**Q1. What is the relationship between classes and modules?**

**Ans:** Modules are collections of methods and constants. They cannot generate instances. Classes may generate instances (objects), and have per-instance state (instance variables).

Classes in python are templates for creating objects. They contain variables and functions which define the class objects. At the same time, modules are python programs that can be imported into another python program. Importing a module enables the usage of the module's functions and variables into another program.

**Q2. How do you make instances and classes?**

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

An object may be varied in a number of ways. In class-based programming, objects are created from classes by subroutines called constructors, and destroyed by destructors. An object is an instance of a class, and may be called a class instance or class object; instantiation is then also known as construction.

**Q3. Where and how should be class attributes created?**

**Ans:** Class attributes are the variables defined directly in the class that are shared by all objects of the class. Instance attributes are attributes or properties attached to an instance of a class. Instance attributes are defined in the constructor.

Class attributes are attributes which are owned by the class itself. They will be shared by all the instances of the class. Therefore they have the same value for every instance. We define class attributes outside all the methods, usually they are placed at the top, right below the class header.

**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.

Instance attributes are defined in the constructor. Defined directly inside a class. Defined inside a constructor using the self parameter.

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

**Ans:** self represents the instance of the class. By using the “self” keyword we can access the attributes and methods of the class in python. It binds the attributes with the given arguments.

The self variable is used to represent the instance of the class which is often used in object-oriented programming. It works as a reference to the object. Python uses the self parameter to refer to instance attributes and methods of the class.

**Q6. How does a Python class handle operator overloading?**

**Ans:** To perform operator overloading, Python provides some special function or magic function that is automatically invoked when it is associated with that particular operator. For example, when we use + operator, the magic method \_\_add\_\_ is automatically invoked in which the operation for + operator is defined.

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

**Ans:** Consider that we have two objects which are a physical representation of a class (user-defined data type) and we have to add two objects with binary '+' operator it throws an error, because compiler don't know how to add two objects. So we define a method for an operator and that process is called operator overloading.

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

**Ans:** A very popular and convenient example is the Addition (+) operator. Just think how the '+' operator operates on two numbers and the same operator operates on two strings. 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:**

1. Class and Object
2. inheritance and polymorphism