🧮 Interview Questions : inheritance in Python

🎯 **Objective**:

To assess a candidate's understanding of object-oriented programming in Python, specifically the concept and implementation of **inheritance**, including types, usage, and advanced techniques.

🟢 Beginner-Level Questions

1. What is inheritance in Python? Why is it useful?
2. How do you define a child class that inherits from a parent class?
3. What is the syntax for inheritance in Python? Provide an example.
4. What is the use of the `super()` function in Python?
5. What does the term "reusability" mean in the context of inheritance?
6. Can a subclass override methods of the superclass? How?
7. What happens if a method is not found in the child class?

🟡 Intermediate-Level Questions

8. What are the different types of inheritance supported in Python?
    - Single
    - Multiple
    - Multilevel
    - Hierarchical
    - Hybrid
9. Explain method overriding with an example.
10. How do constructors (`__init__`) work with inheritance?
11. What is the difference between `super().__init__()` and directly calling `ParentClass.__init__()`?
12. What is the Method Resolution Order (MRO) in Python?
13. How does Python handle conflicts in multiple inheritance?
14. What is the role of `isinstance()` and `issubclass()` in inheritance?

🔴 Advanced-Level Questions

15. Explain the Diamond Problem in Python and how Python handles it.
16. What is the C3 Linearization Algorithm in Python?
17. How does multiple inheritance affect performance and readability in Python?
18. Can you create an abstract base class in Python? How does it relate to inheritance?
19. What is the difference between composition and inheritance? Which one is preferred and when?
20. How can mixins be implemented using inheritance in Python?

🧪 Scenario-Based Questions

21. Design a class hierarchy for different types of vehicles (e.g., Car, Bike, Truck) using inheritance.
22. Create a class `Employee` with subclasses `Manager` and `Developer`. Each subclass should have its own method `get_role()`.
23. Write code to demonstrate method overriding and use of `super()` to extend parent functionality.
24. Build a class structure where `ClassC` inherits from both `ClassA` and `ClassB`, and resolve method name conflicts.
25. You are designing a game. Create a base class `Character` and subclasses `Warrior`, `Archer`, and `Mage` using inheritance. Each should override an `attack()` method.