Day 43

Method Overriding in Python:

What is Method Overriding?

Method Overriding occurs when:

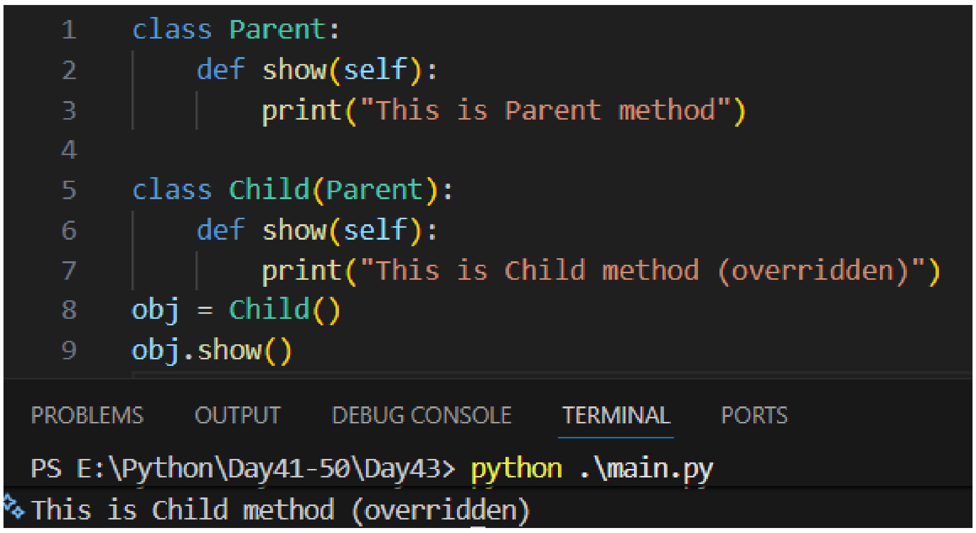
- A child class provides its own implementation of a method
- The method already exists in the parent class
- The method name and parameters are the same

The child's method overrides the parent's method.

Why Method Overriding is Needed?

- To change or extend parent class behavior
- To implement runtime polymorphism
- To customize functionality in derived classes

Example: a basic example.

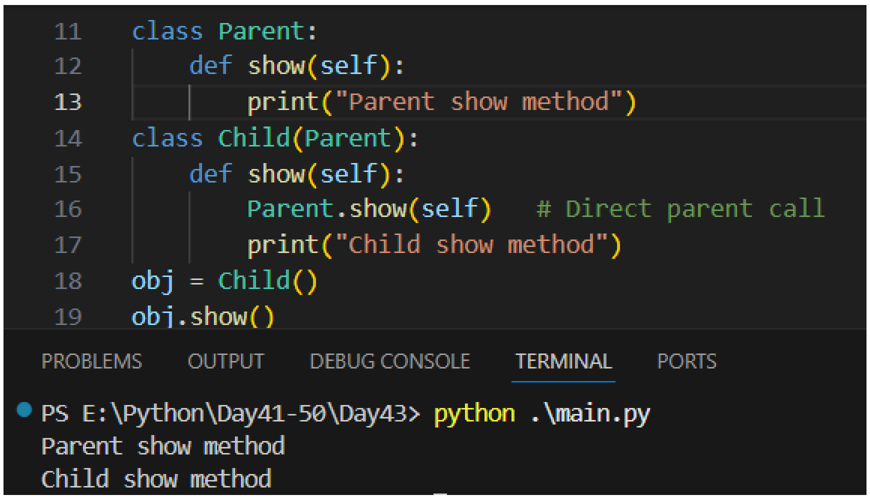


```python
class Parent:
    def show(self):
        print("This is Parent method")

class Child(Parent):
    def show(self):
        print("This is Child method (overridden)")
obj = Child()
obj.show()
```

```
PS E:\Python\Day41-50\Day43> python .\main.py
This is Child method (overridden)
```

