Python Programming

Unit 05 – Lecture 03: Inheritance and Types of Inheritance

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 inheritance and its purpose

Learning Outcomes

- Explain inheritance and its purpose
- Create subclasses and reuse parent methods/attributes

Learning Outcomes

- Explain inheritance and its purpose
- Create subclasses and reuse parent methods/attributes
- Use `super()` to call parent constructor/methods

Learning Outcomes

- Explain inheritance and its purpose
- Create subclasses and reuse parent methods/attributes
- Use `super()` to call parent constructor/methods
- Identify types of inheritance (single, multilevel, multiple, hierarchical)

What is Inheritance?

- Create a new class from an existing class

What is Inheritance?

- Create a new class from an existing class
- Reuse common code and extend behavior

What is Inheritance?

- Create a new class from an existing class
- Reuse common code and extend behavior
- “is-a” relationship (Student is a Person)

Basic Syntax

```python
class Person:
    def __init__(self, name):
        self.name = name

class Student(Person):
    def __init__(self, name, sapid):
        super().__init__(name)
        self.sapid = sapid
```

Types of Inheritance

- Single: A $\rightarrow$ B

Types of Inheritance

- Single: A $\rightarrow$ B
- Multilevel: A $\rightarrow$ B $\rightarrow$ C

Types of Inheritance

- Single: A $\rightarrow$ B
- Multilevel: A $\rightarrow$ B $\rightarrow$ C
- Hierarchical: A $\rightarrow$ (B, C, D)

Types of Inheritance

- Single: A $\rightarrow$ B
- Multilevel: A $\rightarrow$ B $\rightarrow$ C
- Hierarchical: A $\rightarrow$ (B, C, D)
- Multiple: (A, B) $\rightarrow$ C

Why `super()`?

- Calls parent class methods safely

Why `super()`?

- Calls parent class methods safely
- Avoids duplicating parent initialization code

Why `super()`?

- Calls parent class methods safely
- Avoids duplicating parent initialization code
- Important in multiple inheritance (method resolution order)

Demo: Person → Student → PlacementStudent

- File: `demo/inheritance_demo.py`

Demo: Person → Student → PlacementStudent

- File: `demo/inheritance_demo.py`
- Demonstrates:

Demo: Person $\rightarrow$ Student $\rightarrow$ PlacementStudent

- File: `demo/inheritance_demo.py`
- Demonstrates:
  - single and multilevel inheritance

Demo: Person → Student → PlacementStudent

- File: `demo/inheritance_demo.py`
- Demonstrates:
  - single and multilevel inheritance
  - use of `super()`

Checkpoint 1

Question: What does `super()` do in a subclass constructor?

Checkpoint 2

Question: Give one real-world example of inheritance.

Think-Pair-Share

Discuss:

- Should `Car` inherit from `Engine`? Why or why not?

Key Takeaways

- Inheritance reuses and extends code

Key Takeaways

- Inheritance reuses and extends code
- Subclasses can add new attributes and methods

Key Takeaways

- Inheritance reuses and extends code
- Subclasses can add new attributes and methods
- `super()` helps call parent logic correctly

Key Takeaways

- Inheritance reuses and extends code
- Subclasses can add new attributes and methods
- `super()` helps call parent logic correctly
- Multiple inheritance exists, but use it carefully

Exit Question

What type of inheritance is: A $\rightarrow$ (B, C)?