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

Ans: Python is known for its support of OOP, and you can create classes and objects easily, making it a versatile language for developing object-oriented software. By using OOP in Python, you can create well-organized, maintainable, and extensible code, which is particularly useful for building complex applications and systems.

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

Ans: The attribute lookup order is designed to support multiple inheritance in Python, where a class can inherit from multiple base classes. The Method Resolution Order ensures that classes are searched in a specific order, allowing you to define and override attributes and methods in a consistent and predictable manner when dealing with complex class hierarchies.

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

Ans: class objects represent the blueprint or template for creating instance objects, while instance objects are specific instances created from a class with their own unique data and attributes. Class objects define the common structure and behavior, while instance objects hold individual data and state.

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 typically named self, although you can technically use any name you want for it (though it's a widely followed convention to use self). This argument is special because it represents the instance of the class on which the method is called. By convention, this argument is always the first one in the method definition.

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

Ans: The \_\_init\_\_ method in Python is a special method, also known as a constructor, that serves the purpose of initializing an object or instance of a class. It is automatically called when you create a new instance of a class.

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

Ans: To create a class instance in Python:

1. Define a class using the class keyword.
2. Instantiate the class by calling it like a function, passing any required arguments specified in the \_\_init\_\_ method.
3. You now have an instance of the class that you can work with.

Q7. What is the process for creating a class?

Ans: Using class keyword, defining class attributes, defining methods, initializing instances, extending and customizing classes.

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

Ans: In Python, you can define the superclasses (also known as base classes or parent classes) of a class by specifying them in the class definition using parentheses. Superclasses are the classes from which your class inherits attributes and methods. This concept is a fundamental part of inheritance in object-oriented programming.