Q1. What is the meaning of multiple inheritance?

Multiple inheritance is a feature in object-oriented programming languages that allows a class to inherit attributes and behaviors from more than one parent class. In other words, a class can derive features from multiple base classes simultaneously. This allows for code reuse and the creation of complex class hierarchies.

Q2. What is the concept of delegation?

Delegation is a design principle in object-oriented programming where an object passes on a specific task or responsibility to another object. Instead of inheriting behavior directly, an object delegates the task to another object that is specialized in performing that task. This allows for better code organization, reusability, and separation of concerns.
In delegation, an object that delegates a task retains control over the process and can coordinate or modify the behavior of the delegated task. This approach promotes a more modular and flexible design by allowing objects to collaborate without creating deep inheritance hierarchies.
Delegation is often used to achieve composition over inheritance, where the behavior of a class is composed by combining smaller, more specialized classes. This approach can lead to more maintainable and extensible code, as changes to one part of the system are less likely to impact other parts.

Q3. What is the concept of composition?

Composition is a design principle in object-oriented programming where complex objects are created by combining simpler objects. It is an alternative to inheritance for structuring classes and building relationships between them.
In composition, a class contains instances of other classes as member variables. This allows the class to reuse the behavior of the contained objects without inheriting their implementation details. Composition promotes a "has-a" relationship between classes, where an object is composed of one or more other objects.

Q4. What are bound methods and how do we use them?

A bound method is a method that is associated with an instance of a class. When you access a method on an instance, Python automatically binds the instance to the method, providing it as the first argument (self). This binding allows the method to operate on the instance's data and attributes.
Bound methods are a fundamental concept in object-oriented programming, allowing objects to interact with their own state and perform actions that are specific to their characteristics. They encapsulate behavior along with the data, promoting the principles of encapsulation and data abstraction

Q5. What is the purpose of pseudoprivate attributes?

Pseudoprivate attributes are a naming convention used to make instance variables within a class more "private." This means that the attributes are intended to be used only within the class they are defined in and not directly accessed from outside the class.
The purpose of pseudoprivate attributes is to provide a level of encapsulation, ensuring that certain attributes are not accidentally overridden or accessed by external code. However, it's important to note that this is only a convention and doesn't provide true privacy or security; the attributes can still be accessed if you know their mangled names.