Q1. What is the relationship between classes and modules?

A class is more of a unit, and a module is essentially a loose collection of stuff like functions, variables, or even classes.

In a public module, classes in the project have access to the functions and variables of the module. You don't have to specify the module name to address one.

Q2. How do you make instances and classes?

When users define a class, the variables they use within a class are the class variables, and the variables used within a class instance are the instance variables. Users do not use class variables as frequently as they use instance variables.

The Python instances of the class have instance variables. It indicates that every object or instance has its unique instance variables, i.e., the instance variables are different for each object or instance of a class.

Q3. Where and how should be class attributes created?

Class attributes are the variables defined directly in the class that are shared by all objects of the class.

Accessed using class name as well as using object with dot notation, e.g. classname.class\_attribute or object.class\_attribute

Changing value by using classname.class\_attribute = value will be reflected to all the objects.

Q4. Where and how are instance attributes created?

Instance attributes are attributes or properties attached to an instance of a class. Instance attributes are defined in the constructor.

Defined inside a constructor using the self parameter.

Accessed using object dot notation e.g. object.instance\_attribute

Changing value of instance attribute will not be reflected to other objects.

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

The self parameter is a reference to the current instance of the class, and is used to access variables that belongs to the class.

Q6. How does a Python class handle operator overloading?

The operator overloading in Python means 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. The user can notice that the same inbuilt operator or function is showing different behaviour for objects of different classes. This process is known as operator overloading.

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

Ensures that objects of a class behave consistently with built-in types and other user-defined types.

Makes it simpler to write code, especially for complex data types.

Allows for code reuse by implementing one operator method and using it for other operators.

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

The most frequent instance is the adding up operator ‘+’, where it can be used for the usual addition and also for combining two different strings. As mentioned on top, the plus symbol’s practice in dissimilar forms is the largest classic example of the operator level overloading process.

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

Inheritance is the capability of one class to derive or inherit the properties from another class. The class that derives properties is called the derived class or child class and the class from which the properties are being derived is called the base class or parent class.

Polymorphism simply means having many forms. For example, we need to determine if the given species of birds fly or not, using polymorphism we can do this using a single function.