**Q1. What is the meaning of multiple inheritance?**

**ANS:-**

**When a class is derived from more than one base class it is called multiple Inheritance. The derived class inherits all the features of the base case.**

**Q2. What is the concept of delegation?**

**ANS:-**

**Delegation is an object oriented technique (also called a design pattern). Let's say you have an object x and want to change the behaviour of just one of its methods. You can create a new class that provides a new implementation of the method you're interested in changing and delegates all other methods to the corresponding method of x.**

**The delegation is accomplished via the getattr method; consult the language reference for more information about controlling attribute access.**

**Q3. What is the concept of composition?**

**ANS:-**

**It is one of the fundamental concepts of Object-Oriented Programming. A class that references to one or more objects of other classes as an Instance variable. Here, by using the class name or by creating the object we can access the members of one class inside another class. It enables creating complex types by combining objects of different classes. It means that a class Composite can contain an object of another class Component. This type of relationship is known as Has-A Relation.**

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

**ANS:-**

**A bound method is the one which is dependent on the instance of the class as the first argument. It passes the instance as the first argument which is used to access the variables and functions. All functions in the class are by default bound methods.**

**class Sample:**

**def fun(self, some\_value):**

**self.some\_value = some\_value**

**s = Sample()**

**print(s.fun)**

**<bound method Sample.fun of <\_\_main\_\_.Sample object at 0x0000024D6EEA5AC0>>**

**Q5. What is the purpose of pseudoprivate attributes?**

**ANS:-**

**In Python, there is no concept of "private" as such. When naming properties and methods at the level, some special processing is actually done to the name, making it inaccessible to the outside world.**

**class Test:**

**def \_\_init\_\_(self, a, b, c, d):**

**self.\_\_a = a *# \_\_ means pseudo private variable***

**self.b = b**

**self.c = c**

**self.d = d**

**def custom(self, v):**

**return v - self.\_\_a**

**t = Test(1,2,3,4)**

**print(t.\_Test\_\_a) *# this is the way to access Pseudo private attribute***

**print(t.b)**

**print(t.custom(5))**

**1**

**2**

**4**