ASSIGNMENT - 5

Q1. What is the meaning of multiple inheritance?

Ans: Multiple inheritance in Python refers to a class that inherits attributes and methods from more than one parent class. This allows a subclass to inherit from multiple superclasses. While Python supports multiple inheritance, it can lead to complexities in code maintenance and the potential for conflicts when methods or attributes with the same name exist in different parent classes.

Q2. What is the concept of delegation?

Ans: Delegation is a design principle in which one object forwards (or delegates) a method call or responsibility to another object. Instead of directly implementing functionality, an object delegates tasks to other objects it contains, allowing for better code organization and separation of concerns.

Q3. What is the concept of composition?

Ans: Composition is a design principle that involves constructing a class using references to other objects. It allows for building complex functionalities by combining simpler, reusable components. Rather than inheriting behavior, a class using composition contains instances of other classes and uses their functionalities to achieve its own behavior.

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

Ans: Bound methods are methods that are already associated with an instance of a class. When you access a method from an instance of a class (e.g., instance.method()), Python automatically binds the method to the instance. Bound methods pass the instance (i.e., self) as the first argument implicitly. You can call bound methods directly on instances, and they automatically receive the instance as the first argument.

Q5. What is the purpose of pseudoprivate attributes?

Ans: In Python, pseudoprivate attributes are created by prefixing attribute names with double underscores (e.g., \_\_attribute). They are not truly private but rather undergo name mangling (i.e., the interpreter alters the attribute name to avoid clashes in subclasses). The purpose of pseudoprivate attributes is to minimize accidental access or overriding by subclasses or external code, helping to avoid unintentional conflicts and maintain encapsulation within a class.