Python Programming

Unit 05 – Lecture 04: Polymorphism (Overriding and Operator Overloading)

Tofik Ali

School of Computer Science, UPES Dehradun

February 14, 2026

Repository: `https://github.com/tali7c/Python-Programming`

Quick Links

Core Concepts Demo Interactive Summary

Agenda

Learning Outcomes

- Explain polymorphism in OOP

Learning Outcomes

- Explain polymorphism in OOP
- Implement method overriding in subclasses

Learning Outcomes

- Explain polymorphism in OOP
- Implement method overriding in subclasses
- Implement operator overloading using special methods

Learning Outcomes

- Explain polymorphism in OOP
- Implement method overriding in subclasses
- Implement operator overloading using special methods
- Use `__str__` for readable object printing

Polymorphism (Idea)

- "Same interface, different behavior"

Polymorphism (Idea)

- "Same interface, different behavior"
- Example: `area()` behaves differently for Circle and Rectangle

Method Overriding

- A child class provides its own implementation of a parent method

Method Overriding

- A child class provides its own implementation of a parent method
- Python chooses the method based on the object's actual class

Operator Overloading

- Define how operators work for your objects

Operator Overloading

- Define how operators work for your objects
- Examples:

Operator Overloading

- Define how operators work for your objects
- Examples:
  - + uses `__add__`

Operator Overloading

- Define how operators work for your objects
- Examples:
  - + uses __add__
  - == uses __eq__

Operator Overloading

- Define how operators work for your objects
- Examples:
  - `+` uses `__add__`
  - `==` uses `__eq__`
  - `len(obj)` uses `__len__`

Example: Point Addition

```
class Point:
    def __init__(self, x, y):
        self.x = x
        self.y = y

    def __add__(self, other):
        return Point(self.x + other.x, self.y + other
```

Demo: Overriding + Operator Overloading

- File: `demo/polymorphism_operator_overloading_demo.py`

Demo: Overriding + Operator Overloading

- File: `demo/polymorphism_operator_overloading_demo.py`
- Shows:

Demo: Overriding + Operator Overloading

- File: `demo/polymorphism_operator_overloading_demo.py`
- Shows:
  - polymorphic `area()` calls

Demo: Overriding + Operator Overloading

- File: `demo/polymorphism_operator_overloading_demo.py`
- Shows:
  - polymorphic `area()` calls
  - `Point + Point` using `__add__`

Checkpoint 1

Question: If a parent and child both define `describe()`, which one runs for a child object?

Checkpoint 2

Question: Which special method is used for operator +?

Think-Pair-Share

Discuss:

- Should every class implement operator overloading? When is it helpful vs confusing?

Key Takeaways

- Overriding enables polymorphism (same method name, different behavior)

Key Takeaways

- Overriding enables polymorphism (same method name, different behavior)
- Operator overloading uses special methods like `__add__`

Key Takeaways

- Overriding enables polymorphism (same method name, different behavior)
- Operator overloading uses special methods like `__add__`
- Use overloading only when it improves readability and matches meaning

Exit Question

Name the special method used to control printing of an object using `print(obj)`.