```
# 1. Single Inheritance Example
class Animal:
    def sound(self):
        print("This animal makes a sound")

class Dog(Animal):
    def bark(self):
        print("Dog barks")

d = Dog()
d.sound()
d.bark()
```

```
This animal makes a sound
Dog barks
```

```
# Example: Vehicle -> Car
class Vehicle:
    def start_engine(self):
        print("Engine started.")

class Car(Vehicle):
    def drive(self):
        print("Car is being driven.")

c = Car()
c.start_engine()
c.drive()
```

```
Engine started.
Car is being driven.
```

```
# 2. Multilevel Inheritance
class A:
    def show(self):
        print("Class A")

class B(A):
    def display(self):
        print("Class B")

class C(B):
    def print_msg(self):
        print("Class C")

obj = C()
obj.show()
obj.display()
obj.print_msg()
```

```
Class A
Class B
Class C
```

```
# Example: LivingBeing -> Animal -> Dog
class LivingBeing:
    def breathe(self):
        print("Breathing...")

class Animal(LivingBeing):
    def eat(self):
        print("Animal eats food.")

class Dog(Animal):
    def bark(self):
        print("Dog barks.")

d = Dog()
d.breathe()
d.eat()
```

```
d.bark()
```

```
Breathing...
Animal eats food.
Dog barks.
```

```
# 11. Multiple Inheritance
class A:
    def feature1(self):
        print("Feature 1 from A")

class B:
    def feature2(self):
        print("Feature 2 from B")

class C(A, B):
    def feature3(self):
        print("Feature 3 from C")

obj = C()
obj.feature1()
obj.feature2()
obj.feature3()
```

```
Feature 1 from A
Feature 2 from B
Feature 3 from C
```

```
# Example: Person + Employee -> Manager
class Father:
    def showf(self,name):
        self.name=name
        print("Name : ",self.name)

class Mother:
    def showm(self,name):
        self.name=name
        print("Name : ",self.name)

class Child(Father, Mother):
    def showc(self,name):
        self.name=name
        print("Name : ",self.name)

c=Child()
c.showf('John')
c.showm('Riya')
c.showc('Alice')
```

```
Name :  John
Name :  Riya
Name :  Alice
```

```
# 21. Hierarchical Inheritance
class Parent:
    def show(self):
        print("Parent class")

class Child1(Parent):
    def feature1(self):
        print("Child1 feature")

class Child2(Parent):
    def feature2(self):
        print("Child2 feature")

obj1 = Child1()
obj2 = Child2()
obj1.show()
obj1.feature1()
obj2.show()
obj2.feature2()
```

```
Parent class
Child1 feature
Parent class
Child2 feature
```

```
# Example: Account -> SavingsAccount, CurrentAccount
class Account:
    def account_info(self):
        print("This is a bank account.")

class SavingsAccount(Account):
    def savings_interest(self):
        print("Savings account gives interest.")

class CurrentAccount(Account):
    def overdraft(self):
        print("Current account allows overdraft.")

sa = SavingsAccount()
ca = CurrentAccount()
sa.account_info()
sa.savings_interest()
ca.account_info()
ca.overdraft()
```

```
This is a bank account.
Savings account gives interest.
This is a bank account.
Current account allows overdraft.
```

```
# 31. Hybrid Inheritance
class A:
    def method_A(self):
        print("Method A")

class B(A):
    def method_B(self):
        print("Method B")

class C(A):
    def method_C(self):
        print("Method C")

class D(B, C):
    def method_D(self):
        print("Method D")

obj = D()
obj.method_A()
obj.method_B()
obj.method_C()
obj.method_D()
```

```
Method A
Method B
Method C
Method D
```

```
# Example: Person -> Student, Employee -> WorkingStudent
class Person:
    def who_am_i(self):
        print("I am a person.")

class Student(Person):
    def student_info(self):
        print("I study.")

class Employee(Person):
    def employee_info(self):
        print("I work.")

class WorkingStudent(Student, Employee):
    def role(self):
        print("I study and work.")

ws = WorkingStudent()
ws.who_am_i()
ws.student_info()
ws.employee_info()
ws.role()
```

```
I am a person.
I study.
I work.
I study and work.
```

```
# 41. Real-world Example - Employee Hierarchy
class Employee:
    def __init__(self, name):
        self.name = name

    def show(self):
        print("Employee:", self.name)

class Manager(Employee):
    def role(self):
        print("Role: Manager")

class Developer(Employee):
    def role(self):
        print("Role: Developer")

m = Manager("Alice")
d = Developer("Bob")
m.show()
m.role()
d.show()
d.role()
```

```
Employee: Alice
Role: Manager
Employee: Bob
Role: Developer
```

```
# 46. Real-world Example - Employee Hierarchy
class Employee:
    def __init__(self, name):
        self.name = name

    def show(self):
        print("Employee:", self.name)

class Manager(Employee):
    def role(self):
        print("Role: Manager")

class Developer(Employee):
```

```
    def role(self):
        print("Role: Developer")

m = Manager("Alice")
d = Developer("Bob")
m.show()
m.role()
d.show()
d.role()
```

```
Employee: Alice
Role: Manager
Employee: Bob
Role: Developer
```

```
# 49. Real-world Example - Employee Hierarchy
class Employee:
    def __init__(self, name):
        self.name = name

    def show(self):
        print("Employee:", self.name)

class Manager(Employee):
    def role(self):
        print("Role: Manager")

class Developer(Employee):
    def role(self):
        print("Role: Developer")

m = Manager("Alice")
d = Developer("Bob")
m.show()
m.role()
d.show()
d.role()
```

```
Employee: Alice
Role: Manager
Employee: Bob
Role: Developer
```

```
# 50. Real-world Example - Employee Hierarchy
class Employee:
    def __init__(self, name):
        self.name = name

    def show(self):
        print("Employee:", self.name)

class Manager(Employee):
    def role(self):
        print("Role: Manager")

class Developer(Employee):
    def role(self):
        print("Role: Developer")

m = Manager("Alice")
d = Developer("Bob")
m.show()
m.role()
d.show()
d.role()
```

```
Employee: Alice
Role: Manager
Employee: Bob
Role: Developer
```