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

Ans- Python's Object-Oriented Programming (OOP) allows programmers to organize their code in a more structured and modular way, which can make it easier to write, debug, and maintain. The purpose of Python's OOP is to provide a way to create objects that encapsulate data and behaviour, allowing for more efficient and organized code.

The primary purpose of Python's OOP is to improve code reusability and maintainability by encapsulating data and behaviour into discrete, reusable objects that can be easily reused across different parts of the codebase

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

Ans- In Python, when an object attempts to access an attribute or method that is not defined in its own class, the interpreter will search for the attribute or method in its parent classes, using a process known as inheritance search.

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

Ans-In Python, a class object is a blueprint for creating instance objects. A class object defines the properties and behaviour that its instances will have, but it does not contain any specific data or state.

On the other hand, an instance object is a specific object created from a class object. An instance object has its own state (i.e., data), which can be different from other instances of the same class, but it shares the behaviour defined by the class.

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

**Ans-** In Python, the first argument in a class's method function is conventionally named “self”. This argument refers to the instance of the class that the method is being called on. It is not a keyword, and you can technically name this argument anything you want, but using “self” is a widely-accepted convention and makes the code more readable and understandable.

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

Ans- In Python, the ‘\_\_init\_\_’ method (short for "initialize") is a special method that is called automatically when an instance of a class is created. Its purpose is to initialize the attributes of the object and set their initial values.

The ‘\_\_init\_\_’ method is a constructor method that takes the instance object (‘self’) as its first argument, followed by any other arguments that are needed to initialize the object's attributes. Inside the \_\_init\_\_ method, we can set the initial values of the object's attributes using the dot notation (self.attribute\_name = initial\_value).

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

Ans- To create an instance of a class in Python, you need to follow these steps:

1. Define the class: Define the class with its attributes and methods.

2. Create an instance of the class: Create a variable and assign it to the class, followed by any arguments needed by the ‘\_\_init\_\_’ method to initialize the object's attributes.

3. Access attributes and methods: Once you have created an instance of the class, you can access its attributes and methods using the dot notation.

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

Ans- To create a class in Python, you need to follow these steps:

1. Define the class: Define the class with a name and any attributes and methods that are needed.
2. Define the constructor method: Define the ‘\_\_init\_\_’ method, which is a special method used to initialize the object's attributes.
3. Define other methods: Define any other methods that are needed for the class.

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

Ans- The superclasses of a class are the classes that the current class inherits from. In Python, a class can inherit from one or more classes, and these classes are known as the superclass or superclasses of the class.