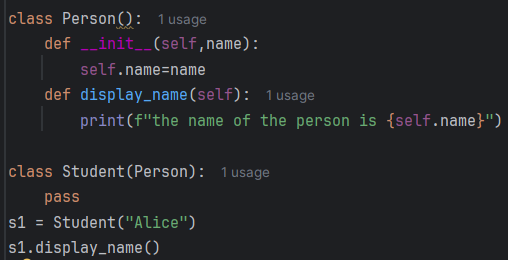
DAY 8: MORNING ASSESSMENT

1. Single Inheritance:

Create a base class Person with a method display\_name(). Inherit it in a class Student and call

the method.

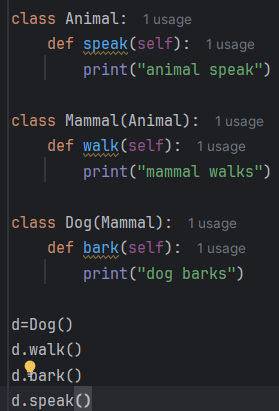


Output: the name of the person is Alice

2. Multilevel Inheritance:

Design 3 classes: Animal → Mammal → Dog, where each class has its own method and Dog

inherits all behaviors.



Output:  
animal swim

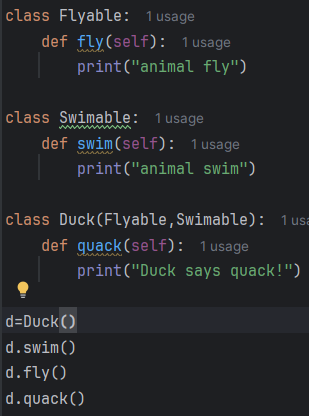
animal fly

Duck says quack!

3. Multiple Inheritance:

Create two classes Flyable and Swimmable, each with a method. Derive a class Duck from both

and call both methods.

  
output: animal swim

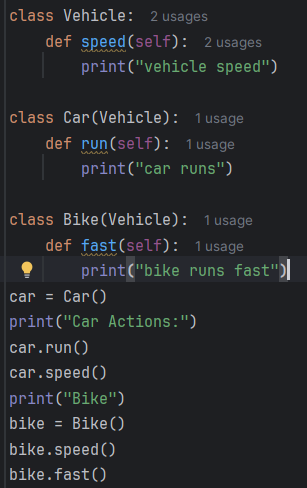
animal fly

Duck says quack!

4. Hierarchical Inheritance:

Define a parent class Vehicle, and create two child classes Car and Bike. Show how each

inherits from Vehicle.

  
output:

Car Actions:

car runs

vehicle speed

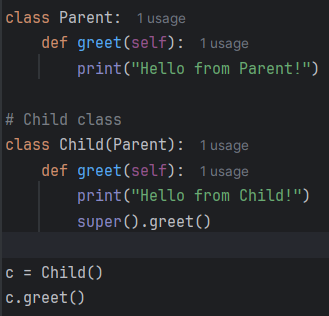
Bike

vehicle speed

bike runs fast

5. Use super() in a derived class to call a parent class's method. What happens if both classes

have the same method name?



Output:

Hello from Child!

Hello from Parent!

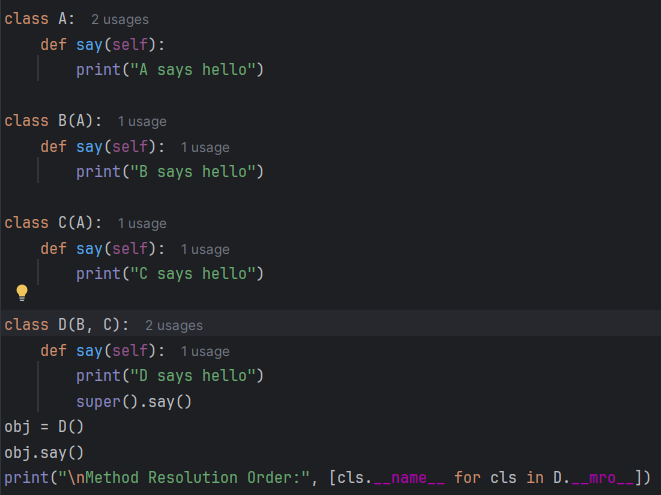
6. What is Method Resolution Order (MRO) in multiple inheritance? Demonstrate using a

diamond problem structure.

MRO (Method Resolution Order) is the order in which Python looks for a method or attribute in a class hierarchy, especially in multiple inheritance.

Each class appears only once in the search path.

The method from the closest class in the MRO is called first.



Output:

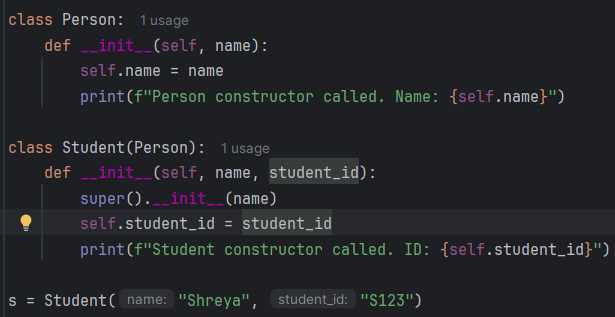
D says hello

B says hello

Method Resolution Order: ['D', 'B', 'C', 'A', 'object']

7. Define a constructor in the base class. In the derived class, call it using super().\_\_init\_\_() and

add new attributes.