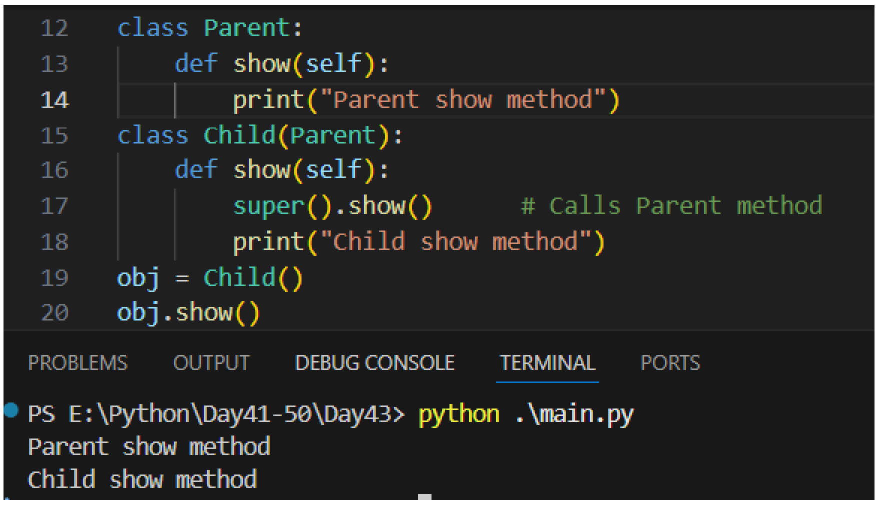
Example: Calling Parent Method Without super() (Not Recommended)



```python
class Parent:
    def show(self):
        print("Parent show method")
class Child(Parent):
    def show(self):
        Parent.show(self)   # Direct parent call
        print("Child show method")
obj = Child()
obj.show()
```

```
PS E:\Python\Day41-50\Day43> python .\main.py
Parent show method
Child show method
```

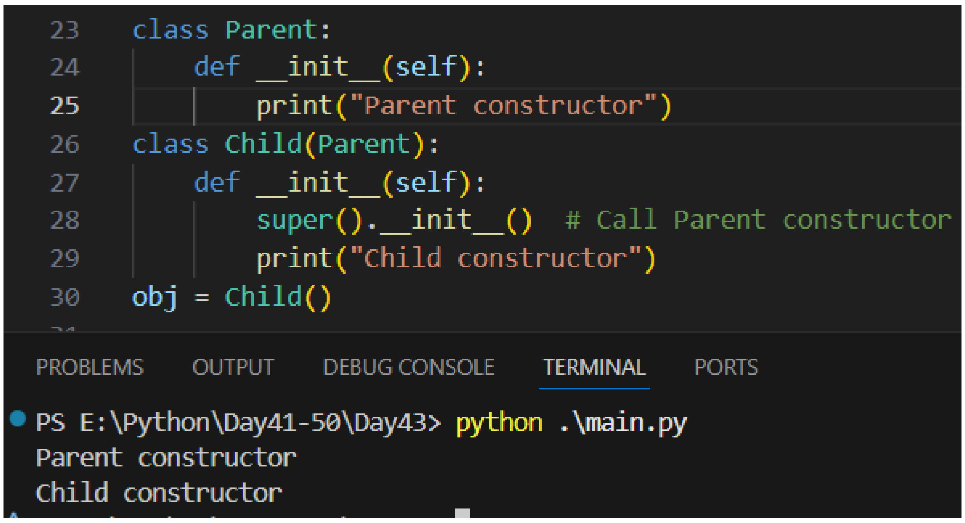
Example: Calling Parent Method With super() (Recommended)



```python
class Parent:
    def show(self):
        print("Parent show method")
class Child(Parent):
    def show(self):
        super().show()      # Calls Parent method
        print("Child show method")
obj = Child()
obj.show()
```

```
PS E:\Python\Day41-50\Day43> python .\main.py
Parent show method
Child show method
```

