Q1. What is the relationship between classes and modules?

ANS: **A class can have its own instance, but a module cannot be instantiated**. We use the ‘class’ keyword to define a class, whereas to use modules, we use the ‘import’ keyword. We can inherit a particular class and modify it using inheritance. But while using modules, it is simply a code containing variables, functions, and classes.

Q2. How do you make instances and classes?

ANS: 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)

Q3. Where and how should be class attributes created?

ANS: As we saw earlier, we created class attributes by declaring them and assigning a value to them**inside the class, outside of the methods of that class.**

Q4. Where and how are instance attributes created?

ANS: [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.

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

ANS: 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.

Q6. How does a Python class handle operator overloading?

ANS: **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*.

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

ANS: Overall, overloading boolean operators in a custom class can make your code more readable, consistent, concise, expressive, and functional. However, it’s important to use operator overloading judiciously and only**when it makes sense for the semantics of your class**.

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: Apart from this, we have also understood concepts such as**instances and instantiating**, which are core to OOPs programming in Python. Next, you will need to understand the principles that govern these instances and classes such as inheritance, polymorphism, abstraction, and encapsulation.