Assignment 1

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

Ans. OOP or Object oriented programming is a design methodology which helps in designing better applications or softwares. It allows us to develop applications using an Object-Oriented approach. In Python, we can easily create and use classes and objects. By which it focuses on writing the reusable code. It is a widespread technique to solve the problem by making things.

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

Ans. An inheritance search looks for an attribute first in the instance object, then in the class the instance was created from, then in all higher superclasses, progressing from left to right (by default). The search stops at the first place the attribute is found.

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

Ans. A class is a type of blueprint that you can use to make objects. A concrete 'thing' that you constructed using a certain class is an object, which is an instance of a class. So, while the terms 'object' and 'instance' are interchangeable, the term 'instance' refers to an object's relationship to its class

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

Ans. 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.

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

Ans. The \_\_init\_\_() method is used to initialize the instance variables when ever an object of the class is created. This method gets invoked when we make a call through the constructor of the class.

Ex:

class Person:

def \_\_init\_\_(self, name, age):

self.name=name

self.age=age

p1=Person(“Latha”,35)

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

Ans. To create instances of a class, you call the class using class name and pass in whatever arguments its *\_\_init\_\_* method accepts.

Ex:

class Employee:

def \_\_init\_\_(self, name, salary):

self.name = name

self.salary = salary

e1 =Employee(“Latha”,50000)

Q7. What is the process for creating a class?

Ans. The *class* statement creates a new class definition. The name of the class immediately follows the keyword *class* followed by a colon as follows −

class Point:

def \_\_init\_\_( self, x=0, y=0):

self.x = x

self.y = y

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

Ans. 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. There exists a hierarchical relationship between classes.

**class** **Robot**:

**def** **\_\_init\_\_**(self, name):

self.name = name

**def** say\_hi(self):

**print**("Hi, I am " + self.name)

**class** **PhysicianRobot**(Robot):

**pass**

x = Robot("Marvin")

y = PhysicianRobot("James")

**print**(x, **type**(x))

**print**(y, **type**(y))

y.say\_hi()