Inheritance & OOP – Exercise Set

1. Constructor Execution Order

Create a base class and two subclasses that form an inheritance chain.

Each class must print a message from its constructor.

Create an object of the most derived class and observe the order in which the constructors are executed.

Use super() in all constructors.

2. Modifying State Through super()

Create a base class that stores a numeric value.

Create two subclasses that modify this value in different ways inside their constructors.

Use super() so that each class in the inheritance chain contributes to the final value.

Print the final result.

3. Multiple Inheritance and MRO

Create two parent classes that both modify the same attribute in their constructors.

Create a child class that inherits from both parents.

Use super() in all constructors.

Print the final value and print the MRO of the child class.

Explain why the final value is produced.

4. Predict and Verify MRO

Create a diamond-shaped inheritance structure with four classes.

Before running the program, write down what you believe the MRO will be.

Then print the actual MRO and compare it with your prediction.

5. Overriding a Method and Calling the Parent

Create a base class with a method that prints a message.

Override this method in a subclass.

Inside the overridden method, call the parent version using super() and then add additional behavior.

Call the method and show both outputs.

6. Class Variables Across Inheritance

Create a base class with a class variable.

Create two subclasses.

Override the class variable in one subclass but not the other.

Create objects from all classes and show how the value differs depending on which class is used.

7. Protected Attributes

Create a base class that uses a protected attribute (single underscore).

Access and modify this attribute from a subclass.

Demonstrate that the attribute can be accessed, and explain why this is allowed but discouraged outside the class hierarchy.

8. Private Attributes and Name Mangling

Create a base class with a private attribute using double underscores.

Attempt to access the attribute directly and observe what happens.

Then access it using name mangling and explain how Python handles private attributes internally.

9. Overriding __str__ in a Subclass

Create a base class with a meaningful __str__ method.

Override __str__ in a subclass.

Use super().__str__() inside the subclass and add subclass-specific information.

Print objects of both classes to show the difference.

10. Design Task – Controlled Inheritance

Design a small inheritance hierarchy that models a real-world concept.

Requirements:

- Use inheritance and super() correctly
- Override at least one method
- Include at least one class variable
- Include either a protected or private attribute
- Do not use polymorphism or duck typing

Create objects and demonstrate that the inheritance structure works as intended.