**Question No. 1**

**Introduction: -**

Object Oriented Programming is a paradigm in which real-world objects are each viewed as separate entities having their own state which is modified only by built in procedures, called method.

To “model human interaction with real world phenomena” and their behavior as classes and methods.

**Object Oriented Programming in Python: -**

Python is a multi-paradigm programming language. Meaning, it supports different programming approach. One of the popular approach to solve a programming problem is by creating objects. This is known as Object-Oriented Programming (OOP).

An object has two characteristics: -

1• attributes

2• behavior

The concept of OOP in Python focuses on creating reusable code. This concept is also known as DRY (Don't Repeat Yourself).

In Python, the concept of OOP follows some basic principles:

**Inheritance: -**

A process of using details from a new class without modifying existing class.

**Encapsulation: -**

Hiding the private details of a class from other objects.

### Polymorphism: -

A concept of using common operation in different ways for different data input.

# Question No. 2

## Benefits of OOP: -

* **Modular Structure : -** It provides a clear modular structure for programs which makes it good for defining abstract data types in which implementation details are hidden.
* **Reused: -** Objects can also be reused within an across applications. The reuse of software also lowers the cost of development. More effort is put into the object-oriented analysis and design, which lowers the overall cost of development.
* **Lower cost of development: -** The reuse of software also lowers the cost of development. Typically, more effort is put into the object-oriented analysis and design, which lowers the overall cost of development
* **Easier to maintain: -**  
  It makes software easier to maintain. Since the design is modular, part of the system can be updated in case of issues without a need to make large-scale changes
* **Faster development: -** Reuse also enables faster development. Object-oriented programming languages come with rich libraries of objects, and code developed during projects is also reusable in future projects.
* **Easily adapted and modified by the programmer: -** It provides a good framework for code libraries where the supplied software components can be easily adapted and modified by the programmer. This is particularly useful for developing graphical user interfaces.
* ***Better Productivity: -***  
  Better Productivity as OOP techniques enforce rules on a programmer that, in the long run, help her get more work done; finished programs work better, have more features and are easier to read and maintain.
* ***Higher quality software: -***  
  Faster development of software and lower cost of development allows more time and resources to be used in the verification of the software. Although quality is dependent upon the experience of the teams, object oriented programming tends to result in higher-quality software

# Question No. 3

The key differences between Method and Function in Python are given below: -

* **Python Method**

1. Method is called by its name, but it is associated to an object (dependent).
2. A method is implicitly passed the object on which it is invoked.
3. It may or may not return any data.
4. A method can operate on the data (instance variables) that is contained by the corresponding class
5. **Functions**
6. Function is block of code that is also called by its name. (independent)
7. The function can have different parameters or may not have any at all. If any data (parameters) are passed, they are passed explicitly.
8. It may or may not return any data.
9. Function does not deal with Class and its instance concept.
10. **Difference between method and function**
11. Simply, function and method both look similar as they perform in almost similar way, but the key difference is the concept of ‘Class and its Object‘.
12. Functions can be called only by its name, as it is defined independently. But methods can’t be called by its name only, we need to invoke the class by a reference of that class in which it is defined, i.e. method is defined within a class and hence they are dependent on that class.

# Question No. 4

## Definition of Terminologies: -

1) **Class: -** Class is a template or blue print about the capability of what an object can do or a class is a blueprint from which you can create the instance, i.e., objects. It is used to bind data as well as methods together as a single unit. Classes have logical existence. And doesn't take any memory spaces when a programmer creates one. The class has to be declared only once. Every class contains attributes and behaviors.

2) **Object: -** An object (instance) is an instantiation of a class. When class is defined, only the description for the object is defined. Therefore, no memory or storage is allocated or an object is the instance of the class, which helps programmers to use variables and methods from inside the class. Its acts as a variable of the class. Objects have a physical existence and takes memory when a programmer creates one. An Objects can be declared several times depending on the requirement.

3) **Attribute: -** Attributes are the characteristics of the class that help to distinguish it from other classes. Attributes are called properties and properties are presented with different data types. For example a person's attributes, include their age, name, and height

4) **Behavior: -** Behaviors are the tasks that an object performs. Behavior are called functions. Behaviors are described using functions. For example a person’s behaviors include the fact that a person can speak, run, walk, and eat.

**class** **Car**():

**def** \_\_init\_\_(self,name,make,model,color,price):

self.name = name

self.make = make

self.model = model

self.color = color

self.price = price

**def** updatecolor(self,color):

self.color = "red"

**def** updateprice(self,price):

self.price = "Rs. 17 million"

**def** updatemodel(self,model):

self.model = 2002

car = Car("C200","mercedez",2019,"black","Rs. 15 million")

print(car.name)

print(car.make)

print(car.model)

print(car.color)

print(car.price)

print(car.color, "updated new color")

print(car.price, "updated new price")

print(car.model, "updated new model")