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. 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?

The whole point of a namespace tool like the class statement is to support name inheritance. In Python, inheritance happens when an object is qualified, and involves searching an attribute definition tree (one or more namespaces). Every time you use an expression of the form object.

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

**A class is a logical entity while object is a physical entity**. A class does not allocate memory space on the other hand object allocates memory space. You can declare class only once but you can create more than one object using a class. Classes can't be manipulated while objects can be manipulated.

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

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?

The \_\_init\_\_ method is the Python equivalent of the C++ constructor in an object-oriented approach. 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?**

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

**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. The new operator returns a reference to the object it created.

Q7. What is the process for creating a class?

**A Class is like an object constructor, or a "blueprint" for creating objects.**

1. Create a Class. To create a class, use the **keyword** class .
2. Create Object. Now we can use the class named MyClass to create objects.
3. The self Parameter.
4. Modify Object Properties.
5. Delete Object Properties.
6. Delete Objects.

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

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.

**Car, Truck and Motorcycle are all subclasses of the superclass Vehicle. They all inherit common attributes from vehicle such as speed, colour etc. while they have different attributes also i.e Number of wheels in Car is 4 while in Motorcycle is 2.**

Inheritance

Inheritance is basically the process of basing a class on another class i.e to build a class on a existing  class. The new class contains all the features and functionalities of the old class in addition to its own.

The class which is newly created is known as the subclass or child class and the original class is the parent class or the superclass.

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

Classes may generate instances (objects), and have per-instance state (instance variables). **Modules may be mixed in to classes and other modules**. The mixed in module's constants and methods blend into that class's own, augmenting the class's functionality. Classes, however, cannot be mixed in to anything.

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

The whole point of a namespace tool like the class statement is to support name inheritance. In Python, inheritance happens when an object is qualified, and involves searching an attribute definition tree (one or more namespaces). Every time you use an expression of the form object.

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

A class describes a data type. An instance of a class is an object of the data type that exists in memory.

**A class is a logical entity while object is a physical entity**. A class does not allocate memory space on the other hand object allocates memory space. You can declare class only once but you can create more than one object using a class. Classes can't be manipulated while objects can be manipulated.

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

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?

\_\_init\_\_ method

"\_\_init\_\_" is a reseved method in python classes. It is called as a constructor in object oriented terminology. This method is called when an object is created from a class and it **allows the class to initialize the attributes of the class**.

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

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 new operator returns a reference to the object it created.

Q7. What is the process for creating a class?

**How to create an object from class in Java?**

Declaration − A variable declaration with a variable name with an object type.

Instantiation − The 'new' keyword is used to create the object.

Initialization − The 'new' keyword is followed by a call to a constructor. This call initializes the new object.

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

The **Python** super() function returns objects represented in the **parent's class** and enables multiple inheritances.