Q1. What is the relationship between classes and modules?

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?

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?

Class attributes are the variables defined directly in the class that are shared by all objects of the class.

Accessed using class name as well as using object with dot notation, e.g. classname.class\_attribute or object.class\_attribute

class Student:

count = 0

def \_\_init\_\_(self):

Student.count += 1

Q4. Where and how are instance attributes created?

|  |  |
| --- | --- |
|  | Defined inside a constructor using the self parameter. |

Accessed using object dot notation e.g. object.instance\_attribute

class Student:

def \_\_init\_\_(self, name, age):

self.name = name

self.age = age

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

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. The reason you need to use self. is because Python does not use the @ syntax to refer to instance attributes.

Q6. How does a Python class handle operator overloading?

**Operator Overloading** means giving extended meaning beyond their predefined operational meaning. For example operator + is used to add two integers as well as join two strings and merge two lists. It is achievable because ‘+’ operator is overloaded by int class and str class. You might have noticed that the same built-in operator or function shows different behavior for objects of different classes, this is called *Operator Overloading*.

print(1 + 2)

# concatenate two strings

print("Geeks"+"For")

# Product two numbers

print(3 \* 4)

# Repeat the String

print("Geeks"\*4)

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

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.

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

A very popular and convenient example is the Addition (+) operator.

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

Inheritance and polymorphism are key ingredients for designing robust, flexible, and easy-to-maintain software