Q1. What is the relationship between classes and modules?

Ans. 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. How do you make instances and classes?

Ans. For creating a class instance. we call a class by its name and pass the arguments which its \_\_init\_\_ method accepts. A class is defined using the class keyword followed by the class name and : operator after the class name, which allows you to continue in the next indented line to define class members.

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

Q3. Where and how should be class attributes created?

Ans. A class attribute is a Python variable that belongs to a class rather than a particular object. It is shared between all the objects of this class and it is defined outside the constructor function, \_\_init\_\_(self,...), of the class. The class attribute (class\_attr) is accessible as both a property of the class and as a property of objects, as it is shared between all of them.

Q4. Where and how are instance attributes created?

Ans. An instance attribute is a Python variable belonging to one, and only one, object. This variable is only accessible in the scope of this object and it is defined inside the constructor function, \_\_init\_\_(self,..) of the class. The instance\_attr is only accessible from the scope of an object.

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

Ans. The self parameter is a reference to the current instance of the class, and is used to access variables those belong to the class. By using the “self” we can access the attributes and methods of the class in python. It binds the attributes with the given arguments.

Q6. How does a Python class handle operator overloading?

Ans. The operator overloading in Python means to provide extended meaning beyond their predefined operational meaning. Such as, we use the "+" operator for adding two integers as well as joining two strings or merging two lists. We can achieve this as the "+" operator is overloaded by the "int" class and "str" class.

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

Ans. The purpose of operator overloading is to provide a special meaning of an operator for a user-defined data type.

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

Ans. The most popular and convenient form of operator overloading is the Addition (+) operator. It performs “Addition” on numbers whereas it performs “Concatenation” on strings.

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

Ans. Major OOP (object-oriented programming) concepts in Python include Class, Object, Method, Inheritance, Polymorphism, Data Abstraction, and Encapsulation.