Q1. What is the relationship between classes and modules?

- Class is used to define a blueprint for a given object, whereas a module is used to reuse a given piece of code inside another program.
- 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.

Q2. How do you make instances and classes?

- The following python syntax defines a class:

```
class ClassName(base_classes):
statements
```

- Instances syntax:

```
Instance_name = ClassName()
```

Q3. Where and how should be class attributes created?

- A Python class attribute is an attribute of the class, rather than an attribute of an *instance* of a class.
- Example:

```
class MyClass(object):
    class_var = 1

    def __init__(self, i_var):
        self.i_var = i_var
```

- Here class_var is the class attribute.

Q4. Where and how are instance attributes created?

- Attributes created in __init__ method are called instance attributes Example:

```
class MyClass(object):
    class_var = 1

    def __init__(self, i_var):
        self.i_var = i_var
```

- Here i_var is the Instance attribute.

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

- "self" represents the instance of the class. By using the "self" keyword we can access the attributes and methods of the class in python.

Q6. How does a Python class handle operator overloading?

- feature in Python that allows the same operator to have different meaning according to the context is called operator overloading.

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

- 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.

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

- 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?

- There are four fundamental concepts of Object-oriented programming – Inheritance, Encapsulation, Polymorphism, and Data abstraction. It is very important to know about all of these in order to understand OOPs.