Q1. What is the relationship between classes and modules?

1. To define a class we use class keyword.
2. Module is the combination of classes, functions and other modules
3. We can modify the attributes of the class by using inheritance. But we can simply use the codes written in a module by importing it.

Q2. How do you make instances and classes?

1. To make classes we use class keyword.
2. To make instance of a class you need to define \_\_init\_\_() method inside class. Then create object by passing the arguments it’s constructor takes then refer it to an instance.

Eg :

class Instance:

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

name = name

i = Instance()

print(i.name)

Q3. Where and how should be class attributes created?

Class attributes belong to class itself, they will be shared by all the instances. They are defined at the class body parts.

Q4. Where and how are instance attributes created?

Instance attributes are defined inside constructor which is defined directly inside a class using parameter self.

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

Self is a reference variable which indicates current class instance/object. And it is used to access variables which belong to the particular class, inside the class.

Q6. How does a Python class handle operator overloading?

In python + operator works as overloaded operator. It performs addition when we use it with int class. It performs concatenation when we use it with string and list class. To handle operator overloading python has inbuilt magic functions. (eg: \_\_add\_\_(self, other),\_\_sub\_\_(self, other), \_\_mul\_\_(self, other), \_\_truediv\_\_(self, other),\_\_floordiv\_\_(self, other), \_\_mod\_\_(self, other), \_\_pow\_\_(self, other), \_\_rshift\_\_(self, other), \_\_lshift\_\_(self, other), \_\_and\_\_(self, other), \_\_or\_\_(self, other), \_\_xor\_\_(self, other))

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

Allowing operator overloading means using operators (+,-,\*,<,>,&,|,^) for different classes to perform different operations. We will allow operator overloading for the classes when we are dealing with same type classes (int-int, str-str, list-list) to perform either addition or concatenation operation.

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

The most popular form of operator loading is +. Which is used for both addition and concatenation.

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

The most important concepts to grasp Python OOP code are

1. Inheritance
2. Polymorphism

Inheritance :

Acquiring the properties of one class to another class is the concept of inheritance.

It is used for code reusability and extendability.

Eg:

class Employee:

def \_\_init\_\_(self, eno, ename, esal):

self.eno = eno

self.ename = ename

self.esal = esal

def display(self):

print('Employee Number : ', self.eno)

print('Employee Name : ', self.ename)

print('Employee Salary : ', self.esal)

e = Employee(1118, 'Pragati', 100000)

e.display()