Object Oriented Programming

<https://www.freecodecamp.org/news/object-oriented-programming-in-python/>

<https://www.analyticsvidhya.com/blog/2020/09/object-oriented-programming/>

https://www.geeksforgeeks.org/python-oops-concepts/

1. What Is Object-Oriented Programming?
2. Object-Oriented Programming (OOP) vs Procedure Oriented Programming (POP)
3. Major Concepts of OOPs-
   1. What is a class?
   2. Object and object instantiation
   3. Class methods
   4. Inheritance in Python Class
   5. Encapsulation
   6. Polymorphism
   7. Data abstraction

What is a Class?

A class is a collection of objects.  Unlike the primitive data structures, classes are data structures that the user defines.

class class\_name:

class body

We define a class with a keyword “class” following the class\_name and colon.

### Objects and object instantiation

When we define a class only the description of the object is created. There is no memory allocation until we create its **object**. The **objector** **instance** contains real data or information.

Instantiation is nothing but creating a new object/instance of a class. Let’s create the object of the above class we defined-

obj1 = Car()

#### Class constructor

The job of the class constructor is to assign the values to the data members of the class when an object of the class is created.

There can be various properties of a car such as its name, color, model, brand name, engine power, weight, price, etc. We’ll choose only a few for understanding purposes.

class Car:

def \_\_init\_\_(self, name, color):

self.name = name

self.color = color

So, the properties of the car or any other object must be inside a method that we call **\_\_init\_\_( )**. This \_\_init\_\_() method is also known as **the constructor method**. We call a constructor method whenever an object of the class is constructed.

The two statements inside the constructor method are –

1. **self.name = name**
2. **self.color = color:**
3. This will create new attributes namely **name** and **color** and then assign the value of the respective parameters to them. The “self” keyword represents the instance of the class. By using the “self” keyword we can access the attributes and methods of the class. It is useful in method definitions and in variable initialization. The “self” is explicitly used every time we define a method.
4. Note: You can create attributes outside of this \_\_init\_\_() method also. But those attributes will be universal to the whole class and you will have to assign the value to them.
5. Suppose all the cars in your showroom are Sedan and instead of specifying it again and again you can fix the value of car\_type as Sedan by creating an attribute outside the \_\_init\_\_().

class Car:

car\_type = "Sedan" #class attribute

def \_\_init\_\_(self, name, color):

self.name = name #instance attribute

self.color = color #instance attribute

Here, **Instance attributes refer to**the attributes inside the constructor method i.e self.name and self.color. And, **Class attributes refer to**the attributes outside the constructor method i.e car\_type.