Inheritance questions

**Additional Practical Questions Using Various Types of Inheritance**

1. **Single Inheritance (Practical):**  
   Define a base class Animal with a method speak(). Create a subclass Dog that inherits from Animal and overrides speak() to print "Bark". Demonstrate how the subclass method speak() overrides the parent method.
2. **Multiple Inheritance (Practical):**  
   Create two classes Player and Coach with a common method get\_role() in each class that prints the role of the person. Then create a class AthleteCoach that inherits both Player and Coach. Call get\_role() from AthleteCoach and demonstrate how Python resolves the method from the first parent class (Player).
3. **Multilevel Inheritance (Practical):**  
   Design a class hierarchy where Person is the base class, Employee inherits Person, and Manager inherits Employee. Each class should have an info() method that prints relevant details. Demonstrate how Manager inherits methods from both Person and Employee.
4. **Hierarchical Inheritance (Practical):**  
   Create a base class Shape with a method draw(). Then, create two subclasses: Circle and Rectangle, each with its specific draw() implementation. Demonstrate how both Circle and Rectangle inherit from Shape and override the draw() method.
5. **Hybrid Inheritance (Practical):**  
   Build a class structure where Device is the base class, and it has two subclasses: Computer and Mobile. Then, create a subclass Smartphone that inherits from both Computer and Mobile. Show how Smartphone can access attributes or methods from both Computer and Mobile, demonstrating hybrid inheritance.

**Single Inheritance**

1. **Create a class Book with attributes title and author, and a method show\_details() to display these details. Then, create a subclass EBook that inherits Book, adds a new attribute file\_size, and overrides show\_details() to include file size information. Demonstrate how the subclass inherits and extends functionality from the parent class.**

**Multiple Inheritance**

1. **Define two classes: HomeAppliance with a method turn\_on(), and SmartDevice with a method connect\_to\_wifi(). Then, create a subclass SmartFridge that inherits from both HomeAppliance and SmartDevice and includes a new method check\_temperature(). Demonstrate how SmartFridge inherits and combines functionality from both parents.**

**Multilevel Inheritance**

1. **Create a base class School with a method show\_name() that prints "This is a school". Then create a subclass Classroom that inherits School and adds a method show\_classroom() that prints "This is a classroom". Next, create another subclass Student that inherits Classroom and adds a method show\_student() that prints "This is a student". Demonstrate the chain of inheritance across multiple levels.**

**Hierarchical Inheritance**

1. **Create a base class Employee with a method get\_salary() that returns the base salary. Create two subclasses, Developer and Designer, both of which inherit from Employee but override get\_salary() to calculate different pay scales for each role. Demonstrate how both subclasses share common properties but implement their unique calculations.**

**Hybrid Inheritance**

1. **Design a class structure with a base class Pet with a method feed(). Create two subclasses: Cat and Dog, each with unique methods meow() and bark(), respectively. Then create a hybrid class RobotDog that inherits from both Dog and another class Robot, which has a method charge(). Demonstrate how RobotDog can access methods from all parent classes in the hybrid hierarchy.**

**Additional Practical Questions with Various Types of Inheritance**

1. **Single Inheritance:**  
   Define a class Vehicle with attributes make and year. Create a subclass Truck that inherits Vehicle, adds a new attribute capacity, and includes a method load\_capacity() that displays the truck’s capacity. Demonstrate how the subclass can use and extend the functionality of Vehicle.
2. **Multiple Inheritance:**  
   Create two parent classes: Artist with an attribute style and a method create\_art(), and Teacher with a method teach(). Then create a subclass ArtTeacher that inherits from both Artist and Teacher. Show how ArtTeacher can access both create\_art() and teach() methods, combining the functionality of both parents.
3. **Multilevel Inheritance:**  
   Design a class hierarchy with Machine as the base class that has a method start(). Create a subclass Computer that inherits Machine and adds a method compute(). Finally, create a subclass Laptop that inherits Computer and adds a method portable(). Demonstrate how Laptop inherits methods from both Machine and Computer.
4. **Hierarchical Inheritance:**  
   Write a base class Plant with a method grow(). Create two subclasses: FloweringPlant and NonFloweringPlant, both inheriting from Plant but with distinct methods: FloweringPlant should have bloom(), and NonFloweringPlant should have spread(). Demonstrate how both subclasses inherit from Plant but have unique behaviors.
5. **Hybrid Inheritance:**  
   Create a structure where Device is the base class with a method power\_on(). Add two subclasses Phone and Tablet, each with unique methods call() and draw(), respectively. Then create a subclass SmartTablet that inherits from both Phone and Tablet and adds an additional method browse\_web(). Show how SmartTablet combines features from both Phone and Tablet in a hybrid inheritance model.

**Bonus Challenge Questions**

1. **Create a class Animal with a method eat(). Then, create two subclasses: Herbivore with a method graze() and Carnivore with a method hunt(). Finally, create a hybrid subclass Omnivore that inherits from both Herbivore and Carnivore and adds a method digest(). Demonstrate how Omnivore can graze, hunt, and digest, showing hybrid inheritance.**
2. **Design a class Building with a method open\_doors(). Create a subclass ResidentialBuilding that inherits Building and adds a method lock\_doors(). Add another subclass CommercialBuilding that inherits Building and adds a method advertise(). Finally, create a Mall class that inherits from both ResidentialBuilding and CommercialBuilding, adding a method host\_event(). Show how Mall combines functionalities of both residential and commercial buildings in a hybrid model.**