute all its iteration. For each iteration of an outer loop the inner loop re-start and completes its execution before the outer loop can continue to its next iteration.

Syntax

```
# outer for loop
for element in sequence
   # inner for loop
   for element in sequence:
      body of inner for loop
   body of outer for loop
```

Flow chart

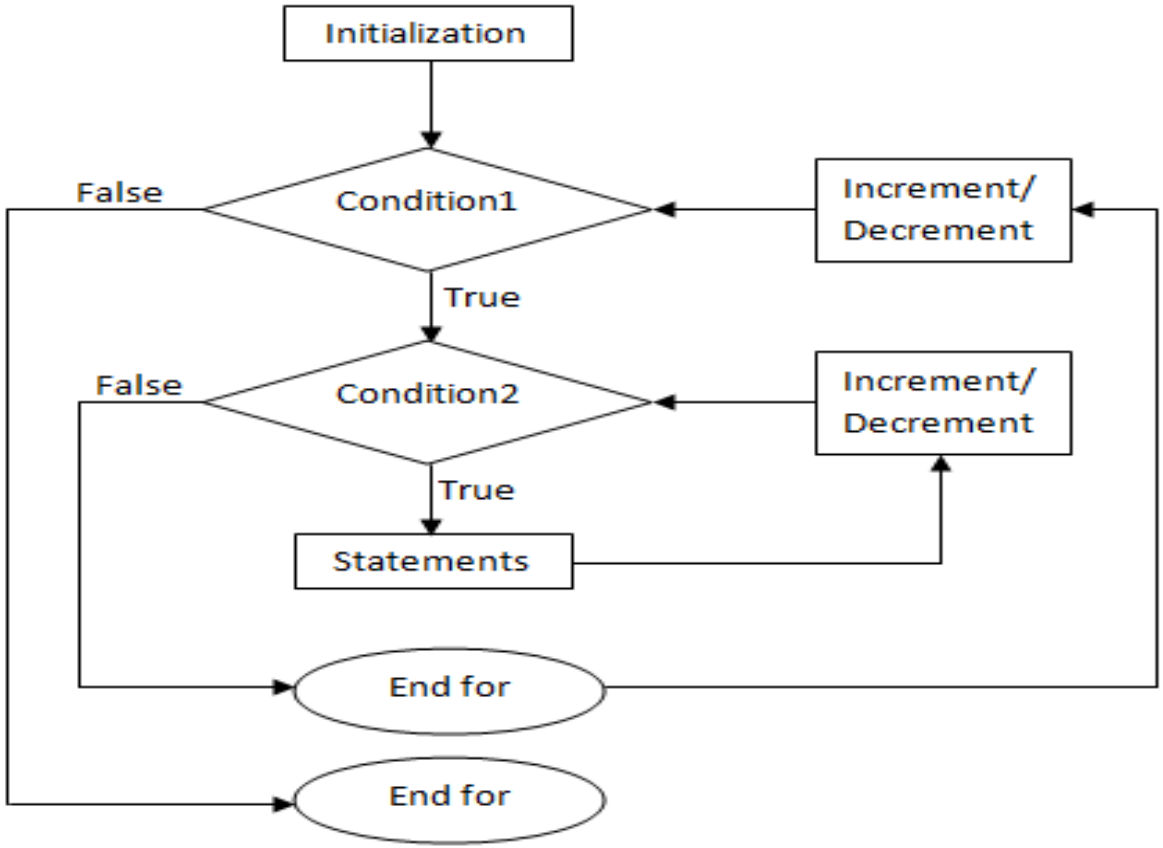


Fig: Flowchart for nested for loop

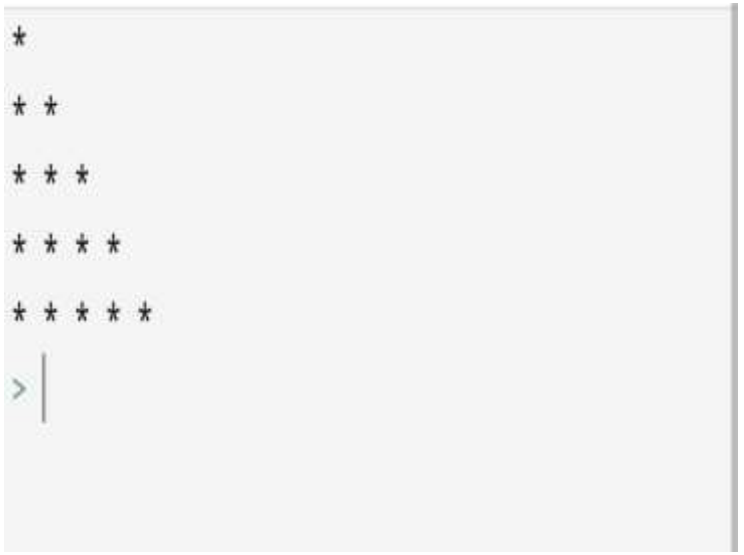
Example

```
# outer loop
for i in range(1, 11):
    # nested loop
    # to iterate from 1 to 10
    for j in range(1, 11):
        # print multiplication
        print(i * j, end=' ')
    print()
```

```
1 2 3 4 5 6 7 8 9 10
2 4 6 8 10 12 14 16 18 20
3 6 9 12 15 18 21 24 27 30
4 8 12 16 20 24 28 32 36 40
5 10 15 20 25 30 35 40 45 50
6 12 18 24 30 36 42 48 54 60
7 14 21 28 35 42 49 56 63 70
8 16 24 32 40 48 56 64 72 80
9 18 27 36 45 54 63 72 81 90
10 20 30 40 50 60 70 80 90 100
>
```

Example

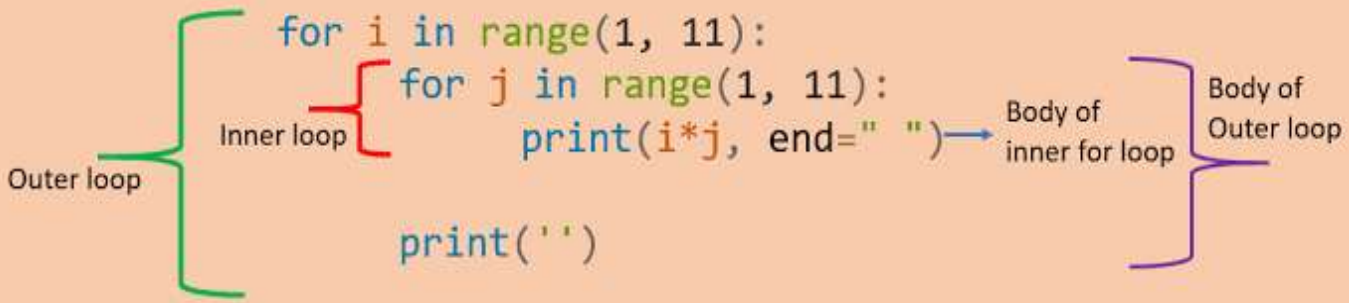
```
rows = 5
# outer loop
for i in range(1, rows + 1):
    # inner loop
    for j in range(1, i + 1):
        print("*", end=" ")
    print('')
```



Python Nested Loop

A Loop inside a loop is known as a nested loop.

In the nested loop, the number of iterations will be equal to the number of iterations in the outer loop multiplied by the iterations in the inner loop.



PYnative.com

Thank you