Q1. What is the relationship between classes and modules?

A **class** can be instantiated but a **module** cannot. A **module** will never be anything other than a library **of** methods. A **class** can be so much more -- it can hold its state (by keeping track **of** instance variables) and be duplicated as many times as you want.

Q2. How do you make instances and classes?

To **create instances** of a **class**, you call the **class** using **class** name and pass in whatever arguments its \_\_init\_\_ method accepts.

Q3. Where and how should be class attributes created?

**Class attributes are attributes** which **are** owned by the **class** itself. They **will** be shared by all the instances of the **class**. Therefore they have the same value for every instance. We define **class attributes** outside all the methods, usually they **are** placed at the top, right below the **class** header.

Q4. Where and how are instance attributes created?

An **instance attribute** is a Python variable belonging to only one object. It is only accessible in the scope of the object and it is defined inside the constructor function of a class.

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

The **self** parameter **is** a reference to the current instance of the **class**, and **is** used to access variables that belongs to the **class**.

Q6. How does a Python class handle operator overloading?

**Python operators** work for built-in **classes**. ... For **example**, the + **operator will** perform arithmetic addition on two numbers, merge two lists, or concatenate two strings. This 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. We can overload all existing operators but we can’t create a new operator. To perform operator overloading, Python provides some special function or magic function that is automatically invoked when it is associated with that particular operator. For example, when we use + operator, the magic method \_\_add\_\_ is automatically invoked in which the operation for + operator is defined.

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

Add operator is the most popular operator overloading

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

inheritance and polymorphism are fundamental concepts of object oriented programming. These concepts help us to create code that can be extended and easily maintainable.