**Python Advance Assignment 1**

**Q1. What is the purpose of Python’s OOP?**

In Python, object-oriented Programming (OOPs) is a programming paradigm that uses objects and classes in programming. The main concept of OOPs is to bind the data and the functions that work on that together as a single unit so that no other part of the code can access this data. It aims to implement real-world entities like inheritance, polymorphisms, encapsulation, etc. in the programming.

**Q2. Where does an inheritance search look for an attribute?**

Inheritance is the capability of one class to derive or inherit the properties from another class. The class that derives properties is called the derived class or child class and the class from which the properties are being derived is called the base class or parent class.

As a dynamically typed scripting language, Python removes much of the syntactic clutter and complexity that clouds OOP in other tools. In fact, most of the OOP story in Python boils down to this expression: object.attribute

We've been using this expression throughout the book to access module attributes, call methods of objects, and so on. When we say this to an object that is derived from a class statement, however, the expression kicks off a search in Python—it searches a tree of linked objects, looking for the first appearance of attribute that it can find. When classes are involved, the preceding Python expression effectively translates to the following in natural language:

Find the first occurrence of attribute by looking in object, then in all classes above it, from bottom to top and left to right.

In other words, attribute fetches are simply tree searches. The term inheritance is applied because objects lower in a tree inherit attributes attached to objects higher in that tree.

**Q3. How do you distinguish between a class object and an instance object?**

Everything in Python is an object such as integers, lists, dictionaries, functions and so on. Every object has a type and the object types are created using classes.Instance is an object that belongs to a class. For instance, list is a class in Python. When we create a list, we have an instance of the list class. Class variables are declared inside a class but outside of any function. Instance variables are declared inside the constructor which is the \_\_init\_\_method.

**Q4. What makes the first argument in a class’s method function special?**

Generally, when we call a method with some arguments, the corresponding class function is called by placing the method's object before the first argument. So, anything like **obj.meth(args)** becomes **Class.meth(obj, args)**. The calling process is automatic while the receiving process is not (its explicit).

This is the reason the first parameter of a function in class must be the object itself. Writing this parameter as self is merely a convention. It is not a keyword and has no special meaning in Python. We could use other names (like this) but it is highly discouraged. Using names other than self is frowned upon by most developers and degrades the readability of the code (**Readability counts**).

**Q5. What is the purpose of the \_\_init\_\_ method?**

The \_\_init\_\_ method is similar to **constructors**in C++ and Java. Constructors are used to initialize the object’s state. The task of constructors is to initialize (assign values) to the data members of the class when an object of class is created. Like methods, a constructor also contains collection of statements (i.e. instructions) that are executed at time of Object creation. It is run as soon as an object of a class is instantiated. The method is useful to do any initialization we want to do with our object.

**Q6. What is the process for creating a class instance?**

A class is a template for storing data and functions in an object. A class can be instantiated to create a unique instance. A class can accept parameters in the constructor. To pass parameters to the class instance, the class must have a \_\_init\_\_  method. Pass the parameters in the constructor of the class.

**Q7. What is the process for creating a class?**

A class is a user-defined blueprint or prototype from which objects are created. Classes provide a means of bundling data and functionality together. Creating a new class creates a new type of object, allowing new instances of that type to be made. Each class instance can have attributes attached to it for maintaining its state. Class instances can also have methods (defined by their class) for modifying their state.

Class creates a user-defined data structure, which holds its own data members and member functions, which can be accessed and used by creating an instance of that class. A class is like a blueprint for an object.

Some points on Python class:

* Classes are created by keyword class.
* Attributes are the variables that belong to a class.
* Attributes are always public and can be accessed using the dot (.) operator. Eg.: Myclass.Myattribute

**Q8. How would you define the superclasses of a class?**

The class from which a class inherits is called the parent or superclass. A class which inherits from a superclass is called a subclass, also called heir class or child class. Superclasses are sometimes called ancestors as well.