Q1. What is the meaning of multiple inheritance?

Multiple inheritance refers to a programming language feature where a class can inherit attributes and behaviors from multiple parent classes. It allows a derived class to inherit characteristics from multiple base classes, enabling code reuse and creating complex class hierarchies. However, it can also introduce challenges such as ambiguity and conflicts when two or more base classes define the same attribute or behavior.

Q2. What is the concept of delegation?

Delegation is a programming concept where an object forwards a task or responsibility to another object. Instead of implementing the task itself, the delegating object assigns it to another object, known as the delegate, to handle the task. Delegation promotes code reuse, modular design, and separation of concerns by allowing objects to collaborate and distribute responsibilities effectively. It is commonly used in object-oriented programming to achieve composition over inheritance.

Q3. What is the concept of composition?

The concept of composition in programming refers to the practice of building complex objects or structures by combining simpler or smaller objects. It involves creating an object that contains other objects as its components or parts. Composition allows for code reuse, modularity, and flexibility, as the composed object can be easily modified by adding, removing, or replacing its components. It promotes a "has-a" relationship between objects, where one object possesses or is composed of other objects.

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

Bound methods in Python are methods that are associated with a specific instance of a class. They are automatically passed the instance as the first argument when called. By using dot notation (instance.method()), we can invoke bound methods, which operate on the instance's data. This allows for object-oriented programming and the manipulation of specific instances within a class.

Q5. What is the purpose of pseudoprivate attributes?

The purpose of pseudoprivate attributes is to indicate that an attribute in a class or object should be treated as private, even though it is not enforced by the language. By using a naming convention, such as adding a double underscore prefix (e.g., "\_\_attribute"), developers signal that the attribute is intended for internal use and should not be accessed directly from outside the class or object.