Q1. What is the relationship between classes and modules?

|  |  |
| --- | --- |
| classes | modules |
| Classes are about objects; it can hold its state (by keeping track of instance variables) and be duplicated as many times as you want. | Modules are about providing methods that you can use across multiple classes. Modules are about functions. It don’t instantiate modules(e.g. Math module) just use methods in them.Module is like library of methods. |
|  |  |

Q2. How do you make instances and classes?

The class statement creates a new class definition. The name of the class immediately follows the keyword class followed by a colon

class Person:  
 def \_\_init\_\_(self, name, age):  
 self.name = name  
 self.age = age  
  
person1 = Person("Alice", 25)  
  
person2 = Person("Bob", 30)

Q3. Where and how should be class attributes created?

lass 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, for legibility.

Q4. Where and how are instance attributes created?

Unlike class attributes, instance attributes are not shared by objects. Every object has its own copy of the instance attribute. They are created in **init** method.

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

Self represents the instance of the class. By using the “self” keyword we can access the attributes and methods of the class in python. It binds the attributes with the given arguments. The reason you need to use self. is because Python does not use the @ syntax to refer to instance attributes. Python decided to do methods in a way that makes the instance to which the method belongs be passed automatically, but not received automatically: the first parameter of methods is the instance

Q6. How does a Python class handle operator overloading?

Python allows for operator overloading, which means that a class can define its own behavior for built-in operators, such as "+", "-", "\*", "/", and so on.

To overload an operator in a Python class, you need to define a special method with a specific name that corresponds to the operator you want to overload.

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

Operator overloading can be a powerful tool for creating expressive and easy-to-read code, but it can also lead to confusion and unexpected behavior if used improperly.

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

popular form of operator overloading is of addition (+) operator. When two integers are passed to + operator, it will return the sum of two integers. When two strings are passed to + operator, it will return concatenation of two strings.

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

1. Inheritance.
2. Polymorphism.