Reading: Loops - for,while

Loop, as the name says, is something that happens repetitively. Python has two types of loops. The loops use the conditional operators to repeat a set of action on meeting a particular condition or until a particular condition is met. We will take a look at each of loops in Python.

A. for Loop

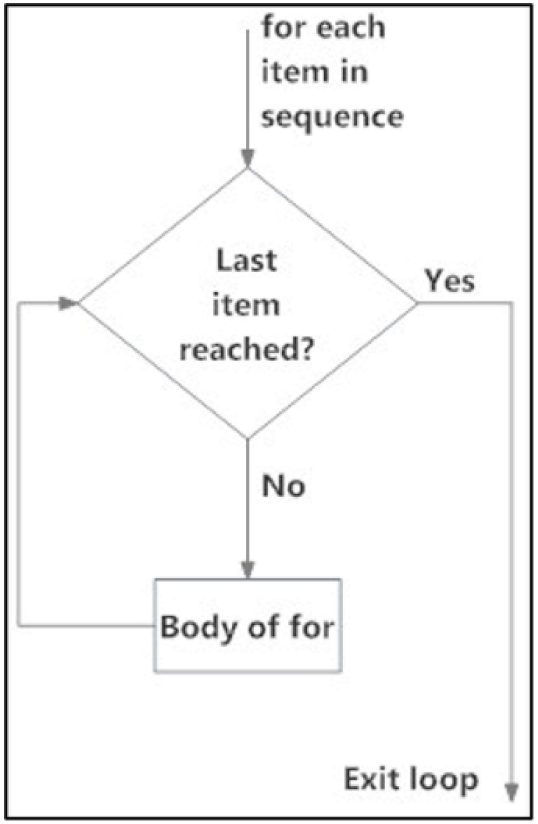
The *for loop* in Python is used to iterate over a sequence (list, tuple, string) or other iterable objects. Iterating over a sequence is called **traversal**.

The syntax is:

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

Here, val is the 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.

The Flowchart of *for loop* is:



Consider the following example:

```
# Program to find the sum of all numbers stored in a list
sample_list = [1,2,3,4,5,6,7]
sum = 0
# iterate over the list
for val in sample_list:
    sum = sum+val
print("The sum is", sum)
```

```
The sum is 28
```

The range() function

We can generate a sequence of numbers using *range()* function. **range(7)** will generate numbers from **0 to 6** (7 numbers).

We can also define the **start**, **stop** and **step size** as **range(start, stop,step_size).**

step_size defaults to 1 if not provided.

Since this function does not store all the values in memory; it would be inefficient. So, it remembers the start, stop, step size and generates the next number on the go. To force this function to output all the items, we can use the function list().

```
print(range(7))
```

```
range(0, 7)
```

```
print(list(range(7)))
```

```
[0, 1, 2, 3, 4, 5, 6]
```

We can use the **range()** function in for loops to iterate through a sequence of numbers. It can be combined with the **len()** function to iterate through a sequence using indexing.

```
fruits = ['apple', 'mango', 'cherry']

for i in range(len(fruits)):
    print("I like", fruits[i])
```

```
I like apple
I like mango
I like cherry
```

B. while Loop

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

This loop is used when we don't know the number of times to iterate beforehand.

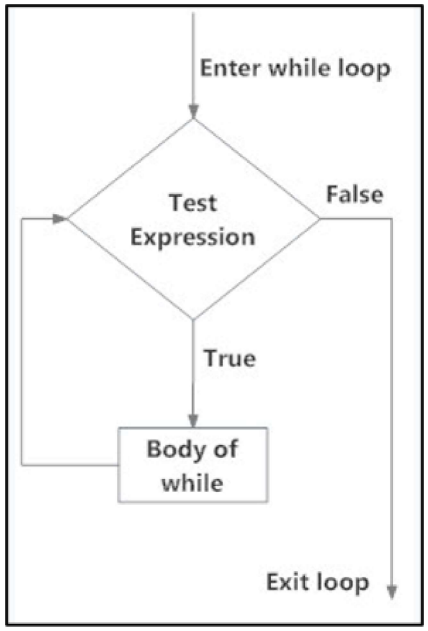
The syntax is:

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

In the 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. The body starts with indentation and the first un-indented line marks the end.

The Flowchart of *while loop* is:



Consider the following example:

```
n = 10
sum = 0
i = 1

while i <= n:
    sum = sum + i
    i = i+1    # update counter

# print the sum
print("The sum is", sum)
```

```
The sum is 55
```

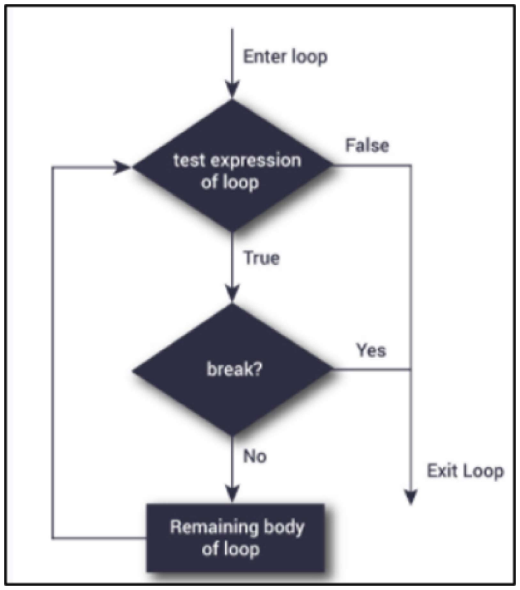
Here, the test expression will be True as long as our counter variable i is less than or equal to n n=10 here).
We must increase the value of the counter variable in the body of the loop. This is very important as failing to do so will result in an infinite loop.
Finally, the result is displayed.

