Q1. What is the relationship between classes and modules?
Answer: In python a class contains the blueprint of an object, its attributes, functions etc, whereas a module is used to reuse a given piece of code in another program

Q2. How do you make instances and classes?
Answer: Call ClassName() is to create a new instance of the class ClassName . To pass parameters to the class instance, the class must have an __init__() method. Pass the parameters in the constructor of the class.

Create a Class. To create a class, use the keyword class

Q3. Where and how should class attributes be created?
Answer: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

Inside a class, you should qualify all references to class attributes with the class name; for example, → **MyClass. attr1**

Q4. Where and how are instance attributes created?
Answer: Instance attributes are defined in the constructor. Defined directly inside a class. Defined inside a constructor using the self parameter.

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

Q5. What does the term "self" in a Python class mean?
Answer: self represents the instance of the class that points to 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?
Answer: When we use an operator on user defined data types then automatically a special function or magic function associated with that operator is invoked. Changing the behavior of an operator is as simple as changing the behavior of a method or function. You define methods in your class and operators work according to that behavior defined in methods. When we use + operator, the magic method __add__ is automatically invoked in which the operation for + operator is defined. There by changing this magic method’s code, we can give extra meaning to the + operator.

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

Answer: When the scope of an operator does not cover the intended function of the user we can consider operator overloading. 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 the compiler doesn'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?
Answer: A very popular and convenient example is the Addition (+) operator

```
class A:
    def __init__(self, a):
        self.a = a

    # adding two objects
    def __add__(self, o):
        return self.a + o.a
ob1 = A(1)
ob2 = A(2)
ob3 = A("Geeks")
ob4 = A("For")

print(ob1 + ob2)
print(ob3 + ob4)
```

Q9. What are the two most important concepts to grasp in order to comprehend Python OOP code?
Answer: In order to develop robust and well-designed software products with Python, it is essential to obtain a comprehensive understanding of OOP and two key concepts of OOP are inheritance and polymorphism.