

Loops

A loop is a programming structure that allows you to execute a block of code repeatedly as long as a condition is true or for each item in a sequence. We can say in other words Loops are very useful when the number of instructions needs to execute more than one time.

There are 2 types of loops in Python Language

1. **For** loop
2. **While** loop

For example

```
print("Hello")
print("Hello")
print("Hello")
print("Hello")
print("Hello")
```

1. **for** Loop

for loop is used to execute a set of statements repeatedly until **for – loops** through a block of code a specified number of times when decide in range. We can say it an open-ended loop. It is usually preferable when the number of range iterations are known as **for** loop.

Syntax:

For variable **in** **range**(Star, Stop , Step)
Statement(s);

Here:

```
range(start, stop, step)
```

means:

- **start** = where to begin
- **stop** = where to stop, excluding this value
- **step** = how much to increase each time

Flowchart

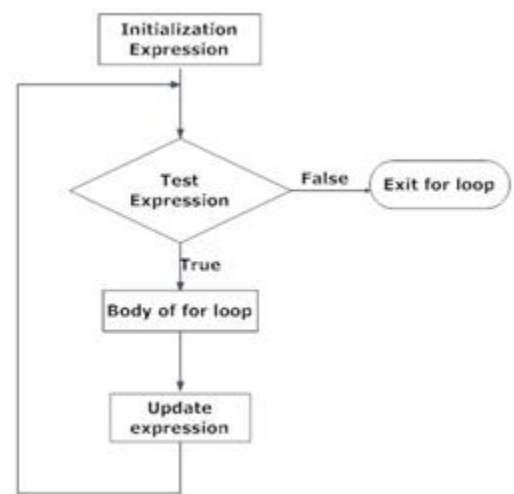


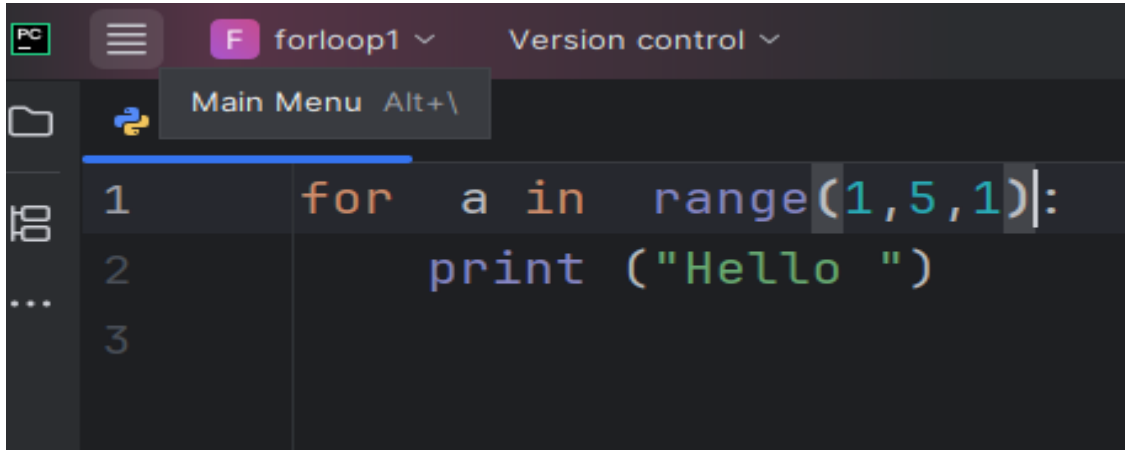
Figure: Flowchart of for loop

Explanation:

The start number is initialized only once at the beginning in range of the 'for' loop. Then, the stop number is checked by the program. If the stop number is true, then 'for' loop is terminated and increment step one by one automatically.

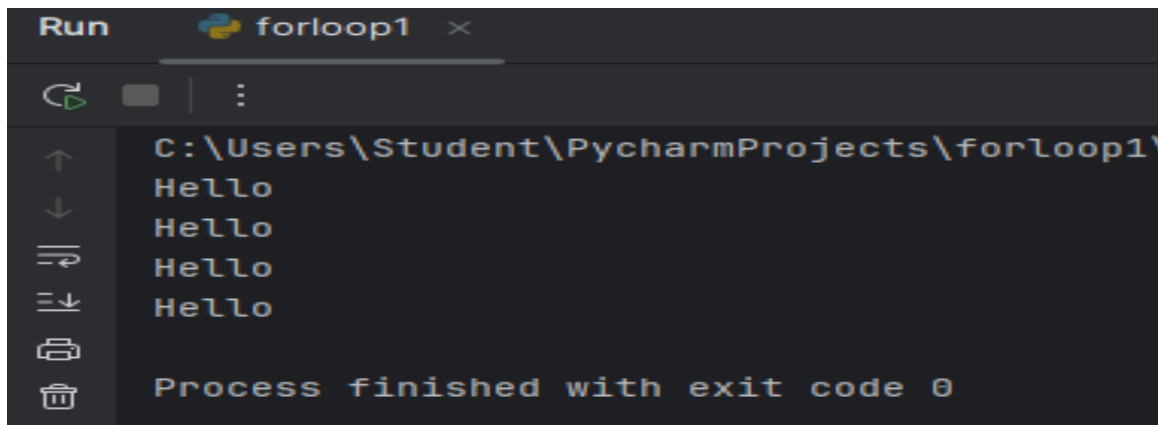
Example-1

```
for a in range(1,5,1)    // if a is less than 5
    print ("Hello ")    // then it will print value of a, then update value 1 by 1
```



```
1 for a in range(1,5,1):
2     print ("Hello ")
3
```

