# Python Oops

Python is a multi-paradigm programming language. It supports different programming approaches.

One of the popular approaches to solve a programming problem is by creating objects. This is known as Object-Oriented Programming (OOP).

## 1. Class

A Class is like an object constructor, or a "blueprint" for creating objects.Class is a template for an object.A reusable chunk of code that has methods and variables.

Like function definitions begin with the **def** keyword in Python, class definitions begin with a **class** keyword.

A class creates a **new local namespace** where all its attributes are defined. Attributes may be data or functions

.As soon as we define a class, a new class object is created with the same name. This class object allows us to access the different attributes as well as to instantiate new objects of that class.

**Create a Class**
To create a class, use the keyword class

**Create Object**
We can use the class name to create objects

## The __init__(Constructor) 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.

**Note:** The __init__() function is called automatically every time the class is being used to create a new object.

Class functions that begin with double underscore __are called special functions as they have special meaning. Of one particular interest is the __init__()function. This special function gets called whenever a new object of that class is instantiated.__init__ is an initialization method used to construct class instances in custom ways.

## Object Methods

Objects are an encapsulation of variables and functions into a single entity. Objects get their variables and functions from classes. Classes are essentially a template to create your objects

Object is an instance of a class.We saw that the class object could be used to access different attributes.Attributes may be data or method. **Methods of an object are corresponding functions of that class.**

### The self Parameter

The self parameter is a reference to the current instance of the class, and is used to access variables that belong to the class.

**Self is not a keyword , you can call it whatever you like, but it has to be the first parameter of any function in the class:**

### The pass Statement

**class definitions cannot be empty**, but if you for some reason have a class definition with no content, put in the pass statement **to avoid getting an error.**

## 2. Inheritance

Inheritance is a way of creating a new class for using details of an existing class without modifying it. The newly formed class is a **derived class (or child class)**. Similarly, the existing class is a **base class (or parent class).**

## 3. Encapsulation

Using OOP in Python, we can restrict access to methods and variables. This prevents data from direct modification which is called encapsulation. In Python, we denote private attributes using underscore as the prefix i.e single _ _ or double __.

## 4. Polymorphism

Polymorphism is an ability (in OOP) to use a common interface for multiple forms (data types).

Suppose, we need to color a shape, there are multiple shape options (rectangle, square, circle). However we could use the same method to color any shape. This concept is called Polymorphism**.**

## 5. Abstraction

Abstraction means hiding the complexity and only showing the essential features of the object. So in a way, Abstraction means hiding the real implementation and we, as a user, know only how to use it.

Abstraction in Python is achieved by using **abstract classes** and **interfaces**.

An abstract class is a class that generally provides incomplete functionality and contains one or more abstract methods. **Abstract methods are the methods that generally don’t have any implementation, it is left to the sub classes to provide implementation for the abstract methods.**

An interface should just provide the method names without method bodies. Subclasses should provide implementation for all the methods defined in an interface.

Note that in Python there is no support for creating interfaces explicitly, you will have to use abstract class. In Python you can create an interface using abstract class. If you create an abstract class which contains only abstract methods that acts as an interface in Python.