Q1. What is the relationship between classes and modules?

Ans: The difference between a class and a module 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?

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

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: 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?

Ans: To perform operator overloading, Python provides some special function or magic function that is automatically invoked when it is associated with that particular operator.

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

Ans: Consider that we have two objects which are a physical representation of a class (user-defined data type) and we have to add two objects with binary '+' operator it throws an error, because compiler don't know how to add two objects. So we define a method for an operator and that process is called operator overloading.

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

Ans: The most frequent instance is the adding up operator '+', where it can be used for the usual addition and also for combining two different strings.

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

Ans: 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.