\_\_init\_\_() is the constructor.

It is automatically called when you create an object of the class.

This constructor belongs to the base class Person.

Output:

Person constructor called. Name: Shreya

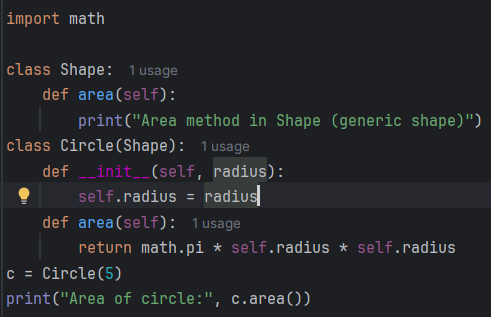
Student constructor called. ID: S123

8. Can you override a method in Python? Write a base class Shape with a method area() and

override it in Circle.

Yes, you can override a method in Python.

When a derived class provides its own implementation of a method that is already defined in the base class, it is called method overriding.

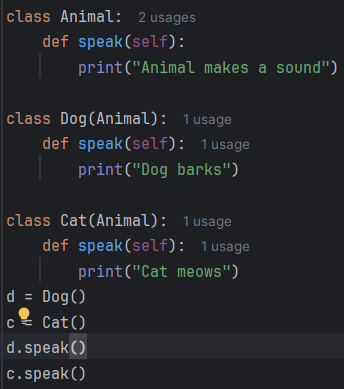


Output: Area of circle: 78.53981633974483

9. Method Overriding:

Write a base class Animal with method speak(). Create subclasses Dog, Cat that override

speak().



Output:

Dog barks

Cat meows

10. Polymorphic Behavior:

Create a list of objects of Dog, Cat, Cow, each inheriting from Animal. Iterate and call speak()

method.



output:

Dog barks

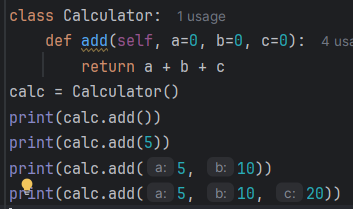
Cat meows

Cow moos

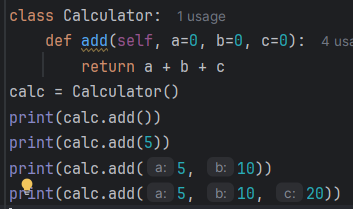
11. Simulated Method Overloading:

Python doesn’t support method overloading directly. Show how you can use default or \*args to

mimic it.

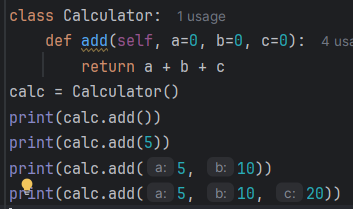
Default:  


Using \*args

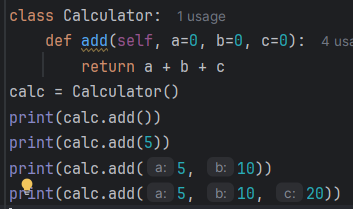


12. Write a class Calculator with a method add() that supports 2 and 3 arguments using default

parameters or \*args.

Default:  


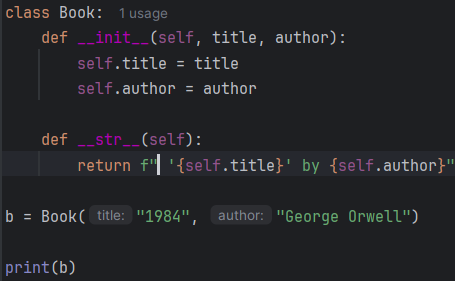
Using \*args:



13. Can you override the \_\_str\_\_() method in Python? Create a class Book that returns a

custom string when printed.

Yes, you can override the \_\_str\_\_() method in Python.

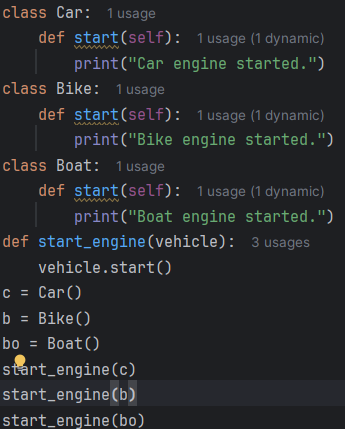


Output: '1984' by George Orwell

14. Demonstrate polymorphism using duck typing. Write a function start\_engine(vehicle) that

takes any object with a method start().

we don't check an object’s type we just call the method. This is duck typing



Output:  
Car engine started.

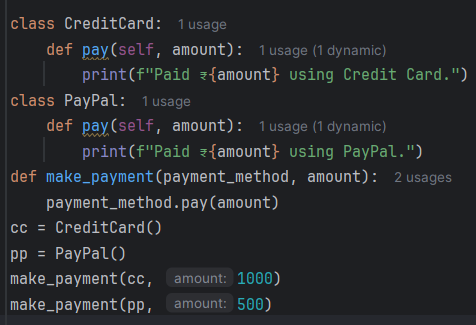
Bike engine started.

Boat engine started.

15. How does polymorphism help in writing more generic functions in Python? Provide a small

real-world code snippet.

Polymorphism allows us to write functions that operate on objects of different types



Output: Paid ₹1000 using Credit Card.

Paid ₹500 using PayPal.