# **Python Assignment-06**

Q#01: Define Object-Oriented programming language?

Object-oriented language (OOL) is a high-level computer programming language that implements objects and their associated procedures within the programming context to create software programs.

Object-oriented language uses an object-oriented programming technique that binds related data and functions into an object and encourages reuse of these objects within the same and other programs.

Object-oriented language was primarily designed to reduce complexity in typical procedural languages through data binding and encapsulation techniques. In object-oriented language, the objects created provide limited or no access to other functions or methods within the program. This enables only authorized or inherited methods/functions to access a particular object.

Object-oriented language typically supports the following features, at minimum:

* The ability to create classes and their associated objects
* Encapsulation
* Inheritance

Python, Java, C++ and Smalltalk are popular examples of object-oriented languages.

Q#02: List down the benefits of OOP.

OOP’s top benefits:

## **1. Modularity for easier troubleshooting**

Something has gone wrong, and you have no idea where to look. Is the problem in the Widget file, or is it the WhaleFlumper? Will you have to trudge through that “sewage.c” file? Hope you commented your code!

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.

That’s the beauty of encapsulation. Objects are self-contained, and each bit of functionality does its own thing while leaving the other bits alone. Also, this modality allows an IT team to work on multiple objects simultaneously while minimizing the chance that one person might duplicate someone else’s functionality.

## **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.

Of course, Limousine and RaceCar still have their unique attributes and functions. If the RaceCar object needs a method to “fireAfterBurners” and the Limousine object requires a Chauffeur, each class could implement separate functions just for itself. However, because both classes inherit key aspects from the Car class, for example the “drive” or “fillUpGas” methods, your inheriting classes can simply reuse existing code instead of writing these functions all over again.

What if you want to make a change to all Car objects, regardless of type? This is another advantage of the OO approach. Simply make a change to your Car class, and all car objects will simply inherit the new code.

## **3. Flexibility through polymorphism**

Riffing on this example, you now need just a few drivers, or functions, like “driveCar,” driveRaceCar” and “DriveLimousine.” RaceCarDrivers share some traits with LimousineDrivers, but other things, like RaceHelmets and BeverageSponsorships, are unique.

This is where object-oriented programming’s sweet polymorphism comes into play. Because 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**

A language like C has an amazing legacy in programming history, but writing software in a top-down language is like playing Jenga while wearing mittens. The more complex it gets, the greater the chance it will collapse. Meanwhile, writing a functional-style program in a language like [Haskell](https://www.haskell.org/) or [ML](https://en.wikipedia.org/wiki/ML_(programming_language)) can be a chore.

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.

This isn’t to say that OOP is the One True Way. However, the advantages of object-oriented programming are many. When you need to solve complex programming challenges and want to add code tools to your skill set, OOP is your friend — and has much greater longevity and utility than Pac-Man or parachute pants.

Q#03: Define the following terms:

1. 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).
2. Object:  An **object** is a component of a program that knows how to perform certain actions and how to interact with other elements of the program. Objects are the basic units of object-oriented programming. A simple example of an object would be a person. Logically, you would expect a person to have a name. This would be considered a property of the person. You could also expect a person to be able to do something, such as walking or driving. This would be considered a method of the person.
3. Attribute: 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. Also, attributes should not be confused with class functions also known as methods. One can think of attributes as noun or adjective, while methods are the verb of the class.
4. Behavior/ method: A **method** in object-oriented programming is a procedure associated with a class. A method defines the behavior of the objects that are created from the class. Another way to say this is that a method is an action that an object is able to perform. The association between method and class is called binding. Consider the example of an object of the type 'person,' created using the person class. Methods associated with this class could consist of things like walking and driving. Methods are sometimes confused with functions, but they are distinct.