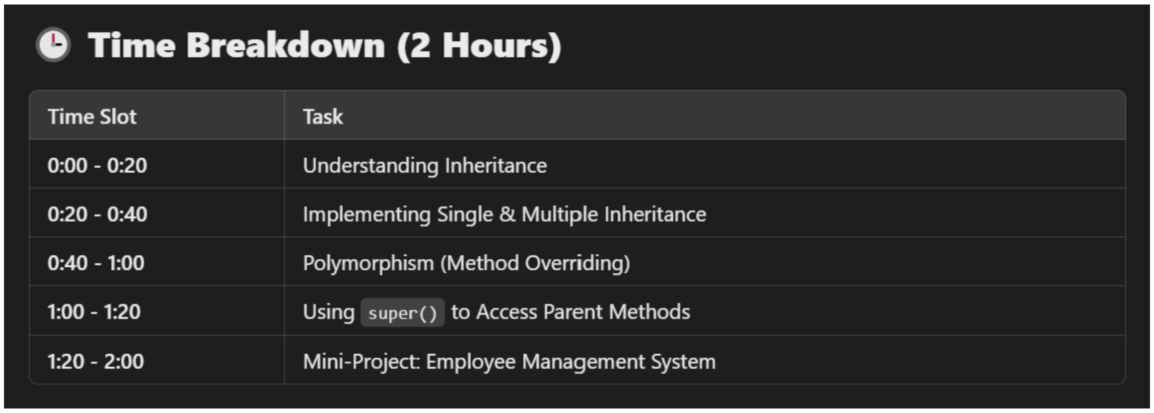
Day 12 of 6 weeks Python course:



🕒 **Time Breakdown (2 Hours)**

Time Slot	Task
0:00 - 0:20	Understanding Inheritance
0:20 - 0:40	Implementing Single & Multiple Inheritance
0:40 - 1:00	Polymorphism (Method Overriding)
1:00 - 1:20	Using `super()` to Access Parent Methods
1:20 - 2:00	Mini-Project: Employee Management System

1️⃣ What is Inheritance? ✅ Definition: Inheritance allows a child class to inherit attributes and methods from a parent class, reducing code duplication. ✅ Basic Example

In [2]:
```python
class Animal:
    def make_sound(self):
        print("Some generic sound")

class Dog(Animal):  # Dog class inherits from Animal
    def make_sound(self):
        print("Bark!")

dog = Dog()
dog.make_sound()  # Output: Bark!
```

```
Bark!
```

💡 The Dog class overrides the make_sound() method of the Animal class. ✅ Why Use Inheritance? •Avoid code duplication. •Extend functionality of an existing class. 2️⃣ Single & Multiple Inheritance ✅ Single Inheritance (One Parent, One Child)

In [4]:
```python
class Person:
    def __init__(self, name):
        self.name = name

    def display(self):
        print(f"Name: {self.name}")

class Student(Person):  # Inheriting from Person
    def __init__(self, name, student_id):
        super().__init__(name)  # Call parent constructor
        self.student_id = student_id

    def display(self):  # Overriding Parent Method
        print(f"Name: {self.name}, ID: {self.student_id}")

student1 = Student("David", 101)
student1.display()  # Output: Name: Deepak, ID: 101
```

```
Name: David, ID: 101
```

✅ Multiple Inheritance (Multiple Parents)

In [6]:
```
class Father:
    def show_father(self):
        print("Father's Traits")

class Mother:
    def show_mother(self):
        print("Mother's Traits")

class Child(Father, Mother):  # Inheriting from two classes
    def show_child(self):
        print("Child's Traits")

child = Child()
child.show_father()  # Output: Father's Traits
child.show_mother()  # Output: Mother's Traits
```

```
Father's Traits
Mother's Traits
```

3️⃣ Polymorphism (Method Overriding) ✅ Polymorphism means using the same method name in different classes. •Example: make_sound() is defined in multiple classes with different behavior. ✅ Example:

In [8]:
```
class Bird:
    def make_sound(self):
        print("Chirp!")

class Dog:
    def make_sound(self):
        print("Bark!")

# Using the same function for different objects
for animal in [Bird(), Dog()]:
    animal.make_sound()
```

```
Chirp!
Bark!
```