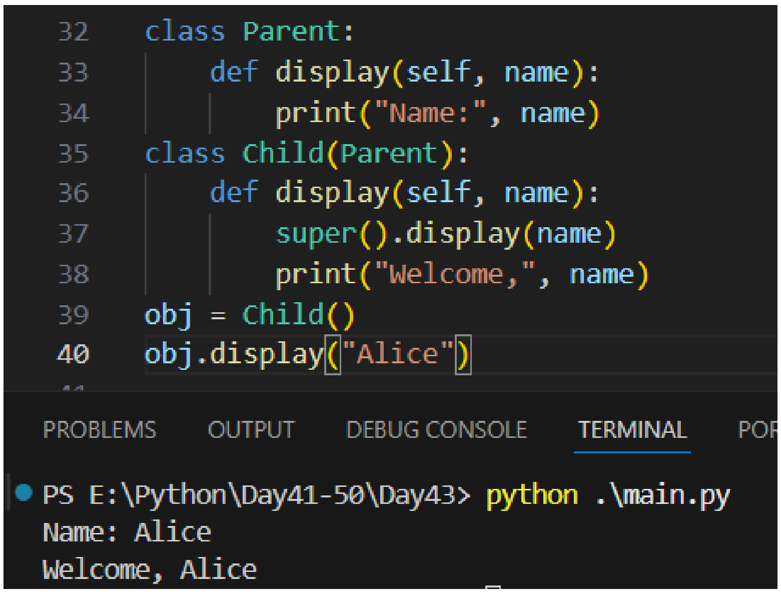
Example: Overriding __init__() Method



```python
class Parent:
    def __init__(self):
        print("Parent constructor")
class Child(Parent):
    def __init__(self):
        super().__init__()  # Call Parent constructor
        print("Child constructor")
obj = Child()
```

```
PS E:\Python\Day41-50\Day43> python .\main.py
Parent constructor
Child constructor
```

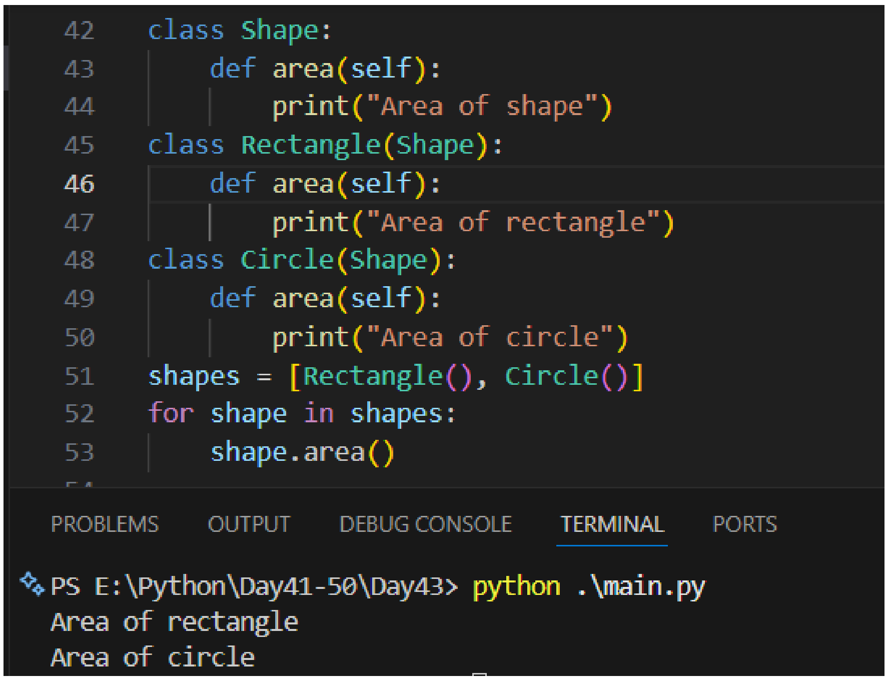
Example: Overriding with Arguments



```python
class Parent:
    def display(self, name):
        print("Name:", name)
class Child(Parent):
    def display(self, name):
        super().display(name)
        print("Welcome,", name)
obj = Child()
obj.display("Alice")
```

```
PS E:\Python\Day41-50\Day43> python .\main.py
Name: Alice
Welcome, Alice
```

Example: runtime polymorphism (method overriding)



```python
class Shape:
    def area(self):
        print("Area of shape")
class Rectangle(Shape):
    def area(self):
        print("Area of rectangle")
class Circle(Shape):
    def area(self):
        print("Area of circle")
shapes = [Rectangle(), Circle()]
for shape in shapes:
    shape.area()
```

```
PS E:\Python\Day41-50\Day43> python .\main.py
Area of rectangle
Area of circle
```

Summary:

- Method overriding allows child classes to change parent behavior
- Supports runtime polymorphism
- super() is the best way to call parent method
- Common in real-world OOP designs

--The End--