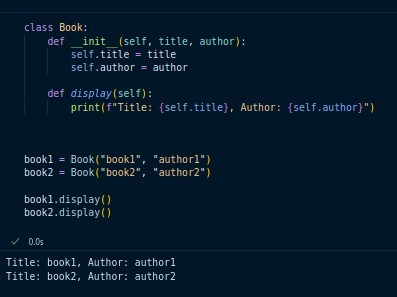
OOP in Python - Tasks

# Task 1: Classes & Objects (Basic)

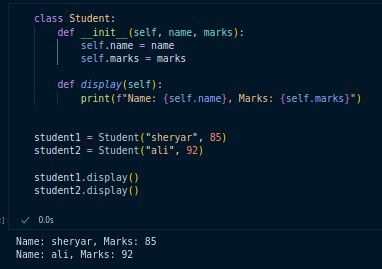
Create a class `Book` with attributes `title` and `author`. Create two book objects and print their details.



# Task 2: Instance Variables & Methods (Basic)

Make a class `Student` with instance variables `name` and `marks`. Add a method

`display()` to print the student’s details.



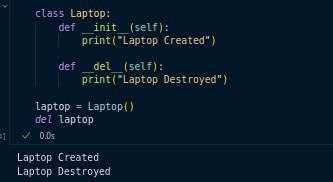
# Task 3: Class Variables & Methods (Intermediate)

Create a class `Employee` with a class variable `company = "ABC Ltd"`. Add a method to count how many employees have been created.



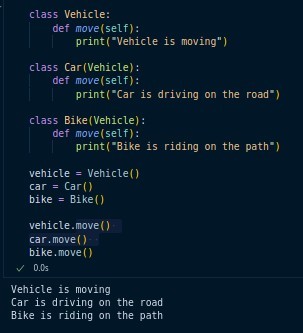
# Task 4: Constructor & Destructor (Intermediate)

Write a class `Laptop` with a constructor that prints 'Laptop Created' when an object is made, and a destructor that prints 'Laptop Destroyed' when the object is deleted.



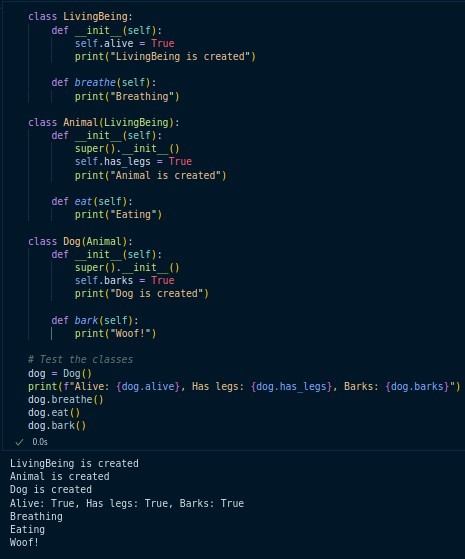
# Task 5: Inheritance (Intermediate)

Create a base class `Vehicle` with a method `move()`. Create two child classes `Car` and `Bike` that override the method.



# Task 6: Multilevel Inheritance (Intermediate)

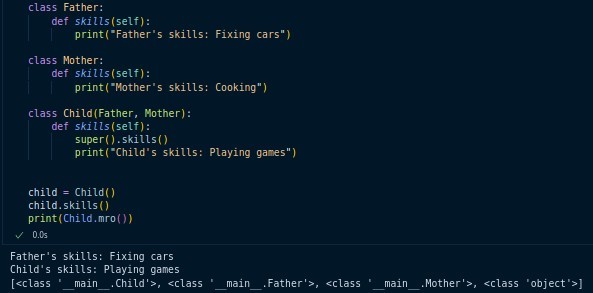
Build three classes: `LivingBeing` → `Animal` → `Dog`. Show how `Dog` inherits properties step by step.



# Task 7: Multiple Inheritance & MRO (Slightly Advanced)

Make two classes `Father` and `Mother` with a method `skills()`. Then create a

`Child` class that inherits from both and demonstrate how Python resolves conflicts (using `mro()`).



# Task 8: Access Specifiers (Intermediate)

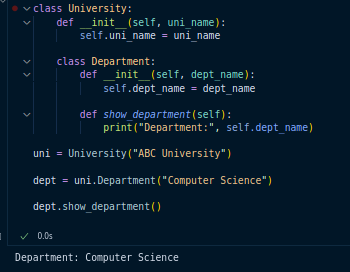
Create a class `Account` with:

* a public variable `balance`,
* a protected variable `\_pin`,
* a private variable ` password`.

Show how they can be accessed or restricted.

# Task 9: Inner/Nested Class (Intermediate)

Make a class `University` with a nested class `Department`. Print the department name by accessing the inner class from an outer class object.



# Task 10: Association, Aggregation & Composition (Advanced)

* Association: Create two independent classes `Teacher` and `Course` and link them by assigning a teacher to a course.
* Aggregation: A `School` class contains multiple `Student` objects but students can exist without the school.
* Composition: A `Car` class always creates an `Engine` object inside it, showing that the engine cannot exist without the car.

