

LoopsLoopsLoops

A loop is basically a way to execute a block of code repeatedly, without having to write the code again and again. There are 2 types of loops in Python, namely: for loop and while loop.

For Loop

For Loops in Python are used for iterating over a sequence, be it a string, a list or a dictionary.

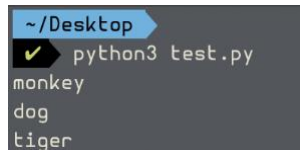
The syntax for a for loop is something like this:

```
1 for item in list:
2     # do something
```

An example that prints the contents of a list is:

```
1 animals = ["monkey","dog","tiger"]
2
3 for animal in animals:
4     print(animal)
```

For which the output is:

A terminal window with a dark background. The prompt is '~ / Desktop'. The command 'python3 test.py' has been executed, indicated by a green checkmark. The output of the script is printed on three separate lines: 'monkey', 'dog', and 'tiger'.

```
~/Desktop
✓ python3 test.py
monkey
dog
tiger
```

You can use the range() function to specify a certain range of numbers.
For example:

```
1 for i in range(5):
2     print(i)
```

This will print all the numbers from 0 to 4.

The range function can take up to 3 arguments, the lower bound, the upper bound and the increment number.

If only one argument is given, the lower bound is taken to be zero and the increment number is taken to be 1, by default,

Note: The upper bound is not included.

The While Loop

The while loop is a type of loop that requires a stop condition and will execute until that condition is met.

The syntax for a while loop is:

```
1 while (condition):  
2     # do something
```

For example:

```
1 x = 1  
2 while (x < 10):  
3     print(x)  
4     x+=1
```

Here, the exit condition is $x < 10$. So once x becomes greater than or equal to 10, the loop will end.

This loop prints the numbers from 1 to 9.

Note: x must be incremented or the while loop will be executed forever as the exit condition will never be met.

The break Statement

We can use the break statement to exit the loop despite the exit condition not being satisfied.

```
1 x = 1  
2 while (x < 10):  
3     if (x==5):  
4         break  
5     print(x)  
6     x+=1
```

This code will print numbers from 1 to 4 as when $x == 5$, the break condition is satisfied and the loop is exited.

The continue Statement

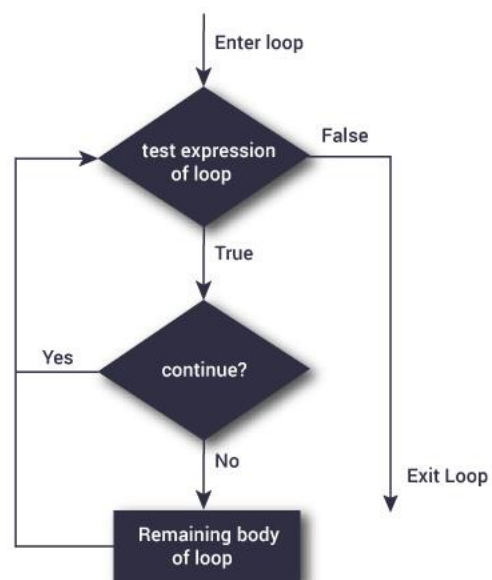
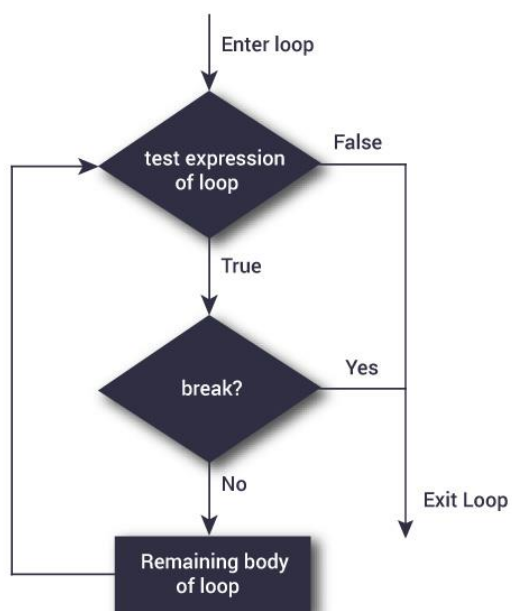
We can use the continue statement to skip the current iteration of the loop and go to the next one.

```
1  x = 1
2  while (x < 10):
3      if (x==5):
4          x+=1
5          continue
6      print(x)
7      x+=1
```

This code will print the numbers from 1 to 9 but not print 5, as when the `x == 5`, the condition to continue is met and the remaining code in the loop is skipped.

Flowcharts

Below are the flowcharts for break and continue, which will help your understanding of the working of the statements better.



Conditionals?:

There will be various situations while writing a program when you will have to take care of different possible conditions that might arise while execution of the program. In such situations, if conditions can be used.

The if Statement

The if statement is a simple conditional, where one condition is given, where if that condition is satisfied, the block of code under the statement is executed.

The syntax for the if statement in Python is

```
1  if condition:
2      # do something
```

An example of the usage of the if statement is

```
1  if charge == 100:
2      print ("Battery is full.")
```

Two other statements that can be paired with the if statement are: else and elif (else if).

The else Statement

The else statement can be paired with the if statement, to specify what to do if the if-statement turns out to be false.

The syntax for the else statement is

```
1  if condition:
2      # do something
3
4  else:
5      # do something else
```

Example of the usage of this is

```
1  if charge == 100:
2      print ("Battery is full.")
3
4  else:
5      print ("Battery is not full.")
```

There is also another statement called elif, which is equivalent to else if, which comes after the if statement, where you specify a condition for the else statement as well.

Implementations of the if statement

There are several ways that you can use the if statement, apart from the regular if-else implementation.

The nested if statement:

The nested if statement is, as the name suggests, when you have if statements inside if statements.

The syntax of this is:

```
1  if condition:
2      if condition:
3          if condition:
4              if condition:
5                  if condition:
6                      # do something
```

Note: You can have as many if statements as you like.