Q1. What is the meaning of multiple inheritance?

ANS:

Multiple inheritance means that a subclass can inherit from two or more superclasses. C++ allows multiple inheritance, but Java allows only single inheritance, that is, a subclass can inherit only one superclass.

Q2. What is the concept of delegation?

ANS:

Delegation is the assignment of authority to another person (normally from a manager to a subordinate) to carry out specific activities. It is the process of distributing and entrusting work to another person, and therefore one of the core concepts of management leadership.

Q3. What is the concept of composition?

ANS:

 the act or process of composing. specifically : arrangement into specific proportion or relation and especially into artistic form. the painting's unique composition. b(1) : the arrangement of type for printing.

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

ANS:

The Bound methods are the methods defined in a class. The bound methods are bound to the class in which they are defined. They take an instance of their class along with the other arguments as its parameters. Bound methods differ from the other unbound methods in their first parameter.

Q5. What is the purpose of pseudoprivate attributes?\

ANS:

One of the main problems that the pseudoprivate attribute feature is meant to alleviate has to do with the way instance attributes are stored. In Python, all instance attributes wind up in the single instance object at the bottom of the class tree. This is different from the C++ model, where each class gets its own space for data members it defines.

Within a class method in Python, whenever a method assigns to a self attribute (e.g., self.attr = value), it changes or creates an attribute in the instance ([inheritance searches](https://www.pythonstudio.us/object-oriented/attribute-inheritance-search.html) happen only on reference, not on assignment). Because this is true even if multiple classes in a hierarchy assign to the same attribute, collisions are possible.

For example, suppose that when a programmer codes a class, she assumes that she owns the attribute name X in the instance. In this class's methods, the name is set, and later fetched: class C1:

def methl(self): self.X = 88 # I assume X is mine def meth2(self): print(self.X)

Suppose further that another programmer, working in isolation, makes the same assumption in a class that he codes: class C2:

def metha(self): self.X = 99 # Me too def methb(self): print(self.X)