Q1. What is the meaning of multiple inheritance?

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.

Syntax:

Class Base1:

Body of the class

Class Base2:

Body of the class

Class Derived(Base1, Base2):

Body of the class

Q2. What is the concept of delegation?

“the delegation pattern is an object-oriented design pattern that allows object composition to achieve the same code reuse as inheritance.”

Let’s say we have a Dog class that is a subclass (and thus inherits the functionality of) an Animal class. If Animal has a method called get\_number\_of\_legs, any instantiation of the Dog class can call the get\_number\_of\_legs method. In Python, an implementation might look like this:

class Animal:

def \_\_init\_\_(self, name, num\_of\_legs):

self.name = name

self.num\_of\_legs = num\_of\_legs

def get\_number\_of\_legs(self):

print(f"I have {self.num\_of\_legs} legs")

class Dog(Animal):

def \_\_init\_\_(self, name, num\_of\_legs):

super().\_\_init\_\_(name, num\_of\_legs)

dog = Dog('Fido', 4)

dog.get\_number\_of\_legs()

# Outputs "I have 4 legs"

It would be technically incorrect to say that Dog delegates get\_number\_of\_legs to Animal because the Dog class actually has that method since it inherits the Animal class. This is what the Wikipedia definition is talking about when it refers to “code reuse.” This is what delegation will duplicate when we use composition.

Q3. What is the concept of composition?

Composition is a concept that models a has a relationship. It enables creating complex types by combining objects of other types. This means that a class Composite can contain an object of another class Component. This relationship means that a Composite has a Component.

Q4. What are C and how do we use them?

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. In Python 3 and newer versions of python, all functions in the class are by default bound methods.

Let’s understand this concept with an example:

class A:

def func(self, arg):

self.arg = arg

print("Value of arg = ", arg)

# Creating an instance

obj = A()

# bound method

print(obj.func)

The instance obj is automatically passed as the first argument to the function called and hence the first parameter of the function will be used to access the variables/functions of the object.

Q5. What is the purpose of pseudoprivate attributes?

This trick can avoid potential name collisions in the instance, but note that it does not amount to true privacy. If you know the name of the enclosing class, you can still access either of these attributes anywhere you have a reference to the instance by using the fully expanded name. On the other hand, this feature makes it less likely that you will accidentally step on a class's names.

Pseudoprivate attributes are also useful in larger frameworks or tools, both to avoid introducing new method names that might accidentally hide definitions elsewhere in the class tree and to reduce the chance of internal methods being replaced by names defined lower in the tree. If a method is intended for use only within a class that may be mixed into other classes, the double underscore prefix ensures that the method won't interfere with other names in the tree, especially in multiple-inheritance scenarios: class Super