Multiple Inheritance

Multiple Inheritance is where a class can inherit attributes and behaviors from more than one parent class. This allows for greater flexibility and code reuse.

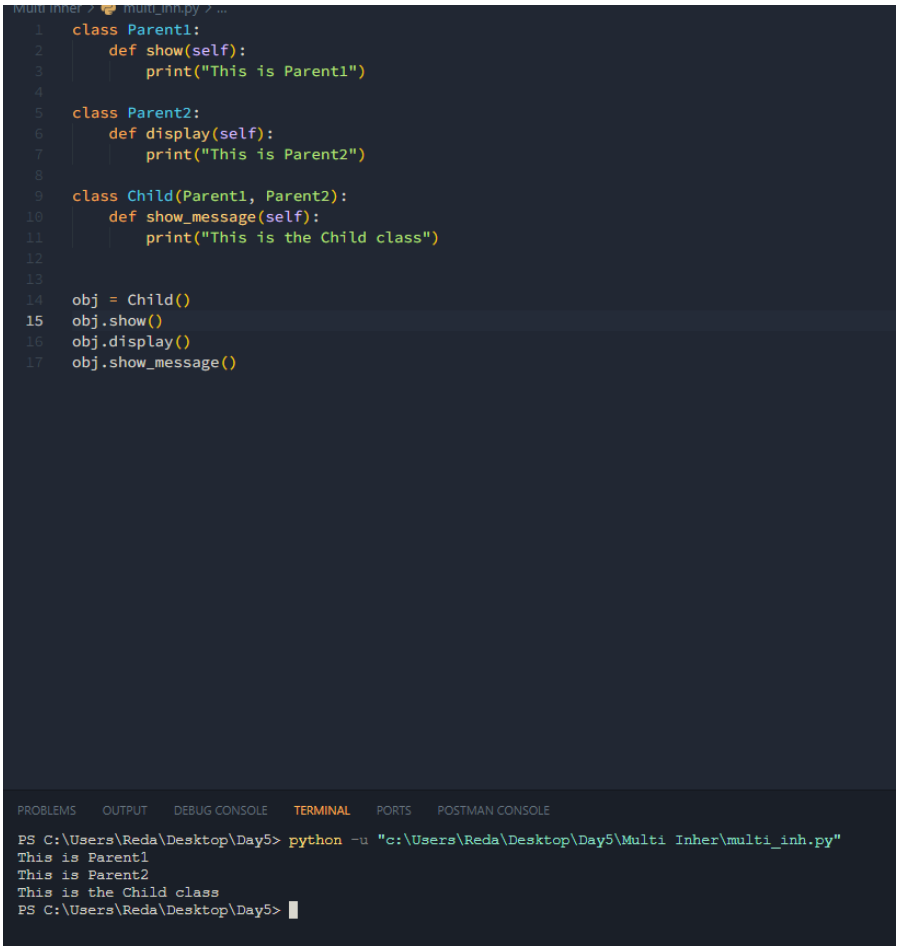
Advantages

1- Code Reusability

2- Enhanced Functionality

3- Improved Modularity

Code example



Understanding super() in Multiple Inheritance

In multiple inheritance, the super() function is used to call methods from parent classes in a structured way. It ensures that each parent class is accessed only once, avoiding redundant calls and potential conflicts. Python determines the method execution order using **MRO (Method Resolution Order)**, ensuring a consistent execution sequence.

- Why user super()?

1- Prevents duplicate method calls in multiple inheritance

2- Follows a structured order using **MRO** to call parent class methods

3- Makes the code more maintainable and avoids ambiguity in method resolution

Code example

```python
class Parent1:
    def show(self):
        print("This is Parent1")

class Parent2:
    def show(self):
        print("This is Parent2")

class Child(Parent1, Parent2):
    def show(self):
        super().show()
        print("This is the Child class")

obj = Child()
obj.show()
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS  POSTMAN CONSOLE

PS C:\Users\Reda\Desktop\Day5> python -u "c:\Users\Reda\Desktop\Day5\Multi Inher\multi_inh.py"
This is Parent1
This is Parent2
This is the Child class
PS C:\Users\Reda\Desktop\Day5> python -u "c:\Users\Reda\Desktop\Day5\Multi Inher\super_inh.py"
This is Parent1
This is the Child class
PS C:\Users\Reda\Desktop\Day5>
```