This prints "Hello" four times.



```
Run forloop1 x
C:\Users\Student\PycharmProjects\forloop1\
Hello
Hello
Hello
Hello
Process finished with exit code 0
```

Example-2

```
print ("Odd Series")
for a in range(1,10,2):
    print(a)
```

Output

```
1
3
5
7
9
```

Example-3	Output
<code>print ("Reverse Order linear Series")</code>	1
<code>for a in range(10,0,-1):</code>	3
<code> print(a)</code>	5
	7
	9

Example-4	Output
<code>tno= print ("Enter any No for Table : ")</code>	Enter any No for Table : 3
<code>for no in range(1,10,1):</code>	3 x 1 = 3
<code> result = tno * no</code>	3 x 2 = 6
<code> print(tno,"x",no, "=",result)</code>	3 x 3 = 9
	3 x 4 = 12
	:
	3 x 10 30

Example-5	Output
<code>print (" No. Square Cube ")</code>	No. Square Cube
<code>print (" ===== ")</code>	1 1 1
<code>for no in range(1,10,1):</code>	2 4 8
<code> sq = no ** 2</code>	3 9 27
<code> cub = no ** 3</code>	4 16 64
<code> print(no," ",sq, " ",cub)</code>	:
	9 81 729

Note: A for loop is used when you want to **iterate over a sequence or collection**, such as a list, tuple, string, or dictionary.

Example-6 fruits = ["Apple", "Banana", "Mango"] for fruit in fruits: print(fruit)	Output Apple Banana mango
--	--

Here:

- fruits is a list.
- fruit is the loop variable.
- The loop takes each item from fruits one at a time.
- The print() statement executes for every item.

Example-7 text = "THIS IS PYTHON CLASS" for character in text: print(character)	Output T H I S I S P Y T H O N C L A S S
--	---

2. while loop

while loop is a control flow statement that allows code to be executed repeatedly based on a given Boolean condition. The condition is evaluated, and if the condition is true, the code within the block is executed first. This repeats until the condition becomes false. Because **while** loop checks the condition before the block is executed, it is also known as a **pre-test loop**.

Syntax:

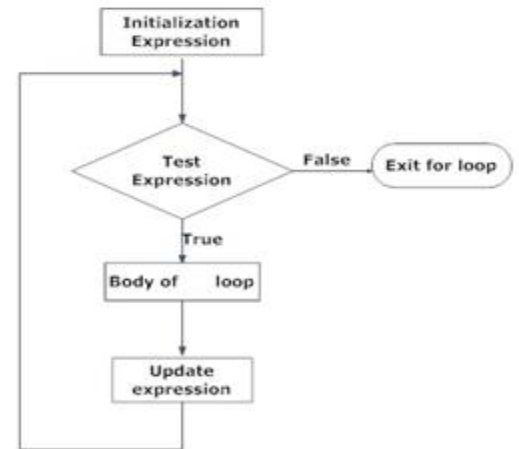
Initialization expression

while Test Expression

Statement(s);

Update Expression

Flow Chart



Explanation:

first checks whether A is TRUE, which it is, so then the {loop body} is entered, where it executes statement(s) that are in the body of loop, after execution of every statement once it checks the condition again if the condition remains TRUE it will execute the body again. Process repeating until the A remains true.

