**Q1. What is the relationship between classes and modules?**

**Answer:** Modules are files present inside a package, whereas a class is used to encapsulate data and functions together inside the same unit.

**Q2. How do you make instances and classes?**

**Answer:** The class statement creates a new class definition. The name of the class immediately follows the keyword class followed by a colon as follows −

class ClassName:

'Optional class documentation string'

class\_suite

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?**

**Answer:** Class attributes are attributes which are owned by the class itself. They will be shared by all the instances of the class. Therefore, they have the same value for every instance. We define class attributes outside all the methods, usually they are placed at the top, right below the class header.

Example:

class A:

a = "I am a class attribute!"

x = A()

y = A()

x.a

#### OUTPUT:

'I am a class attribute!'

y.a

#### OUTPUT:

'I am a class attribute!'

A.a

#### OUTPUT:

'I am a class attribute!

**Q4. Where and how are instance attributes created?**

**Answer:** [Class attributes](https://www.geeksforgeeks.org/g-fact-34-class-or-static-variables-in-python/) 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.

|  |
| --- |
| # Write Python code here  class sampleclass:      count = 0     # class attribute        def increase(self):          sampleclass.count += 1    # Calling increase() on an object  s1 = sampleclass()  s1.increase()  print(s1.count)    # Calling increase on one more  # object  s2 = sampleclass()  s2.increase()  print(s2.count)    print(sampleclass.count) |

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

**Answer:** ‘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.

**Q6. How does a Python class handle operator overloading?**

**Answer**: To perform operator overloading, Python provides some special function or magic function that is automatically invoked when it is associated with that particular operator. For example, when we use + operator, the magic method \_\_add\_\_ is automatically invoked in which the operation for + operator is defined.

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

**Answer**: 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. We can overload all existing operators but we can’t create a new operator.

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

**Answer**: 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?**

**Answer**: inheritance and polymorphism.