Question 1:
Define Object Oriented Programming Language?

Stands for "Object-Oriented Programming." OOP (not Oops!) refers to a programming methodology based on objects, instead of just functions and procedures. These objects are organized into classes, which allow individual objects to be grouped together. Most modern programming languages including Java, C/C++, and PHP, are object-oriented languages, and many older programming languages now have object-oriented versions.

An "object" in an OOP language refers to a specific type, or "instance," of a class. Each object has a structure similar to other objects in the class, but can be assigned individual characteristics. An object can also call functions, or methods, specific to that object.

Object-oriented programming makes it easier for programmers to structure and organize software programs. Because individual objects can be modified without affecting other aspects of the program, it is also easier to update and change programs written in object-oriented languages. As software programs have grown larger over the years, OOP has made developing these large programs more manageable.

Question 2:
List down the Benefits of OOP?

- Re-usability
- Data Redundancy
- Code Maintenance
- Security
- Design Benefits
- Better productivity
- Easy troubleshooting
- Polymorphism Flexibility
- Problems solving

Question 3:
Differentiate between function and method?

Python Method

- Method is called by its name, but it is associated to an object (dependent).
- A method is implicitly passed the object on which it is invoked.
- It may or may not return any data.
- A method can operate on the data (instance variables) that is contained by the corresponding class

Functions

- Function is block of code that is also called by its name. (independent)
- The function can have different parameters or may not have any at all. If any data (parameters) are passed, they are passed explicitly.
- It may or may not return any data.
- Function does not deal with Class and its instance concept.

Question 4:
Define the following terms:

1. **Class**

In object-oriented programming, 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).

The user-defined objects are created using the class keyword. The class is a blueprint that defines a nature of a future object. An instance is a specific object created from a particular class. Classes are used to create and manage new objects and support inheritance—a key ingredient in object-oriented programming and a mechanism of reusing code.

2. **Object**

An object, in object-oriented programming (OOP), is an abstract data type created by a developer. It can include multiple properties and methods and may even contain other objects. In most programming languages, objects are defined as classes.

Objects provide a structured approach to programming. By defining a dataset as a custom object, a developer can easily create multiple similar objects and modify existing objects within a program. Additionally, objects provide "encapsulation," meaning the data within an object is protected from being modified or destroyed by other functions or methods unless explicitly allowed.

3. **Attribute**

In using or programming computers, an attribute is a changeable property or characteristic of some component of a program that can be set to different values.

In the Hypertext Markup Language (HTML), an attribute is a characteristic of a page element, such as a font. An HTML user can set font attributes, such as size and color, to different values. In some programming languages, such as PowerBuilder PowerScript, an attribute is a property of an object or may be considered a container for the property of the object. For example, color might be an attribute of a text object, containing the value of "red."

In a database management system (DBMS), an attribute may describe a component of the database, such as a table or a field, or may be used itself as another term for a field.

4. **Behavior**

The behavior of an object is defined by its methods, which are the functions and subroutines defined within the object class. Without class methods, a class would simply be a structure.

Methods determine what type of functionality a class has, how it modifies its data, and its overall behavior. Methods go beyond what properties do for an object, in that they aren't bound by a rigid implementation guideline and aren't tied to a member data item. They can be subroutines or functions and declared as public, protected, or private.

A class that inherits another class can also override methods of the inherited class to enhance or change its behavior. If the new class changes a method's behavior, it overrides the method and doesn't call the base class's method