Example-1:

```

1 a=1
2 while a<=5:      # if a is less than and equals to 5
3     print(a)      # then it will print value of a, then update value 1 by 1
4     a=a + 1
5
Run forloop1 x
C:\Users\Student\PycharmProjects\forloop1\.venv\Scripts\python.exe C:\Users\Student\PycharmProjects\forloop1\forloop1.py
1
2
3
4
5
Process finished with exit code 0
    
```

©Copy Right

Example-2 <pre>print ("Reverse Order linear Series") b=10 while b >= 0: print(b) b = b - 1</pre>	Output 10 9 8 7 6 5 4 3 2 1
Example-3 <pre>print ("Even Series") b=0 while b <= 10: b = b + 2 print(b)</pre>	Output Even Series 2 4 6 8 10 12

for vs while Loop

for loop

Used to iterate over a sequence or range

Often used when the number of iterations is known

Automatically moves to the next item

Example: processing every item in a list

while loop

Used while a condition remains true

Often used when the number of iterations is not known beforehand

You generally need to update the condition variable

Example: repeatedly asking for input until it is valid

➤ Nested Loop

if loop uses within a loop then this terminology is called **"Nested loop"**

Syntax:

```
for variable in range:           # Outer Loop
                                # Outer loop start here
for variable in range:           # Inner Loop
                                # Inner loop start here
    statement;                   # execute this statement
```

The screenshot shows a PyCharm IDE window titled 'NestedLoop1'. The editor displays a Python script named 'nestedloop1.py' with the following code:

```
1 print("Nested Loop1")
2 for a in range(1,5):
3     for b in range(1,4):
4         print (b)
5     print ("---")
6
```

Below the editor, the 'Run' tab is active, showing the execution path and output. The output is as follows:

```
C:\Users\Student\PycharmProjects\NestedLoop1\.venv\Scripts\python.exe
1
2
3
---
1
2
3
---
1
2
3
---
1
2
3
---
Process finished with exit code 0
```

Example	Output
<pre>print(" Nested Loop example-2") for a in range(1,5): for b in range(1,4): print(b,end="") print("")</pre>	Nested Loop Example-2 123 123 123 123

Noted: Python also provides loop-control statements such as **break**, **continue**, and **pass**.

The **Continue** statement/keyword.

Sometimes a situation arises where we want to take the control to the beginning of the loop (**for** example **for** and **while** loop.) skipping the rest statements inside the loop which have not yet been executed. The keywords **continue** allow us to **do** this. When the keywords **continue** executed inside a loop the control automatically passes to the beginning of loop. Continue is usually associated with the **if**.

Example	Output
<pre>a=1 print(" Series 1 to 10") while a <= 10: if a== 3: a += 1 continue else: print(a) a += 1</pre>	Series 1 to 10 1 2 4 5 6 7 8 9 10

©Copy Right

Example	Series 1 to 10
<pre>print(" Series 1 to 10") for a in range(1,10): if a== 3: continue else: print(a)</pre>	1 2 4 5 6 7 8 9

The **break** Statement/keyword

break is used to immediately terminate a loop.

Example	Series 1 to 9
<pre>print(" Series 1 to 9") for a in range(1,10): if a== 3: # if condition is true then break # then programming control break from in the loop else: print(a)</pre>	1 2

pass Statement/keyword

pass does nothing. It is used as a placeholder when Python requires a statement but you don't want to execute anything yet.

Example	Series 1 to 9
<pre>print(" Series 1 to 9") for a in range(1,10): pass</pre>	

Exercise

Theory Questions

1. Write down the syntax of Nested loops.
2. Difference between break and continue statement or keywords.
3. Describe list of for VS while loop?
3. Which type of looping structure is used to iterate through elements of string?

Practical Questions

1. Write a program to print table of a number entered by user using **while** loop.
2. Write a program to print following output using nested **for** loop

```
1
12
123
1234
12345
```

3. Write down the code to display series from 0 to 5 and their sum numbers, using loop of your choice.

Count	Total
0	0
1	1
2	3
3	6
4	10
5	15

4. Write down the code to display following output

1	2	3	4	5
2	4	6	8	10
3	6	9	12	15
4	8	12	16	20
5	10	15	20	25

4. Write a program series of prime number from 1 to 25.

Objective MCQ's:

- Each repetition of a looping statement is called a(n) _____.
 - Recurrence
 - Iteration
 - Duplication
 - Re-execution
- Which of the following is the correct syntax for a **for** loop statement.
 - for**(i=0, i<=10 , i++)
 print("display from a for statement ")
 - for**(i=0; i<=10; i++)
 print ("display from a for statement ")
 - for** i **as** 10;
 print ("display from a for statement ")
 - for** a **in** **range**(1,10)
 print ("display from a for statement ")
- Counter variable _____.
 - Can only be increment.
 - Can only be decrement
 - Can be incremented or decremented
 - Do not change
- Which of the following is the correct syntax for **while** statement?
 - while** (i <10 , i++)
 print (i)
 - while** i in range (i ,10)
 print (i)
 - while** i <10:
 print (i)
 - while** (i <10)
 print (i)
- These keywords are used in the loops, if depend on requirement.
 - continue
 - break;
 - a and b, both
 - case