Dear Learner,

In this module, you will get to learn about Functions,Files and Object Oriented Programming in Python.

* Request you to go through the content and complete it.
* After completing this course , you will have an understanding of how to define functions, create Python scripts,Modules and Programs and run modules as a script and how to use command line arguments in scripts.
* You will also learn about how to define classes and instantiate objects , use of instance methods and static methods, inheritance and properties.
* After going through the course contents provided, request you to take the quiz to understand your learning improvements.
* It is mandatory to complete the practice assignments provided in the Python\_Hands On course to proceed for the next module.

1. **Object Oriented Programming**

Till now, we have designed our programs using functions and loops. This is called procedural way of programming. There is another way of organizing our code into which is to combine data and functionalities together and wrap it inside something that is called an object. This is called object oriented approach of programming. It provides a modular structure of programming as compared to procedural way of programming. This is very helpful while developing large programs.

Classes and Objects are the main aspects of OOP. Lets discuss them one by one.

1.1.  **Object**

An object is an entity (tangible or intangible) that has well defined structure and behavior. In OOP, a program is seen as comprising of a collection of **objects,** that **act on each other.**Each object has a distinct role or responsibility.

**1.2. Class**

A class is a blueprint of an object. This defines a set of attributes that characterizes any object that is instantiated from this class.An object is a realized version of a class. It can also be defined as an abstract datatype.

**Classes Vs Objects**

|  |  |
| --- | --- |
| **Class** | **Object** |
| Class is a type/template for similar objects | Object is an instance of the class, with each instance behaving identically |
| Class is purely a static concept, represented by program text | Object is dynamic/run-time entity, occupying space in memory |
| Class is a logical entity | Object is a physical entity |

1.3. **Features of Object Oriented Programming**

There are many features of OOP. Of that , we consider four of them to be the pillars of Object Oriented Approach of Programming. They are :

**a**. Abstraction

b.Encapsulation

c. Inheritance

d.Polymorphism

**Abstraction**is a process where you show only “relevant” data and “hide” unnecessary details of an object from the user. Consider your mobile phone, you just need to know what buttons are to be pressed to send a message or make a call, What happens when you press a button, how your messages are sent, how your calls are connected is all abstracted away from the user.

**Encapsulation**is the process of combining data and functions into a single unit called class. In Encapsulation, the data is not accessed directly; it is accessed through the functions present inside the class. In simpler words, attributes of the class are kept private and public getter and setter methods are provided to manipulate these attributes. Thus, encapsulation makes the concept of data hiding possible.

Abstraction and Encapsulation can be achieved using access specifiers or including files as modules. These will be discussed elaborately in the property section in Python later.

Inheritance and Polymorphism will be discussed later.

2. **Classes and Objects in Python**

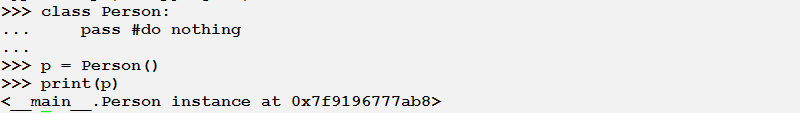
Now, let us see how to declare a class in Python

We define classes using the class keyword, similar to how we define functions using the def keyword.

**2.1 Basic Syntax**

|  |
| --- |
| class Classname:  statement 1  …................  statement N |

**Example**

****

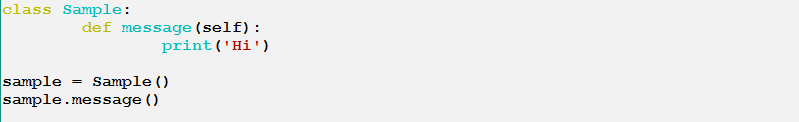
Objects can store data using ordinary variables that *belong*to the object. Variables that belong to an object or class are referred to as **fields**. Objects can also have functionality by using functions that *belong*to a class. Such functions are called **methods**of the class. This terminology is important because it helps us to differentiate between functions and variables which are independent and those which belong to a class or object. Collectively, the fields and methods can be referred to as the **attributes**of that class.

Fields are of two types - they can belong to each instance/object of the class or they can belong to the class itself. They are called **instance variables**and **class variables**respectively.

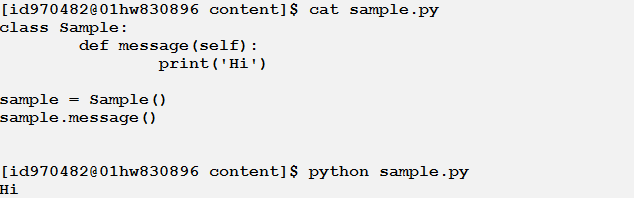
2.1. **The self keyword**

When we define a class method, a variable is needed to be provided at the beginning of the parameter list. But whenever this method is called, no value is needed to be provided for this parameter. Python provides it by default. This variable is nothing but the object itself which is referred using the self keyword in the method.

