Chapter 5 - Inheritance

Theory:

Inheritance allows a class to inherit attributes and methods from another class.

Code Example:

```
# ■ What is Inheritance?
# Inheritance means one class (child) can reuse or extend the features (methods and
attributes)
# of another class (parent). It helps in reusability and organizing related classes.
# ■ Think of it like:
# A Programmer is also an Employee — so why rewrite all Employee details again? Just
inherit them!
# ■ Syntax:
# class Parent:
# # parent class code
#
# class Child(Parent):
# # child class code
# ■ Example: Parent and Child Classes using Inheritance
# -------------------
# Parent Class
# -------------------
class Employee:
company = "ITC"
def __init__(self, name, salary):
self.name = name
self.salary = salary
def show(self):
print(f"Name: {self.name}, Salary: {self.salary}")
# -------------------
# Child Class (inherits from Employee)
# -------------------
class Programmer(Employee): # Inheriting from Employee
company = "ITC Infotech"
def __init__(self, name, salary, language):
super().__init__(name, salary) # Call parent constructor
self.language = language
def showLanguage(self):
print(f"{self.name} is good with {self.language} language")
# ■ Usage
p1 = Programmer("Milan", 40000, "Python")
p1.show() # Inherited method from Employee
p1.showLanguage() # Method from Programmer
# ■ What super() does?
# super().__init__(...) calls the constructor of the parent class,
# so we don’t have to repeat self.name = name, etc.
# ■ Output:
# Name: Milan, Salary: 40000
# Milan is good with Python
# ■ Real-Life Example: Vehicle and Car
class Vehicle:
def start(self):
print("Vehicle started")
```

```
class Car(Vehicle):
def drive(self):
print("Car is driving")
c = Car()
c.start() # Inherited from Vehicle
c.drive() # From Car class
# ■ Types of Inheritance in Python:
# ■ 1. Single Inheritance
# One child inherits from one parent.
# ■ Like: Programmer is an Employee.
class Employee1:
def work(self):
print("Working...")
class Programmer1(Employee1):
def code(self):
print("Coding...")
p = Programmer1()
p.work() # from Employee
p.code() # from Programmer
# ■ 2. Multiple Inheritance
# One child inherits from two or more parents.
# ■ Like: Manager is both an Employee and a Speaker.
class Employee2:
def work(self):
print("Working...")
class Speaker:
def speak(self):
print("Speaking...")
class Manager(Employee2, Speaker): # inherits from both
pass
m = Manager()
m.work() # from Employee2
m.speak() # from Speaker
# ■ 3. Multilevel Inheritance
# Child of a child of a parent.
# ■ Like: Person → Employee → Programmer
class Person:
def breathe(self):
print("Breathing...")
class Employee3(Person):
def work(self):
print("Working...")
class Programmer2(Employee3):
def code(self):
print("Coding...")
p = Programmer2()
p.breathe() # from Person
p.work() # from Employee3
p.code() # from Programmer2
# ■ 4. Hierarchical Inheritance
# Multiple children from one parent.
# ■ Like: Employee is parent, Programmer and Manager are children.
class Employee4:
```

```
def work(self):
print("Working...")
class Programmer3(Employee4):
def code(self):
print("Coding...")
class Manager2(Employee4):
def manage(self):
print("Managing...")
p = Programmer3()
p.work()
p.code()
m = Manager2()
m.work()
m.manage()
# ----------------------------
# Final Example as you asked
# ----------------------------
# Parent Class
class Employee:
company = "ITC"
def show(self):
print(f"The name is {self.name} and the salary is {self.salary}")
# ■ Instead of repeating code for Programmer (like below), we use inheritance
# class Programmer:
# company = "ITC Infotech"
# def show(self):
# print(f"The name is {self.name} and the salary is {self.salary}")
# def showLanguage(self):
# print("The name is {self.name} he is good with {self.language} language")
# ■ We inherit Employee in Programmer
class Programmer(Employee):
company = "ITC Infotech"
def __init__(self, name, salary, language):
self.name = name
self.salary = salary
self.language = language
def showLanguage(self):
print(f"The name is {self.name}, he is good with {self.language} language")
# Creating objects
a = Employee()
a.name = "Amit"
a.salary = 30000
b = Programmer("Rohan", 50000, "Python")
# Output the company names
print(a.company, b.company) # Output: ITC ITC Infotech
# multiple inheritance
class Employee:
company = "ITC"
name = "milan"
salary = 128281
def show(self):
print(f"The name is {self.name} and the salary is {self.salary}")
class Coder:
```

```
language = "python"
def printLanguages(self):
print(f"Out of all languages here is your language {self.language}")
class Programmer(Employee , Coder):
company = "ITC Infotech"
def showLanguage(self):
print(f"The name is {self.name}, he is good with {self.language} language")
# Creating objects
a = Employee()
b = Programmer()
b.show()
b.printLanguages()
b.showLanguage()
print(a.company, b.company) # Output: ITC ITC Infotech
```