Q1. What is the meaning of multiple inheritance?

Ans: - Multiple inheritance in Python is a feature where a class can be derived from more than one superclass. Here is an example:

```
class Mammal:
    def mammal_info(self):
        print("Mammals can give direct birth.")

class WingedAnimal:
    def winged_animal_info(self):
        print("Winged animals can flap.")

class Bat(Mammal, WingedAnimal):
    pass

# Create an object of Bat class
b1 = Bat()
b1.mammal_info()
b1.winged_animal_info()
```

Q2. What is the concept of delegation?

Ans: - Delegation in Python refers to the process of passing a method call from one object (the delegator) to another object (the delegate) that performs the actual behavior of the method. Here is an example:

```
class Delegator:
    def __getattr__(self, called_method):
        def wrapper(*args, **kwargs):
            delegation_config = getattr(self, 'DELEGATED_METHODS', None)
            if not isinstance(delegation_config, dict):
                raise AttributeError("'{}' object has not defined any delegated
methods".format(self.__class__.__name__))
            for delegate_object_str, delegated_methods in delegation_config.items():
                if called_method in delegated_methods:
                    break
            else:
                raise AttributeError("'{}' object has no attribute
'{}'".format(self.__class__.__name__, called_method))
            delegate_object = getattr(self, delegate_object_str, None)
            return getattr(delegate_object, called_method)(*args, **kwargs)
        return wrapper
```

Q3. What is the concept of composition?

Ans: - Composition in Python is an object-oriented design concept that models a “has-a” relationship. In composition, a class (known as the composite) contains an object of another class (known as the component). Here is an example:

```
class Component:
    def __init__(self):
        print('Component class object created...')

class Composite:
    def __init__(self):
        self.obj1 = Component()
        print('Composite class object also created...')
```

In this example, the Composite class contains an object of the Component class.

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

Ans: - Bound methods in Python are methods that are dependent on the instance of the class as the first argument. They allow a class’s function to be called on an instance, instead of alone. Here is an example: -

```
class MyClass:
    def my_method(self):
        print("This is a bound method.")

# Create an instance of MyClass
my_instance = MyClass()

# Call the method of the instance
my_instance.my_method()  # Outputs: This is a bound method.
```

Q5. What is the purpose of pseudoprivate attributes?

Ans: - Pseudoprivate attributes in Python are used to “mangle” attribute names and they become less likely to conflict with attributes in subclasses. They are denoted by a double underscore prefix (and at most a single underscore suffix). For example, __X in a class becomes _classname__X automatically. This feature is mostly intended to avoid namespace collisions in instances, not to restrict access to names in general. Here is an example:

```
class MyClass:
    def __init__(self):
        self.__X = 10

    def print_X(self):
        print(self.__X)

my_instance = MyClass()
my_instance.print_X()  # Outputs: 10
```

In this example, __X is a pseudoprivate attribute. It is accessed within the class using self.__X, but outside the class, it would be accessed using the mangled name