Python Programming
Unit 05 – Lecture 04 Notes
Polymorphism (Overriding and Operator Overloading)

Tofik Ali

February 14, 2026

Contents

1 Lecture Overview

Polymorphism means the same method call can behave differently based on the object. In Python, this happens naturally through method overriding. Python also supports operator overloading using special (dunder) methods.

2 Core Concepts

2.1 Method Overriding

If a subclass defines a method with the same name as the parent method, it overrides it. When you call the method, Python uses the object's actual class.

```
class A:
    def greet(self):
        print("Hello from A")
```

```
class B(A):
    def greet(self):
        print("Hello from B")

b = B()
b.greet() # Hello from B
```

2.2 Operator Overloading

Operators map to special methods. Some examples:

- `+` → `__add__`
- `-` → `__sub__`
- `==` → `__eq__`
- `<` → `__lt__`
- `len(x)` → `__len__`
- `print(x)` → `__str__`

2.3 Point Example

```
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.y)
```

This allows `p3 = p1 + p2`.

2.4 When to Use Operator Overloading

Good cases:

- mathematical objects (Point, Vector, Matrix)
- domain objects where `+` has a clear meaning

Bad cases:

- when the meaning is unclear or surprising (hurts readability)

3 Demo Walkthrough

File: `demo/polymorphism_operator_overloading_demo.py`

Observe:

- different shapes implement `area()`
- `Point` supports `+` and has a readable `__str__`

4 Interactive Checkpoints (with Solutions)

Checkpoint 1 Solution

Question: If parent and child define `describe()`, which runs for child object?
Answer: The child's method runs (overriding).

Checkpoint 2 Solution

Question: Which special method for +?
Answer: `__add__`

5 Practice Exercises (with Solutions)

Exercise 1: Add `__str__` to Point

Task: Print `Point(10,20)` for a point object.
Solution:

```python
def __str__(self):
    return f"Point({self.x},{self.y})"
```

Exercise 2: Implement Subtraction

Task: Implement `p1 - p2`.
Solution:

```python
def __sub__(self, other):
    return Point(self.x - other.x, self.y - other.y)
```

6 Exit Question (with Solution)

Question: special method for printing?
Answer: `__str__`