Q1. What is the relationship between classes and modules?

The difference between a class and a module in python is that **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?

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?

A class attribute is shared by all instances of the class. To define a class attribute, you place it outside of the \_\_init\_\_() method.

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**

Q6. How does a Python class handle operator overloading?

Operator Overloading means **giving extended meaning beyond their predefined operational meaning**. For example operator + is used to add two integers as well as join two strings and merge two lists. It is achievable because '+' operator is overloaded by int class and str class.

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

By overloading standard operators on a class, **you can exploit the intuition of the users of that class**. This lets users program in the language of the problem domain rather than in the language of the machine.

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

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?

**inheritance and polymorphism**.s