Q1. What is the relationship between classes and modules?  
Ans: Modules are collections of methods and constants. They cannot generate instances. Classes may generate instances (objects), and have per-instance state (instance variables). Modules may be mixed in to classes and other modules.

Q2. How do you make instances and classes?  
Ans:

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?  
ans:  
In Python, class attributes are defined directly within the class definition and are shared among all instances of the class. They can be accessed using the class name and through an instance of the class.

Q4. Where and how are instance attributes created?

Ans:   
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's defined inside the constructor function, \_\_init\_\_(self,..) of the class.

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

Ans:  
Operator overloading provides the ability to override the functionality of a built-in operator in user-defined classes. Operator overloading in Python is done using special functions or magic methods. Python provides a predefined list of special functions or magic methods for different operators.

Q7. When do you consider allowing operator overloading of your classes?  
ans:  
Operator overloading is mostly useful when you're making a new class that falls into an existing "Abstract Base Class" (ABC) -- indeed, many of the ABCs in standard library module collections rely on the presence of certain special methods (and special methods, one with names starting and ending with double underscores

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

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?  
ans:  
Conclusion. 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