C. break Statement

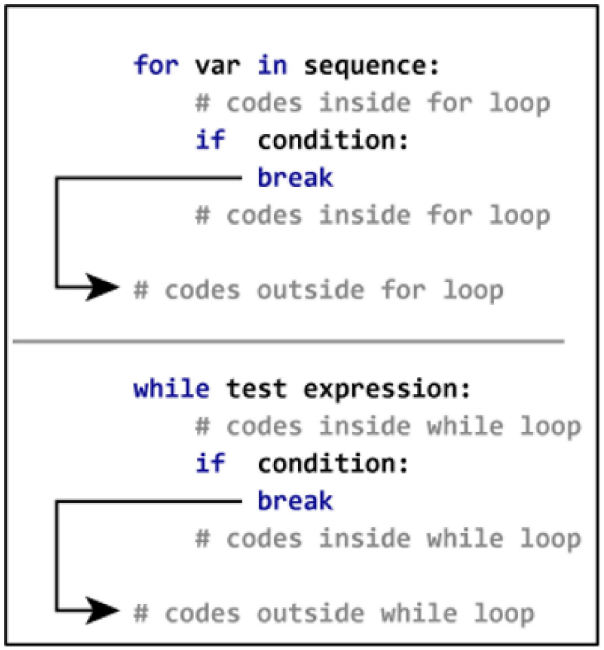
The **break** statement terminates the loop containing it. Control of the program flows to the statement immediately after the body of the loop.

If the break statement is inside a nested loop, it will terminate the innermost loop.

The Flowchart of *break* Statement is:



The working of break statement in for loop and while loop is shown below:



Consider the following example:

```
for val in "sample_string":
    if val == "i":
        break
    print(val)

print("The end")
```

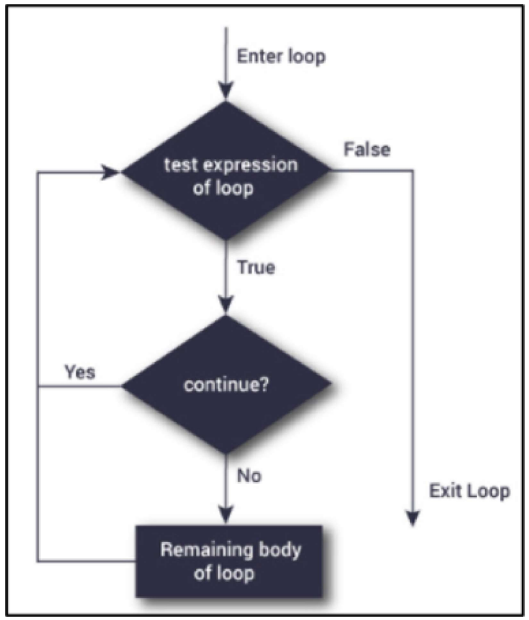
```
s
a
m
p
l
e

_
s
t
r
The end
```

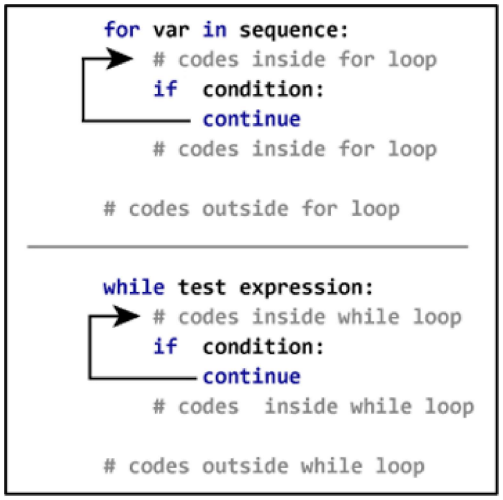
D. continue Statement

The **continue** statement is used to skip the rest of the code inside a loop for the current iteration only. Loop does not terminate but continues with the next iteration.

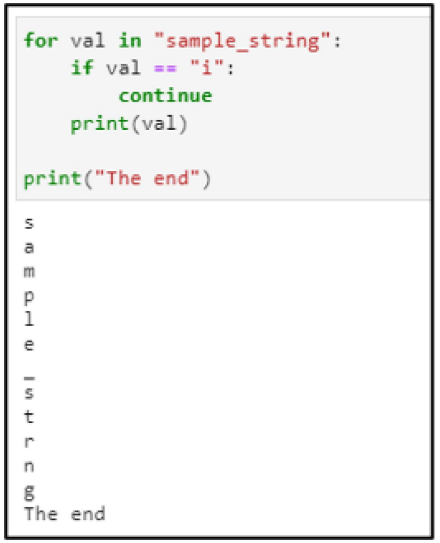
The Flowchart of *continue* Statement is:



The working of continue statement in for and while loop is shown below:



Consider the following example:



In today's modern age of disruption, SkillUp Online is your ideal learning platform that enables you to upskill to the most in-demand technology skills like Data Science, Big Data, Artificial Intelligence, Cloud, Front-End Development, DevOps & many more. In your journey of evolution as a technologist, SkillUp Online helps you work smarter, get to your career goals faster and create an exciting technology led future.

Corporate

- Home
- About Us
- Enterprise
- Blog
- Press

Support

- Contact us
- Terms of Service
- Privacy Policy