DATA SCIENCE COURSE TUTORIAL # 37

3.24 Polymorphism in Object Oriented Programming

What Is Polymorphism

Polymorphism means many forms. It allows different classes to use the same method name but perform different actions. Polymorphism helps write flexible and reusable code because the same method call can behave differently depending on the object.

Why Polymorphism Is Important

Polymorphism improves readability. It reduces repeated code. It allows you to create a common interface for different classes. It makes programs easier to extend and maintain.

Polymorphism with Methods

Different classes can have methods with the same name, but each method can perform a different task.

Example:

```
class Cat:
    def sound(self):
        print("Meow")

class Dog:
    def sound(self):
        print("Bark")

c = Cat()
d = Dog()

c.sound()
d.sound()
```

Output:

```
Meow
Bark
```

Although both classes have a method named sound, each one prints a different output.

Polymorphism Using a Common Function

You can use a single function to work with different classes as long as they have the same method name.

Example:

```
def animal_sound(animal):
    animal.sound()

animal_sound(Cat())
animal_sound(Dog())
```

Output:

```
Meow
Bark
```

This shows how one function can work with multiple object types.

Polymorphism with Inheritance

In inheritance, child classes can override parent class methods. This is also a form of polymorphism.

Example:

```
class Person:
    def show(self):
        print("I am a person.")

class Student(Person):
    def show(self):
        print("I am a student.")

p = Person()
s = Student()

p.show()
s.show()
```

Output:

```
I am a person.
I am a student.
```

The child class Student changes the behavior of the parent show method.

Method Overriding

Method overriding happens when the child class replaces a method from the parent class with its own version.

Example:

```
class Shape:
    def area(self):
        print("Area not defined.")

class Circle(Shape):
    def area(self):
        print("Area of circle is pi r r.")
```

Calling the same method on different objects produces different results.

Polymorphism with Built In Functions

Some built in functions in Python also show polymorphism.

Example:

```
print(len("Hello"))
print(len([1, 2, 3, 4]))
```

Output:

```
5
4
```

The len function behaves differently based on the object type.

Why Use Polymorphism

Polymorphism helps to create modular and flexible code. It allows one interface to be used for many types of objects. It supports method overriding and dynamic behavior.

Summary

- Polymorphism means many forms.
- Same method name can behave differently in different classes.
- Polymorphism works through method overriding.

- It supports inheritance based behavior changes.
- It improves flexibility, readability and code reuse.