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

A module is way to organize related code and it contains classes, methods and other code.

A class is a blueprint that lays the foundation for OOP. It contains the properties (attributes) and methods as a single unit and several instances of this class can be created with individual values for the attributes.

Example :

Student class contains attributes and methods for creating student objects. All the classes and methods related to data cleaning can be placed in a module data cleaning and imported into our file. Then we can call those methods and make use of it.

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

The definition of class is specified in the class using ‘class’ keyword followed by class name.

class <class\_name>:

<code> / pass

Instance is created by specifying class name and passing values for the class constructor parameters if any.

<instance\_name> = <class\_name>([<constructor parameters]>)

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

Class attributes are created outside the constructor of the class(\_\_init\_\_ method). It is specified in the indented block following the class name. Only one copy of class attributes are created and are shared by all the instances of that class.

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

Instance attributes are created inside the constructor of the class(\_\_init\_\_ method). Each instance have its own specific copy of the instance attribute.

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

“self” usually refers to the particular instance of the class that is being accessed at the moment. It acts as a handle to the instance.

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

Operator overloading allows extending the original meaning of an operator. Python supports overloading all existing operators, however, it doesn’t allow creation of a new operator.

To overload an operator, we write our own version of the default magic function associated with that operator.

For example, to overload + operator, we define our modified code inside the corresponding magic function ‘\_\_add\_\_’. When this operator is used, it automatically invokes this magic function.

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

1. To overwrite the pre existing way in which the operator behaves
2. To handle the situations where the existing magic function throws an error as it is unsupported.
3. To extend the capability of an operator to work with user defined objects

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

Overloading the + operator to work with use defined objects

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

1. Inheritance: It allows creating a child class inheriting the features of parent class. Example: creating a Manager class based on Employee class.
2. Polymorphism: A method behaves differently in different context. Depending on which object called the method, the corresponding definition gets executed.

Note: To understand the above 2 concepts, one must master creating classes and objects, which lays the foundation for creating OOP based programs.