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

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

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

* An Instance on the other hand is simply a variation of an object created from a class. You create an object variant (Instance) using a constructor which is a method within a class specifically defined for this purpose. 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?**

* 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. Unlike class attributes, instance attributes are not shared by objects.

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

* 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?**

* 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?**

* Python class handle operator overloading in python by user defined function using “None” keyword as default parameter. Code explanation: The first parameter of “add” method is set to none. This will give us the option to call it with or without a parameter.

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

* 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. This makes user-defined types more similar to the basic primitive data types in terms of behaviour.

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

* The most common use of the addition operator '+' is for concatenating and combining two different strings. As mentioned on top, the plus symbol's practice in dissimilar forms is the largest classic example of the operator-level overloading process

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

* Both inheritance and polymorphism are fundamental concepts of object oriented programming. These concepts help us to create code that can be extended and easily maintainable