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

Modules are just a way to organize our code. Modules contain either classes or functions which we use in our code by importing the whole module or specific class or function. When we import any module, it means we import the whole .py file in our code.

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

We create classes with keyword ‘class’ and name of class then constructor of the class with all positional and keyword arguments then define all methods related to the class with pointer used in constructor.

class SuperHero:

def \_\_init\_\_(self, name, powers, costume\_col, gender = ‘male’):

self.name = name

--------------other attributes

def method\_name(self, attributes):

Bat = SuperHero(‘Batman’, ’rich’, ’black’)

Here Bat is an instance of class SuperHero. So we create instances by providing all positional and keyword arguments.

**Q3. Where and how should be class attributes created?**

class SuperHero:

def \_\_init\_\_(self,name,powers,costume\_col,gender = ‘male’):

self.name = name

--------------other attributes

def method\_name(self,attributes):

Attributes should be created at the time of creating class and its constructor.

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

Bat = SuperHero(‘Batman’, ’rich’, ’black’)

Bat\_vehicle = ‘Batmobile’

We create instance attributes after defining instance with name of instance dot name of attributes.

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

While creating constructor of any class the first attributes by default will be a pointer. Generally, we use ‘self’ as a pointer. A pointer in a class is used to connect attributes and methods to the class.

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

In python a specific operator behaves differently on different class objects. For e.g., ‘+’ operator do addition operation on int and float type objects while it will do concatenation, if we use it on two str objects. We can define its operation in our custom class with method ‘\_\_init\_\_’. So in this case we overloaded ‘+’ operator.

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

When we use common operators like ‘+’, ‘-’, ‘\*’ etc. on our custom class object then it will throw an error ‘TypeError: unsupported operand type(s) for +’. This happens because we didn’t define ‘\_\_add\_\_’, ‘\_\_sub\_\_’ etc. methods in our class as for int it is defined to add numbers, for str it is defined to concatenate. We also need to define this to perform ‘+’ operation on our class object.

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

‘+’ is the most popular form of operator overloading.

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

Polymorphism and Inheritance are two most important concepts.