📘 Python Inheritance – Detailed Notes with Real-Time Examples

🕰️1. Introduction to Inheritance

- **Inheritance** allows a class (child) to acquire properties and methods of another class (parent).
- Promotes **code reusability** and **modularity**.
- Types: **Single, Multiple, Multilevel, Hierarchical, Hybrid**.

🕰️2. Single Inheritance

- One child inherits from one parent.

Example: Employee System

```
class Person:
    def __init__(self, name):
        self.name = name
    def greet(self):
        print(f"Hello, {self.name}!")

class Employee(Person):
    def __init__(self, name, emp_id):
        super().__init__(name)
        self.emp_id = emp_id
    def show_details(self):
        print(f"Name: {self.name}, ID: {self.emp_id}")

emp1 = Employee("Alice", 101)
emp1.greet()
emp1.show_details()
```

Real-Time: Single inheritance can be used in **Employee Management Systems** where employees inherit personal info from a general `Person` class.

🕰️3. Multiple Inheritance

- Child inherits from **more than one parent**.

Example: Employee + Company System

```python
class Person:
    def greet(self):
        print("Hello from Person")

class Company:
    def company_info(self):
        print("Company: XYZ Corp")

class Employee(Person, Company):
    def work(self):
        print("Employee working")

emp = Employee()
emp.greet()
emp.company_info()
emp.work()
```

Real-Time: Used in **HR software**, where employee objects need info from `Person` and `Company` classes.

🕰️4. Multilevel Inheritance

- Inheritance chain: Parent → Child → Grandchild

Example: Employee → Manager → Senior Manager

```python
class Employee:
    def work(self):
        print("Employee working")

class Manager(Employee):
    def manage(self):
        print("Managing Team")

class SeniorManager(Manager):
    def strategize(self):
        print("Planning Strategy")

senior_mgr = SeniorManager()
senior_mgr.work()
senior_mgr.manage()
senior_mgr.strategize()
```

Real-Time: Corporate hierarchy software where `SeniorManager` inherits employee tasks and managerial duties.

🕰️5. Hierarchical Inheritance

- Multiple children inherit from **one parent**.

Example: School Management System

```
class Person:
    def greet(self):
        print("Hello")

class Student(Person):
    def study(self):
        print("Student Studying")

class Teacher(Person):
    def teach(self):
        print("Teacher Teaching")

stud = Student()
teach = Teacher()
stud.greet()
stud.study()
teach.greet()
teach.teach()
```

Real-Time: School Management Software where multiple roles inherit common `Person` properties.

🕰️6. Hybrid Inheritance

- Combination of multiple types (Single, Multiple, Multilevel, Hierarchical)

Example: Online Course Platform

```
class User:
    def login(self):
        print("User logged in")

class Instructor(User):
    def create_course(self):
        print("Instructor created course")

class Student(User):
    def enroll_course(self):
        print("Student enrolled in course")
```

```
class PremiumStudent(Student, Instructor):
    def access_premium_content(self):
        print("Accessing premium content")

ps = PremiumStudent()
ps.login()
ps.enroll_course()
ps.create_course()
ps.access_premium_content()
```

Real-Time: E-learning platforms where premium users can inherit features of both student and instructor.

🕰️7. Summary

- **Inheritance Types:** Single, Multiple, Multilevel, Hierarchical, Hybrid
- **super():** Access parent class methods/constructors
- **Code Reusability:** Child can reuse parent properties and methods
- **Real-Time Usage:** Employee systems, HR software, Corporate hierarchy, School management, E-learning platforms

---

📥Download Options

- Download as DOCX
- Download as TXT
- Download as PDF