**LAB 2**

**Iterative Structures in Python**

Python programming language provides following types of loops to handle looping requirements. Python provides three ways for executing the loops. While all the ways provide similar basic functionality, they differ in their syntax and condition checking time.

**While Loop:**

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

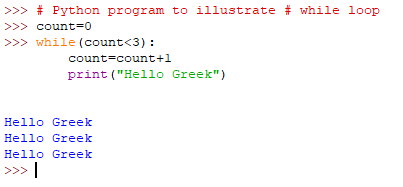
.

**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.

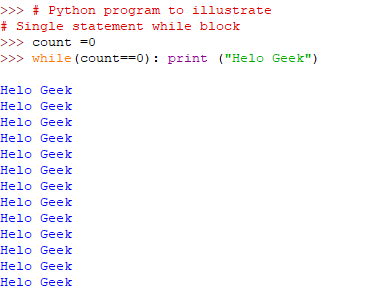
**Example:**



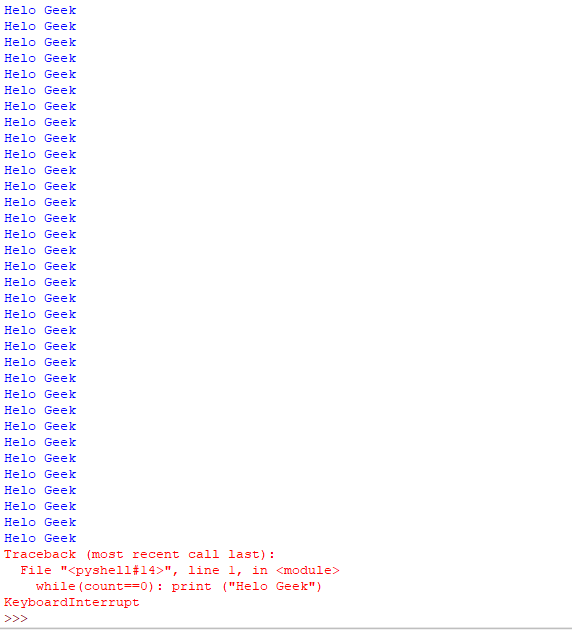
**Single statement while block:**

Just like the if block, if the while block consists of a single statement the we can declare the entire loop in a single line as shown below:

Example:



But it leads to infinite Loop



**for in Loop:**

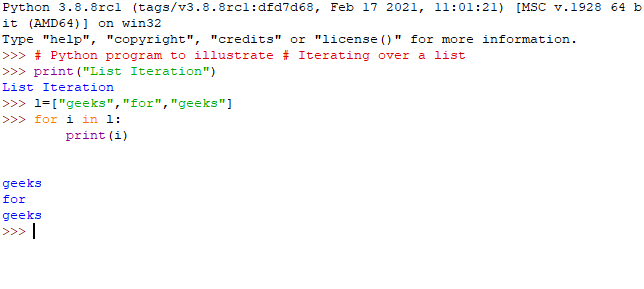
For loops are used for sequential traversal. For example: traversing a list or string or array etc. In Python, there is no C style for loop, i.e., for (i=0; i<n; i++). There is “for in” loop which is similar to for each loop in other languages. Let us learn how to use for in loop for sequential traversals.

**Syntax:**

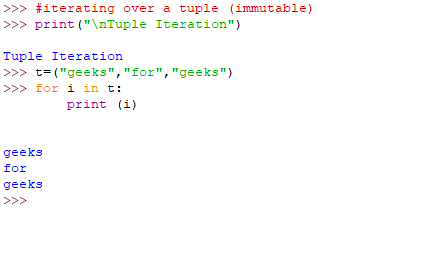
for iterator\_var in sequence: statements(s)

It can be used to iterate over iterators and a range

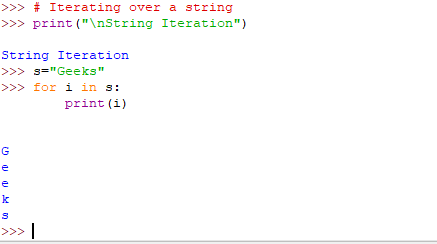
Example 1



Example 2



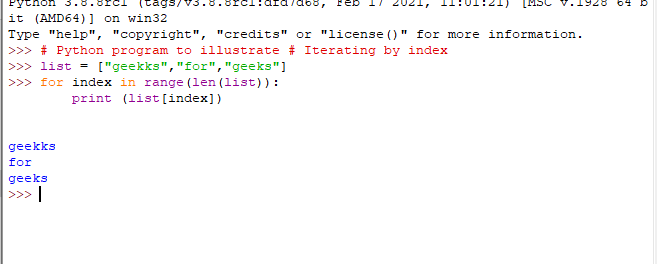
Example 3



**Iterating by index of sequences:**

We can also use the index of elements in the sequence to iterate. The key idea is to first calculate the length of the list and in iterate over the sequence within the range of this length.

