Q1. What is the relationship between classes and modules?

**a class is used to define a blueprint for a given object, whereas a module is used to reuse a given piece of code inside another program**.

Q2. How do you make instances and classes?

**Call ClassName() 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.**

Q3. Where and how should be class attributes created?

**Inside a class, we should qualify all references to class attributes with the class name; for example, MyClass. In the class body parts usually at the top.**

Q4. Where and how are instance attributes created?

**Inside the constructor, call the class using class name and pass in whatever arguments its \_\_init\_\_ method accepts**

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

**The self parameter is a reference to the current instance of the class, and is used to access variables that belongs to the class.**

Q6. How does a Python class handle operator overloading?

**Python operators work for built-in classes. But the same operator behaves differently with different types. For example, the + operator will perform arithmetic addition on two numbers, merge two lists, or concatenate two strings.**

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

**At least one of the operands must be a user-defined object. Only existing operators can be overloaded**

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

**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?

**inheritance and polymorphism**.