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

ANSWER.

Python's OOP promotes code organization, reusability, maintainability, and scalability, making it a powerful paradigm for building complex software systems.

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

ANSWER.

In Python, when you access an attribute (such as a method or a variable) of an object, the interpreter performs an inheritance search to determine where to find the attribute. This search follows a specific order called the Method Resolution Order (MRO), which is determined by the inheritance hierarchy of classes.

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

ANSWER.

A class is a general template or blueprint, while an instance is a specific occurrence or realization of that template, with its own unique data and state.

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

ANSWER.

In most object-oriented programming languages like Python, the first argument in a class's method function is conventionally named `self`. This argument refers to the instance of the class itself when the method is called.

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

ANSWER.

The `\_\_init\_\_` method in Python is a special method used for initializing new objects created from a class. It stands for "initialize" and is also known as the constructor method. The purpose of the `\_\_init\_\_` method is to set up the initial state of an object by assigning values to its attributes or performing any necessary setup operations.

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

ANSWER.

Creating a class instance typically involves several steps, which may vary slightly depending on the programming language you're using, but I'll outline the general process:

1. Define the Class: First, you need to define a class. A class is like a blueprint or template for creating objects. It defines the properties (attributes) and behaviors (methods) that all instances of the class will have.

2. Instantiate the Class: To create an instance of a class, you use the class name followed by parentheses. This is known as instantiation. When you instantiate a class, memory is allocated to store the instance, and any necessary initialization code defined in the class constructor (often called `\_\_init\_\_` in many languages) is executed.

3. Initialization: During instantiation, you can pass arguments to the class constructor if it requires any parameters. These arguments are used to initialize the instance's attributes.

4. Instance Attributes: After instantiation, you can access the instance's attributes using dot notation. Attributes are variables associated with a specific instance of the class.

5. Instance Methods: You can also call methods defined in the class on the instance using dot notation. Methods are functions defined within the class that can operate on the instance's attributes.

Q7. What is the process for creating a class?

ANSWER.

Creating a class involves defining a blueprint or template for creating objects with similar properties and behaviors. Here's a general process for creating a class:

1. Define the Class: Use the `class` keyword followed by the class name to begin the class definition.

2. Define Attributes: Inside the class definition, define attributes (variables) that will store data associated with each instance of the class. These attributes are often initialized in the class constructor (`\_\_init\_\_` method).

3. Define Methods: Define methods (functions) inside the class to perform operations on the instance's attributes or perform other tasks related to the class.

4. Initialization (Optional): If you need to initialize attributes with specific values when an instance is created, define a constructor method (`\_\_init\_\_` in Python) to handle this initialization.

5. Instantiate Objects: Once the class is defined, you can create instances (objects) of the class by calling the class name followed by parentheses. This invokes the class constructor to initialize the instance.

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

ANSWER.

To define superclasses of a class, you employ inheritance. Inheritance is a fundamental concept in object-oriented programming where a class (subclass) can inherit attributes and methods from another class (superclass). Here's how you define superclasses for a class:

1. Create the Superclass: Define a class that contains attributes and methods which you want other classes to inherit.

2. Define Subclasses: Define subclasses that inherit from the superclass. In Python, you can do this by putting the superclass name inside parentheses after the subclass name in the class definition.

3. Access Inherited Attributes and Methods: Once a subclass inherits from a superclass, it gains access to all the attributes and methods defined in the superclass.