**Python assignment # 06:**

**Question 1:**

Define Object Oriented Programming Language?

**Object-oriented programming** (**OOP**) is a [programming paradigm](https://en.wikipedia.org/wiki/Programming_paradigm) based on the concept of "[objects](https://en.wikipedia.org/wiki/Object_(computer_science))", which can contain [data](https://en.wikipedia.org/wiki/Data), in the form of [fields](https://en.wikipedia.org/wiki/Field_(computer_science)) (often known as *attributes* or *properties*), and code, in the form of procedures (often known as [*methods*](https://en.wikipedia.org/wiki/Method_(computer_science))).

Many of the most widely used programming languages (such as C++, Java, Python, etc.) are [multi-paradigm](https://en.wikipedia.org/wiki/Multi-paradigm_programming_language) and they support object-oriented programming to a greater or lesser degree, typically in combination with [imperative](https://en.wikipedia.org/wiki/Imperative_programming), [procedural programming](https://en.wikipedia.org/wiki/Procedural_programming). Significant object-oriented languages include [Java](https://en.wikipedia.org/wiki/Java_(programming_language)), [C++](https://en.wikipedia.org/wiki/C%2B%2B), [C#](https://en.wikipedia.org/wiki/C_Sharp_(programming_language)), [Python](https://en.wikipedia.org/wiki/Python_(programming_language)), [PHP](https://en.wikipedia.org/wiki/PHP), [JavaScript](https://en.wikipedia.org/wiki/JavaScript), [Ruby](https://en.wikipedia.org/wiki/Ruby_(programming_language)), [Perl](https://en.wikipedia.org/wiki/Perl), [Object Pascal](https://en.wikipedia.org/wiki/Object_Pascal), [Objective-C](https://en.wikipedia.org/wiki/Objective-C), [Dart](https://en.wikipedia.org/wiki/Dart_(programming_language)), [Swift](https://en.wikipedia.org/wiki/Swift_(programming_language)), [Scala](https://en.wikipedia.org/wiki/Scala_(programming_language)), [Common Lisp](https://en.wikipedia.org/wiki/Common_Lisp), [MATLAB](https://en.wikipedia.org/wiki/MATLAB), and [Smalltalk](https://en.wikipedia.org/wiki/Smalltalk).

**Question 2:**

List down the Benefits of OOP?

1**. Modularity for easier troubleshooting**

When working with object-oriented programming languages, you know exactly where to look. “Oh, the car object broke down? The problem must be in the Car class!” You don’t have to muck through anything else.

## 2. Reuse of code through inheritance

Suppose that in addition to your Car object, one colleague needs a RaceCar object, and another needs a Limousine object. Everyone builds their objects separately but discover commonalities between them. In fact, each object is really just a different kind of Car. This is where the inheritance technique saves time: Create one generic class (Car), and then define the subclasses (RaceCar and Limousine) that are to inherit the generic class’s traits.

## 3. Flexibility through polymorphism

 a single function can shape-shift to adapt to whichever class it’s in, you could create one function in the parent Car class called “drive” — not “driveCar” or “driveRaceCar,” but just “drive.” This one function would work with the RaceCarDriver, LimousineDriver, etc. In fact, you could even have “raceCar.drive(myRaceCarDriver)” or “limo.drive(myChauffeur).”

## 4. Effective problem solving

Object-oriented programming is often the most natural and pragmatic approach, once you get the hang of it. OOP languages allows you to break down your software into bite-sized problems that you then can solve — one object at a time.

**Question 3:**

Differentiate between function and method?

A **function** is a piece of code that is called by name. It can be passed data to operate on (i.e. the parameters) and can optionally return data (the return value). All data that is passed to a function is explicitly passed.

A **method** is a piece of code that is called by a name that is associated with an object. In most respects it is identical to a function except for two key differences:

1. A method is implicitly passed the object on which it was called.
2. A method is able to operate on data that is contained within the class (remembering that an object is an instance of a class - the class is the definition, the object is an instance of that data).

**Question 4:**

Define following terms:

**Class:**

In [object-oriented programming](https://brilliant.org/wiki/objects/), a **class** is a blueprint for creating **objects** (a particular data structure), providing initial values for state (member variables or attributes), and implementations of behavior (member functions or methods).

**Object:**

In [object-oriented programming (OOP)](https://searchmicroservices.techtarget.com/definition/object-oriented-programming-OOP), objects are the things you think about first in designing a program and they are also the units of code that are eventually derived from the process. Each object is an instance of a particular class or subclass with the class's own methods or procedures and data variables.

**Attributes:**

In Object-oriented programming (OOP), classes and objects have attributes. **Attributes**are data stored inside a class or instance and represent the state or quality of the class or instance. In short, attributes store information about the instance.

**Behavior:**

A class's behavior determines how an instance of that class operates; for example, how it will "react" if asked to do something by another class or object or if its internal state changes. Behavior is the only way objects can do anything to themselves or have anything done to them.

**Question 5:**

Write a code in python in which create a class named it Car which have 5 attributes such like (model, color and name etc.) and 3 methods. And create 5 object instance from that class.

class Car:

def \_\_init\_\_(self,name, model, color, car\_no, engine\_type):

self.name=name

self.model=model

self.color=color

self.car\_no=car\_no

self.engine\_type=engine\_type

def car\_color():

print(“The color of the car is “,self.color)

def car\_model():

print(“The model of car is”, self.model)

def car\_name():

print(“the name of my car is” ,self.name)

c1=Car(“BMW”, 2016, “black”,”hg167”,”inline”)

c2=Car(“corolla”, 2011, “white”,”1hk76”,”inline”)

c3=Car(“mehran”, 2005, “grey”,”87KKA”,”inline”)

c1=Car(“cultus”, 2014, “Blue”,”MJ987”,”inline”)

c1=Car(“BMW”, 2019, “Purple”,”KHI64”,”inline”)