Q1. What is the relationship between classes and modules?

1. 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?

1. **Instantiating a Class** :   
   The new operator requires a single, postfix argument: a call to a constructor. The name of the constructor provides the name of the class to instantiate. The constructor initializes the new object. The new operator returns a reference to the object it created.
2. In Python, a class can be created **by using the keyword class, followed by the class name**.

Q3. Where and how should be class attributes created?

1. 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**

Q4. Where and how are instance attributes created?

1. 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?

1. 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

Q6. How does a Python class handle operator overloading?

1. 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?

1. The purpose of operator overloading is **to provide a special meaning of an operator for a user-defined data type**. With the help of operator overloading, you can redefine the majority of the operators. You can also use operator overloading to perform different operations using one operator.).

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

1. A very popular and convenient example is the **Addition (+) operator**. Just think how the '+' operator operates on two numbers and the same operator operates on two strings. It performs “Addition” on numbers whereas it performs “Concatenation” on strings

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

1. The most important concept of Object-Oriented Programming for me is not inheritance, encapsulation, or polymorphism, but **the notion of keeping the data and the related code close**. The point of this is being able to keep your program structured and traceable. All other properties of OOP are related to this concept.