Q1. What is the relationship between classes and modules?

The relationship between classes and modules in Python is that a module can contain one or more classes. A module is a file containing Python definitions and statements, and it can include class definitions. Classes, on the other hand, are a way to bundle data and functionality together, and they can be defined within a module.

Q2. How do you make instances and classes?

Instances of a class are created by calling the class as if it were a function. This process is known as instantiation

Q3. Where and how should be class attributes created?

Class attributes should be created within the class block but outside any method. They are defined directly beneath the class header and are shared by all instances of the class.

Q4. Where and how are instance attributes created?

Instance attributes are created within the \_\_init\_\_ method of a class. They are specific to each instance of the class and are initialized during the creation of the instance.

Q5. What does the term "self" in a Python class mean?

In a Python class, the term "self" refers to the instance of the class. It is a conventional name for the first parameter of instance methods and is used to reference the instance itself. By convention, it is named "self," but you could use any name.

Q6. How does a Python class handle operator overloading?

Python classes handle operator overloading by defining special methods with double underscores (dunder methods or magic methods). For example, to overload the addition operator +, you can define the \_\_add\_\_ method in your class.

Q7. When do you consider allowing operator overloading of your classes?

Operator overloading is considered when you want to define custom behaviour for standard operators in your class. It allows instances of your class to use standard operators in a way that makes sense for your specific application.

Q8. What is the most popular form of operator overloading?

The most popular form of operator overloading is the \_\_add\_\_ method, which allows you to define the behaviour of the + operator. Other common forms include \_\_sub\_\_, \_\_mul\_\_, and so on, for different operators.

Q9. What are the two most important concepts to grasp in order to comprehend Python OOP code?

The two most important concepts to grasp in order to comprehend Python OOP code are:

Inheritance: Understanding how classes can inherit attributes and methods from other classes.

Encapsulation: Understanding how to encapsulate data and behaviour within a class, controlling access to attributes and methods.