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

Answer: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. Classes may generate instances (objects), and have per-instance state (instance variables). Modules may be mixed in to classes and other modules. The mixed in module's constants and methods blend into that class's own, augmenting the class's functionality. Classes, however, cannot be mixed in to anything.

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

Answer:
To create instances of a class, you call the class using class name and pass in whatever arguments its **init** method accepts.

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

Answer:
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.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.

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

Answer:
Instance attributes are attributes or properties attached to an instance of a 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?

Answer: self represents the instance of the class. By using the “self” we can access the attributes and methods of the class in python. It binds the attributes with the given arguments. The reason you need to use self. is because Python does not use the @ syntax to refer to instance attributes.

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.

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

Answer: the + operator will perform arithmetic addition on two numbers, merge two lists, or concatenate two strings. This feature in Python that allows the same operator to have different meaning according to the context is called operator overloading.

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

Answer: Operator overloading is the process of using an operator in different ways depending on the operands. You can change the way an operator in Python works on different data-types. 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?

Answer: we will elaborate on two key concepts of OOP which are inheritance and polymorphism. Both inheritance and polymorphism are key ingredients for designing robust, flexible, and easy-to-maintain software. These concepts are best explained via examples. Let's start with creating a simple class