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

**Answer:** In Python, object-oriented programming (OOPS) is a programming paradigm that uses object and classes in programming. It aims to implement real-world entities like inheritance, polymorphisms, encapsulation, etc. in the 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.

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

**Answer:** The inheritance search is simply a search of the tree from bottom to top looking for the lowest occurrence of an attribute name. Code implies the shape of such trees.

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

**Answer:** A class is a template for creating objects in program whereas, the object is an instance of a class. A class is a local entity while object is a physical entity. A class doesn’t allocate memory spaces when it is created while object allocates memory spaces when they are created.

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

**Answer:** The first argument of every class method, including init, is always a reference to the current instance of the class. By convention, this argument is always named self. In the init method, self refers to the newly created object; in other class methods, it refers to the instance whose method was called.

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

**Answer:** The \_\_init\_\_ function is called every time an object is created from a class. The \_\_init\_\_ method lets the class initialize the object's attributes and serves no other purpose. It is only used within classes.

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

**Answer:** When you create an object, you are creating an instance of a class, therefore "instantiating" a class. The new operator requires a single, postfix argument: a call to a constructor. The name of the constructor provides the name of the class to instantiate. The constructor initializes the new object.

Q7. What is the process for creating a class?

**Answer:** To create a class, use the keyword class:

Example: Create a class named MyClass, with a property named x:

Class MyClass:

x = 5

Now we can use the class named MyClass to create objects:

Create an object named p1, and print the value of x:

p1= MyClass()

print(p1.x)

The examples above are classes and objects in their simplest form, and are not really useful in real life applications. To understand the meaning of classes we have to understand the built-in \_\_init\_\_() function. All classes have a function called \_\_init\_\_(), which is always executed when the class is being initiated. Use the \_\_init\_\_() function to assign values to object properties, or other operations that are necessary to do when the object is being created:

Create a class named Person, use the \_\_init\_\_() function to assign values for name and age:

class Person:

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

self.name = name

self.age = age

p1 = Person("John", 36)

print(p1.name)

print(p1.age)

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

**Answer:** A superclass is the class from which many subclasses can be created. The subclasses inherit the characteristics of a superclass. The superclass is also known as the parent class or base class. In the above example, Vehicle is the Superclass and its subclasses are Car, Truck and Motorcycle.