Lecture 2.8

Inheritance in Python

Agenda

- Introduction to Inheritance
- Single Inheritance
- Multiple Inheritance & MRO
- Multi-level Inheritance
- Hierarchical Inheritance
- Hybrid Inheritance & The Diamond Problem

Introduction to Inheritance

- Inheritance allows a class (child/derived) to acquire the properties and methods of another class (parent/base).
- Key Benefits:
- - Code Reusability: Avoid rewriting existing code.
- - Transitivity: If B inherits from A, then B has all characteristics of A.
- Syntax: ‘class ChildClass(ParentClass):‘

Single Inheritance

- Definition: A child class inherits from exactly one parent class.
- It is the simplest and most common form of inheritance.
- Properties and methods of the parent are directly accessible via the child instance.
- Use ‘super()‘ to call parent class methods, especially constructors.

Single Inheritance Example

- “‘python class Animal: def speak(self): return ’Animal speaks’
  class Dog(Animal): def bark(self): return ’Dog barks’
  d = Dog() print(d.speak()) # Inherited print(d.bark()) # Own method “‘

Multiple Inheritance

- Definition: A child class inherits from more than one parent class.
- Python supports multiple inheritance natively, unlike Java.
- Syntax: ‘class Child(Parent1, Parent2):‘
- Potential issue: Ambiguity when multiple parents have methods with the same name.

Multiple Inheritance & MRO

- “‘python class Flyer: def fly(self): return 'Flying' class Swimmer: def swim(self): return 'Swimming' class Duck(Flyer, Swimmer): pass “‘
- Method Resolution Order (MRO) dictates how Python searches for base classes.
- Accessed via ‘ClassName.mro()‘ or ‘ClassName.__mro__‘.
- Python uses the C3 Linearization algorithm for MRO.

Multi-level Inheritance

- Definition: A class is derived from a class which is also derived from another class.
- Creates a linear hierarchy or chain of inheritance.
- Example: Grandparent -¿ Parent -¿ Child.
- The child class has access to methods from all ancestors.

Multi-level Inheritance Example

- “‘python class Vehicle: def start(self): print('Vehicle starting') class Car(Vehicle): def drive(self): print('Car driving') class SportsCar(Car): def turbo(self): print('Turbo on') sc = SportsCar() sc.start() sc.drive() sc.turbo() “‘

Hierarchical Inheritance

- Definition: More than one derived class is created from a single base class.
- Useful when multiple classes share a common set of properties.
- Example: Shape is a base class. Circle, Square, and Triangle inherit from Shape.
- Promotes strong code modularity.

Hybrid Inheritance

- Definition: A combination of two or more types of inheritance.
- Often involves a mix of multiple and hierarchical inheritance.
- Can lead to the 'Diamond Problem' (resolved by Python's MRO).
- Requires careful design to prevent complex dependencies.

Summary

- Inheritance enables code reusability and logical hierarchies.
- Single: 1 Parent -¿ 1 Child.
- Multiple: ¿1 Parent -¿ 1 Child (MRO is critical here).
- Multi-level: Grandparent -¿ Parent -¿ Child.
- Hierarchical: 1 Parent -¿ ¿1 Children.
- Hybrid: Combination of multiple types.