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

Ans 1 : Python’s OOP stands for Object Oriented Programming. OOP help in implementing real world problems. OOP uses objects and classes in programming. For implementing real world problems, OOP uses inheritence, polymorphism, abstraction, etc.

OOP helps in implementing security as class and objects are implemented and no other code outside that class or object can access that code.

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

Ans 2 : In inheritance search when an object attribute is created, it first look for the instance of the object attribute, then for the class and then the superclass.

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

Ans 3 : A class object is created when we create or define a class whereas an instance object is created when we call a class.

For example :

Class classname:

some code here

This code will create a new class and it is called class object

Now to call or use this class object we need to create a object of its type.

Obj1 = classname()

Obj1 is an instance object and it is used to call or create an instance of the class classname.

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

Ans 4 : The first argument in a class’s method function is self (generally used, we can use any name). The first argument (self) used to initialise the value of the variable member of the methods. For example :

def function\_name(self, param1):

self.param1 = param1

In self.param1 = param1 -> In this expression, on the left self.param1, we can use any name for param1, it is the variable of the method. In the expression, on the right param1, param1 is the parameter passed in the definition of the function.

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

Ans 5 : \_\_init\_\_ method is generally a constructor which is used to initialise the value of the data members of the class as soon as the object of the class is created. We can create as many as object of a class and \_\_init\_\_ method initialise the value of the data member of each object created for that class. For example :

class class\_name :

def \_\_init\_\_(self, var1):

self.var1 = var1

def function\_name (self):

print(self.var1)

obj1 = class\_name(‘Any Data’)

obj1.function\_name()

Explanation : In the class defenition, the function \_\_init\_\_ is called as soon as the object of the class is created and the value of var1 is initialised as the value passed.

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

Ans 6 : An object is known as a class instance. When we create a class, it works as blueprint. We can create any number of objects or class instances with different data. The purpose of creating a class instance or object is to reuse a structure of code with different data which is known as code reusability.

Q7. What is the process for creating a class?

Ans 7 : The process for creating a class :

First we define the class :

Class class\_name:

def \_\_init\_\_(parameters):

some definetion here

def funct1():

some definetion

Any number of methods you can define

To use the class, we need to create the instance of the class also known as object.

object\_name = class\_name(arguments)

To call the methods of the class :

object\_name.method\_name

eg :

object\_name.funct1()

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

Ans 8 : If we have defined two classes say a parent and child class. In the class child we can use super() to use the functionality of the parent class. Thus this also help in implementing the multiple inheritence in python OOPs. For eg :

Class A:

def \_\_init\_\_(self, var1,var2):

self.var1 = var1

self.var2 = var2

Class B(A) :

def \_\_init\_\_(self, var1, var2, var3):

super().\_\_init\_\_(var1, var2)

self.var3 = var3

As we are using super() keyword, it helps in implementing multiple inheritence.