Q1. What is the relationship between classes and modules?

Ans: A module in python is simply a way to organize the code, and it contains either python classes or just functions. If you need those classes or functions in your project, you just import them. For instance, the math module in python contains just a bunch of functions, and you just call those needed ( math. sin ).

Classes may generate instances (objects), and have per-instance state (instance variables). A class may inherit from another class, but not from a module. A module may not inherit from anything.

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.

A Class is like an object constructor, or a "blueprint" for creating objects. To create a class, use the keyword class. To understand the meaning of classes we have to understand the built-in \_\_init\_\_() function.

All classes have a function called \_\_init\_\_(), which is always executed when the class is being initiated. Use the \_\_init\_\_() function to assign values to object properties, or other operations that are necessary to do when the object is being created.

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 attributes for storing class constants, track data across all instances, and setting default values for all instances of the class. Use class\_name.class\_attribute or object\_name.class\_attribute to access the value of the class\_attribute.

Q4. Where and how are instance attributes created?

Ans: We add instance attributes usually by defining the constructor method “\_\_init\_\_”. Within \_\_init\_\_ we pass the first parameters self which represents an object from a class that is currently being defined, and usually, we attach instance attributes to these objects using the dot operator.

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. It does not have to be named self , you can call it whatever you like, but it has to be the first parameter of any function in the class.

Q6. How does a Python class handle operator overloading?

Ans: 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?

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. As mentioned on top, the plus symbol's practice in dissimilar forms is the largest classic example of the operator level overloading process.

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.