# Assignment No.6

## Question No.1

### Define Object Oriented Programming Language?

Object-oriented programming (OOP) is a programming language model in which programs are organized around data, or objects, rather than functions and logic. An object can be defined as a data field that has unique attributes and behavior. Examples of an object can range from physical entities, such as a human being that is described by properties like name and address, down to small computer programs, such as widgets. This opposes the historical approach to programming where emphasis was placed on how the logic was written rather than how to define the data within the logic.

The first step in OOP is to identify all of the objects a programmer wants to manipulate and how they relate to each other, an exercise often known as data modeling. Once an object is known, it is generalized as a class of objects that defines the kind of data it contains and any logic sequences that can manipulate it. Each distinct logic sequence is known as a method and objects can communicate with well-defined interfaces called messages.

Simply put, OOP focuses on the objects that developers want to manipulate rather than the logic required to manipulate them. This approach to programming is well-suited for programs that are large, complex and actively updated or maintained. Due to the organization of an object-oriented program, this method is also conducive to collaborative development where projects can be divided into groups. Additional benefits of OOP include code reusability, scalability and efficiency.

## Principles of OOP

Object-oriented programming is based on the following principles:

- **Encapsulation**- The implementation and state of each object are privately held inside a defined boundary, or class. Other objects do not have access to this class or the authority to make changes but are only able to call a list of public functions, or methods. This characteristic of data hiding provides greater program security and avoids unintended data corruption.
- **Abstraction**- Objects only reveal internal mechanisms that are relevant for the use of other objects, hiding any unnecessary implementation code. This concept helps developers make changes and additions over time more easily.
- **Inheritance**- Relationships and subclasses between objects can be assigned, allowing developers to reuse a common logic while still maintaining a unique hierarchy. This property of OOP forces a more thorough data analysis, reduces development time and ensures a higher level of accuracy.
- **Polymorphism**- Objects can take on more than one form depending on the context. The program will determine which meaning or usage is necessary for each execution of that object, cutting down on the need to duplicate code.

# Question No.2

## List down the Benefits of OOP?

- It provides a clear ***modular structure*** for programs which makes it good for defining abstract datatypes in which implementation details are hidden
- 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.
- 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
- 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.
- 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*** **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. OOP programmers take new and existing software objects and "stitch" them together to make new programs. Because object libraries contain many useful functions, software developers don't have to reinvent the wheel as often; more of their time goes into making the new program.

# Question No.3

Differentiate between function and method?

## Function

A function is a block of code to carry out a specific task, will contain its own scope and is called by name. All functions may contain zero(no) arguments or more than one argument. On exit, a function can or cannot return one or more values.

## Basic function syntax

```
def functionName( arg1, arg2,….):
  …….
  # Function_body
  ……..
```

## Method

A method in python is somewhat like a function, except it is associated with object/classes. Methods in python are very similar to functions except for two major differences.

The method is implicitly used for an object for which it is called.

The method is accessible to data that is contained within the class.

## General Method Syntax

```
class ClassName:
  def method_name():
    ……..
    # Method_body
    ……
```

# Question No.4

Define the following terms

## Class:

A class is a code template for creating objects. Objects have member variables and have behavior associated with them. In python a class is created by the keyword class . An object is created using the constructor of the class. This object will then be called the instance of the class.

## Object:

Python is an object-oriented programming language. Unlike procedure-oriented programming, where the main emphasis is on functions, object-oriented programming stress on objects. Object is simply a collection of data (variables) and methods (functions) that act on those data. And, class is a blueprint for the object

## Attribute:

As an object-oriented language, Python provides two scopes for attributes: class attributes and instance attributes.

While the instance attribute in Python has the same characteristics and definition as the other object-oriented languages, the class attribute is always mistakenly considered to be the exact equivalent of the static attribute in Java or C++. To be accurate, class attributes in Python and static attributes in Java or C++ have a lot in common, however, they have behavioral differences that I will highlight in this article.

### Class Attribute vs. Instance Attribute

Let's start with the basics:

- An instance attribute is a Python variable belonging to one, and only one, object. This variable is only accessible in the scope of this object and it is defined inside the constructor function, `__init__(self,..)` of the class.
- A class attribute is a Python variable that belongs to a class rather than a particular object. It is shared between all the objects of this class and it is defined outside the constructor function, `__init__(self,...)`, of the class.

## Behavior:

Behave. behave is behavior-driven development, Python style. Behavior-driven development (or BDD) is an agile software development technique that encourages collaboration between developers, QA and non-technical or business participants in a software project.