💡 Python automatically calls the correct method based on the object type. ✅ Method Overriding (Changing Parent Methods in Child Class)

In [10]:
```
class Vehicle:
    def move(self):
        print("This vehicle moves")

class Car(Vehicle):
    def move(self):  # Overriding the parent method
        print("This car drives")

car = Car()
car.move()  # Output: This car drives
```

```
This car drives
```

💡 The Car class replaces the move() method from Vehicle. 4️⃣ Using super() to Call Parent Methods (20 mins) ✅ Why Use super()? •Access methods from a parent class inside a child class. •Useful when extending functionality without rewriting existing code. ✅ Example:

In [14]:
```
class Animal:
    def __init__(self, name):
        self.name = name

    def make_sound(self):
        print("Animal makes a sound")

class Dog(Animal):
    def __init__(self, name, breed):
        super().__init__(name)  # Call Parent Constructor
        self.breed = breed

    def make_sound(self):
        super().make_sound()  # Call Parent Method
        print("Dog Barks!")

dog = Dog("Max", "Labrador")
dog.make_sound()
```

```
Animal makes a sound
Dog Barks!
```

🎯 Mini-Project: Employee Management System 📌 Project Goal •Store Employee details (name, ID, department). •Use Inheritance for different employee types (Manager, Developer). •Implement Method Overriding for salary calculation. 🖥️ Code Implementation

In [16]:
```
class Employee:
    def __init__(self, name, emp_id, department):
        self.name = name
        self.emp_id = emp_id
        self.department = department

    def show_details(self):
        print(f"Employee: {self.name}, ID: {self.emp_id}, Department: {self.departm

    def calculate_salary(self):
        return 50000  # Base salary

class Manager(Employee):
    def __init__(self, name, emp_id, department, bonus):
        super().__init__(name, emp_id, department)
        self.bonus = bonus

    def calculate_salary(self):
        return super().calculate_salary() + self.bonus

class Developer(Employee):
    def __init__(self, name, emp_id, department, tech_stack):
        super().__init__(name, emp_id, department)
        self.tech_stack = tech_stack

    def show_details(self):
        super().show_details()
        print(f"Tech Stack: {', '.join(self.tech_stack)}")

# Employee List
employees = []

# Function to add an employee
```

```python
def add_employee():
    name = input("Enter name: ")
    emp_id = input("Enter ID: ")
    department = input("Enter department: ")
    role = input("Enter role (Manager/Developer): ")

    if role.lower() == "manager":
        bonus = int(input("Enter bonus amount: "))
        employee = Manager(name, emp_id, department, bonus)
    else:
        tech_stack = input("Enter tech skills (comma-separated): ").split(", ")
        employee = Developer(name, emp_id, department, tech_stack)

    employees.append(employee)
    print(f"{name} added successfully!\n")

# Function to display employees
def view_employees():
    if employees:
        print("\n📋 Employee List 📋")
        for emp in employees:
            emp.show_details()
            print(f"Salary: ₹{emp.calculate_salary()}\n")
    else:
        print("No employees found.\n")

# Main Menu
while True:
    print("\n🏢 Employee Management System 🏢")
    print("1. Add Employee")
    print("2. View Employees")
    print("3. Exit")

    choice = input("Enter your choice (1-3): ")

    if choice == "1":
        add_employee()
    elif choice == "2":
        view_employees()
    elif choice == "3":
        print("Exiting System. Goodbye!")
        break
    else:
        print("Invalid choice!\n")
```

```
🏢 Employee Management System 🏢
1. Add Employee
2. View Employees
3. Exit
kovind added successfully!


🏢 Employee Management System 🏢
1. Add Employee
2. View Employees
3. Exit
```

```
📋 Employee List 📋
Employee: kovind, ID: 805, Department: fire
Salary: ₹150000


🏢 Employee Management System 🏢
1. Add Employee
2. View Employees
3. Exit
Exiting System. Goodbye!
```

📌 Summary of Day 12 ✔️ Learned Inheritance & Polymorphism ✔️ Practiced Method Overriding & super() ✔️ Completed a Mini-Project: Employee Management System