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

Ans: Classes in python act as a blueprint based on which objects are created. Objects are the very basis for object-oriented programming. The objects are the real-world entities, and class acts as a template that defines those objects.

These real-world entities have behavior associated with them, and classes define that behavior. A class contains variables and functions which act on the objects.

**To define a class, we use the keyword class to do so. A class can be defined in the following manner:**

class class\_name:

  var = 'Class variable'

  def class\_function1(self):

    #function1 body

Modules in Python are files with a .py extension using which we can reuse elements inside that file. When we create a python program, the program may contain inside it functions, variables, and even classes.

If we want to reuse the same piece of function code or the same class, rewriting it would make our code redundant and repetitive. Instead, we can import that entire file as a module into another program.

Doing so makes our code reusable and improves its readability. An entire project can then be broken down into smaller sections, and thus the code becomes more manageable.

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

Ans :

USE THE CLASS NAME TO CREATE A NEW INSTANCE

Call ClassName() to create a new instance of the class ClassName.

class C:

pass

an\_instance = C()

print(an\_instance)

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

Ans : Class attributes belong to the class itself they will be shared by all the instances. Such attributes are defined in the class body parts usually at the top, for legibility.

Class Attribute :

A class attribute is a Python Variable that belongs to a class rather than a particular object. This is shared between all other objects of the same class and is defined outside the constructor function \_\_init\_\_(self,…), of the class.

**Q4. Where and how are instance attributes created?**

Ans : Instance Attribute

An instance attribute is a Python variable belonging to only one object. It is only accessible in the scope of the object and it is defined inside the constructor function of a class. For example, \_\_init\_\_(self,..).

**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.

**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 : In order to develop robust and well-designed software products with Python, it is essential to obtain a comprehensive understanding of OOP. In this article, we will elaborate on two key concepts of OOP which are inheritance and polymorphism.