See the below example:

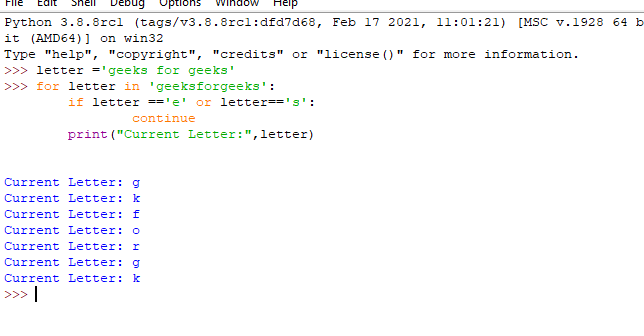


Loop Control Statements:

Loop control statements change execution from its normal sequence. When execution leaves a scope, all automatic objects that were created in that scope are destroyed. Python supports the following control statements.

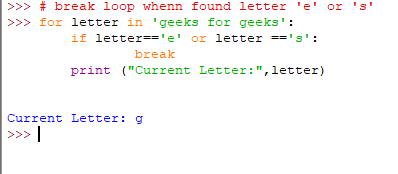
*Continue Statement:*

It returns the control to the beginning of the loop.



*Break Statement:*

It brings control out of the loop



**Python Functions**

A function is a block of code which only runs when it is called. You can pass data, known as parameters, into a function.

A function can return data as a result.

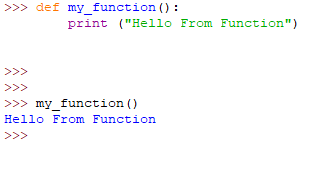
**Creating a Function**

In Python a function is defined using the def keyword:

Calling a Function

To call a function, use the function name followed by parenthesis:

Example



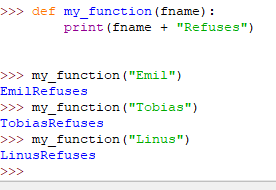
**Parameters**

Information can be passed to functions as parameter.

Parameters are specified after the function name, inside the parentheses. You can add as many parameters as you want, just separate them with a comma.

The following example has a function with one parameter (fname). When the function is called, we pass along a first name, which is used inside the function to print the full name:

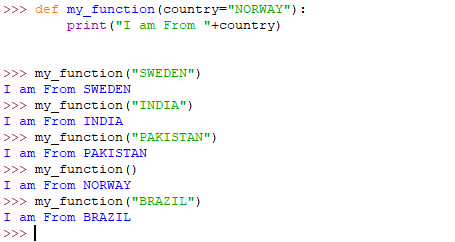
Example



**Default Parameter**

The following example shows how to use a default parameter value.

If we call the function without parameter, it uses the default value:

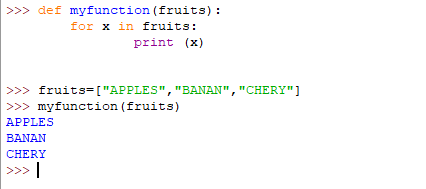


**Passing a List as a Parameter**

You can send any data types of parameter to a function (string, number, list, dictionary etc.), and it will be treated as the same data type inside the function.

E.g. if you send a List as a parameter, it will still be a List when it reaches the function:

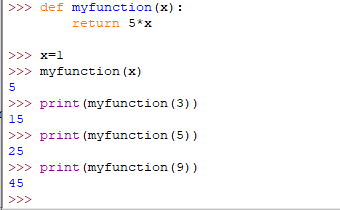
Example



**Return Values**

To let a function return a value, use the return statement:

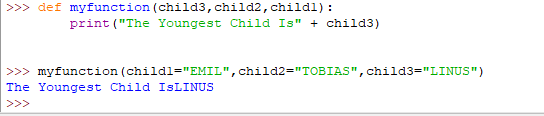
Example



**Keyword Arguments**

You can also send arguments with the *key* = *value* syntax. This way the order of the arguments does not matter.

Example



**Python Classes/Objects**

Python is an object oriented programming language.

Almost everything in Python is an object, with its properties and methods. A Class is like an object constructor, or a "blueprint" for creating objects.

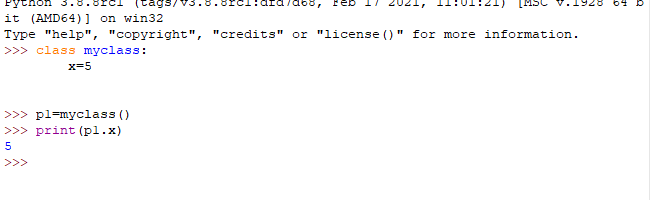
**Create a Class**

To create a class, use the keyword class.

**Create Object**

Now we can use the class named myClass to create objects:

**Example**



**The init () Function**

The examples above are classes and objects in their simplest form, and are not really useful in real life applications.

To understand the meaning of classes we have to understand the built-in init () function.

All classes have a function called init (), which is always executed when the class is being initiated.

Use the init () function to assign values to object properties, or other operations that are necessary to do when the object is being created:

**Example**

