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

**Ans:-** Classes in python act as a blueprint based on which objects are created. The objects are the real-world entities and class acts as a template that defines those objects.

A module is used to reuse a given piece of code inside another program. Modules in Python are files with a *.****py*extension** and we can import that entire file as a module into another program.

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

**Ans:- Instances:-** Instance methods are defined inside a class, and it is pretty similar to defining a regular function. Use the def keyword to define an instance method in Python. Use self as the first parameter in the instance method when defining it.

**Class:-** Classes in python act as a blueprint based on which objects are created .While defining a new class, we also create a new class object that contains all the attributes (variables and methods) related to that class.

**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 and  is defined outside the constructor function, \_\_init\_\_(self,...) , of the class.

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

**Ans:-**  Instance attributes are defined in the constructor and  defined directly inside a class using the self parameter. Instance attributes are defined in the \_\_init\_\_() function.

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

**Ans:-** SELF represents the instance of class. This keyword allows you to access variables, attributes, and methods of a defined class in Python. The self parameter doesn't have to be named “self,” as you can call it by any other name.

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

**Ans:-** In Python, overloading is achieved by overriding the method which is specifically for that operator, in the user-defined class. For example, \_\_add\_\_(self, x) is a method reserved for overloading + operator, and \_\_eq\_\_(self, x) is for overloading == .

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

**Ans:-** When one or both operands are of a user-defined class or structure type, operator overloading makes it easier to specify user-defined implementation for such operations.

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

**Ans:-** A very popular and convenient example is the **Addition (+) operator**.

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

**Ans:-** Both inheritance and polymorphism are fundamental concepts of object oriented programming. These concepts help us to create code that can be  used for designing robust, flexible, and easy-to-maintain software.