📘 Python Inheritance – Detailed Notes

🕰️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.

Syntax:

```
class Parent:
    pass

class Child(Parent):
    pass
```

Example:

```
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()
```

🕰️3. Multiple Inheritance

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

Example:

```
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()
```

🕰️4. Multilevel Inheritance

- Inheritance chain: Parent → Child → Grandchild

Example:

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

class Employee(Person):
    def work(self):
        print("Working")

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

mgr = Manager()
mgr.greet()
```

```
mgr.work()
mgr.manage()
```

🕰️5. Hierarchical Inheritance

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

Example:

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

class Employee(Person):
    def work(self):
        print("Employee Working")

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

emp = Employee()
stud = Student()
emp.greet()
emp.work()
stud.greet()
stud.study()
```

🕰️6. Hybrid Inheritance

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

Example:

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

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

class Employee(Person, Company):
```

```
    def work(self):
        print("Working")

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

mgr = Manager()
mgr.greet()
mgr.company_info()
mgr.work()
mgr.manage()
```

🕰️7. `super()` Function

- Used to call **parent class methods or constructor**.

Example:

```
class Person:
    def __init__(self, name):
        self.name = name

class Employee(Person):
    def __init__(self, name, emp_id):
        super().__init__(name)
        self.emp_id = emp_id

emp = Employee("Alice", 101)
print(emp.name, emp.emp_id)
```

🕰️8. Real-Time Examples

1️⃣School Management System

```
class Person:
    def __init__(self, name, age):
        self.name = name
        self.age = age

class Student(Person):
    def __init__(self, name, age, grade):
```

```
        super().__init__(name, age)
        self.grade = grade

class Teacher(Person):
    def __init__(self, name, age, subject):
        super().__init__(name, age)
        self.subject = subject

s1 = Student("Alice", 14, "8th")
t1 = Teacher("Mr. Bob", 35, "Math")
print(s1.name, s1.age, s1.grade)
print(t1.name, t1.age, t1.subject)
```

2️⃣Banking System

```
class Account:
    def __init__(self, owner, balance):
        self.owner = owner
        self.balance = balance

class Savings(Account):
    def deposit(self, amount):
        self.balance += amount
        print(f"Deposited {amount}, New Balance: {self.balance}")

class Current(Account):
    def withdraw(self, amount):
        if amount <= self.balance:
            self.balance -= amount
            print(f"Withdrawn {amount}, Remaining Balance: {self.balance}")
        else:
            print("Insufficient Balance")

s = Savings("Alice", 1000)
s.deposit(500)

c = Current("Bob", 2000)
c.withdraw(2500)
```

3️⃣E-commerce User System

```
class User:
    def __init__(self, name):
        self.name = name
```

```
class Customer(User):
    def place_order(self, item):
        print(f"{self.name} placed order for {item}")

class Seller(User):
    def add_product(self, product):
        print(f"{self.name} added {product} for sale")

cust = Customer("Alice")
seller = Seller("Bob")
cust.place_order("Laptop")
seller.add_product("Smartphone")
```

4️⃣ Hospital Management

```
class Person:
    def __init__(self, name):
        self.name = name

class Patient(Person):
    def take_medicine(self, med):
        print(f"{self.name} takes {med}")

class Doctor(Person):
    def prescribe(self, med):
        print(f"{self.name} prescribes {med}")

p = Patient("John")
d = Doctor("Dr. Smith")
p.take_medicine("Paracetamol")
d.prescribe("Antibiotics")
```

5️⃣ Online Course Platform

```
class User:
    def __init__(self, username):
        self.username = username

class Student(User):
    def enroll(self, course):
        print(f"{self.username} enrolled in {course}")

class Instructor(User):
```

```
    def create_course(self, course):
        print(f"{self.username} created course {course}")

stu = Student("Alice")
inst = Instructor("Mr. Bob")
stu.enroll("Python Basics")
inst.create_course("Advanced Python")
```

🕰️9. 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, Banking systems, School management, E-commerce user hierarchy, Hospital, Online learning platforms

---

📥Download Options

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