DATA SCIENCE COURSE TUTORIAL # 36

3.23 Inheritance in Object Oriented Programming

What Is Inheritance

Inheritance is an important concept in Object Oriented Programming. It allows one class to take features from another class. The class that gives features is called the parent class. The class that receives features is called the child class. Inheritance helps reuse code, reduce repetition, and create clean program structures.

Why Use Inheritance

Inheritance is useful because it allows classes to share common properties and behaviors. It also improves code organization and helps build advanced systems using simple base classes.

Creating a Parent Class

A parent class is a class whose properties and methods will be inherited by other classes.

Example:

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

    def show(self):
        print("Name:", self.name)
        print("Age:", self.age)
```

Creating a Child Class

A child class inherits from a parent class using parentheses.

Example:

```
class Student(Person):
    pass
```

Here, Student is a child class and Person is the parent class.

Using Inherited Properties and Methods

The child class automatically gets all attributes and methods of the parent class.

Example:

```
s1 = Student("Ali", 20)
s1.show()
```

Output:

```
Name: Ali
Age: 20
```

Adding New Features in Child Class

A child class can have its own methods along with inherited ones.

Example:

```
class Student(Person):
    def study(self):
        print(self.name, "is studying.")

s1 = Student("Aisha", 22)
s1.show()
s1.study()
```

Output:

```
Name: Aisha
Age: 22
Aisha is studying.
```

Using init in Child Class

If the child class has its own **init** function, you must call the parent constructor using super().

Example:

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

```
    def show_details(self):
        print("Name:", self.name)
        print("Age:", self.age)
        print("Roll No:", self.roll)
```

Example of Creating Object:

```
s1 = Student("Ali", 20, 101)
s1.show_details()
```

Output:

```
Name: Ali
Age: 20
Roll No: 101
```

Types of Inheritance

Python supports different types of inheritance.

Single Inheritance One child class inherits from one parent class.

Multilevel Inheritance A class inherits from a child class which itself has a parent class.

Multiple Inheritance A class inherits from more than one parent class.

Example of Multilevel Inheritance

```
class A:
    def funA(self):
        print("Function A")

class B(A):
    def funB(self):
        print("Function B")

class C(B):
    def funC(self):
        print("Function C")

obj = C()
obj.funA()
obj.funB()
obj.funC()
```

Output:

```
Function A
Function B
Function C
```

Example of Multiple Inheritance

```
class A:
    def funA(self):
        print("Function A")

class B:
    def funB(self):
        print("Function B")

class C(A, B):
    pass

obj = C()
obj.funA()
obj.funB()
```

Output:

```
Function A
Function B
```

Benefits of Inheritance

Inheritance reduces code repetition. It supports code reuse. It helps build structured programs. It simplifies maintenance by connecting related classes.

Summary

- Inheritance allows one class to take features from another class.
- Parent class provides data and methods. Child class receives them.
- super() is used to call the parent constructor.
- Python supports single, multilevel and multiple inheritance.
- Inheritance helps build clean, reusable and organized code.