Let us look at an example now.



The output of this block is as such



***Note:****the self variable is almost similar to the this pointer in C++ and the this reference in Java or C#.*

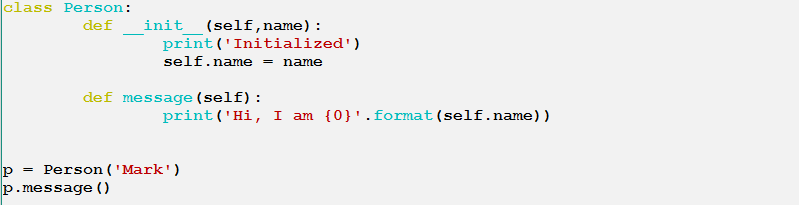
* 1. **Init Method**

There are various methods which have special significance in Python classes. The init method is one such method. It is used to initialize the object of a class. As soon as an object of a is created or instantiated, the init method is called automatically. We need not explicitly call the method.

Its almost similar to a constructor in c++ or Java.

Let us look at an example now how the init method is implemented

Filename: person.py



Output:

https://g91.tcsion.com/per/g91/pub/2030/LX/ckeditor_assets/pictures/2030/2783/image_41352_original.png

* 1. **Class Variables and Object Variables**

The variables (or data fields) of a class or an object are nothing but ordinary variables that are bound to the name spaces of the classes and objects. This means that these names are valid within the context of these classes and objects only.

There are two types of *data fields*- class variables and object variables which are classified depending on whether the class or the object *owns*the variables respectively.

**Class variables**are shared - they can be accessed by all instances of that class. There is only one copy of the class variable and when any one object makes a change to a class variable, that change will be seen by all the other instances.

**Object variables**are owned by each individual object/instance of the class. In this case, each object has its own copy of the field i.e. they are not shared and are not related in any way to the field by the same name in a different instance.

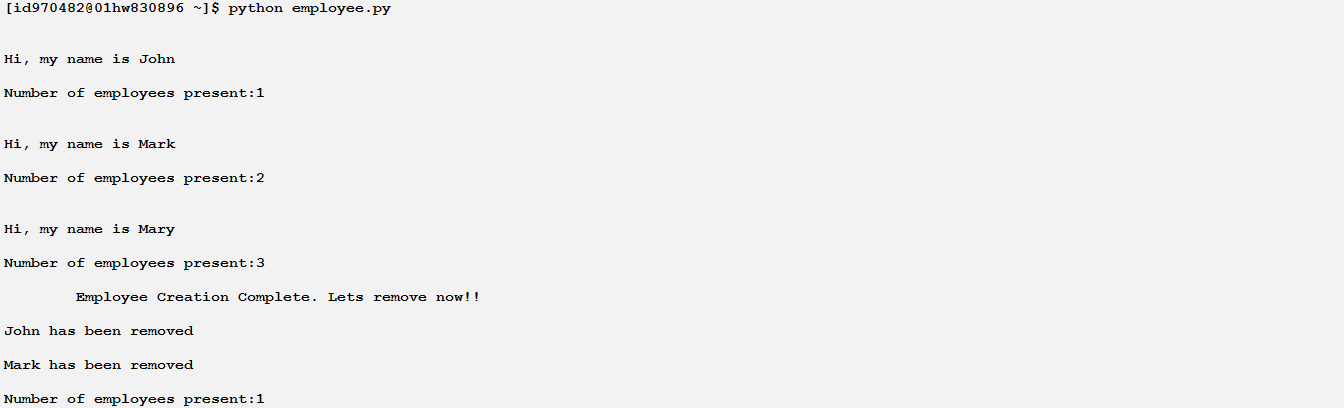
Let us look at an example which will make it a little easy to understand.

In this example, a person class is declared which has a class variable as count and a class method is declared to get the count of employees.

There are two other functionalities, one to remove and employee and another to receive a greeting from any employee. Both these methods are object variables i.e. they will be called by an instance of that class. Let us look at the implementation below.



Output:



As mentioned above, we can see that the methods to greet and to remove employee are being called by the objects employee1, employee2 and employee3. So the values returned in the output are different for every object. But the method get\_count() is called bt=y the class and hence returns the value with respect to the class.

1. **References & Video Link**

<https://pdfs.semanticscholar.org/presentation/6ee1/add06c8938ed02a1b4a5d01515c234051677.pdf>

<https://www.udemy.com/python-oops-beginners/>

<https://www.johnny-lin.com/pyintro/ed01/free_pdfs/ch07.pdf>

<http://pdf.th7.cn/down/files/1411/Mastering%20Object-oriented%20Python.pdf>

<http://abi.inf.uni-tuebingen.de/Teaching/Old/SS11/BILW/handouts-1/Object%20Oriented%20programming%20in%20Python.pdf>