وراثت (Inheritance) در پایتون

وراثت یکی از ویژگی‌های اصلی برنامه‌نویسی شی‌گراست که به کلاس‌ها اجازه می‌دهد از ویژگی‌ها و متدهای کلاس دیگر استفاده کنند. با استفاده از وراثت می‌توانیم سلسله‌مراتبی از کلاس‌ها ایجاد کنیم که موجب **کاهش تکرار کد** و **استفاده مجدد از کد** می‌شود.

مفهوم وراثت در پایتون

ارث‌بری در پایتون چیست؟

وراثت فرآیندی است که به یک کلاس **فرزند** اجازه می‌دهد از ویژگی‌ها و متدهای **کلاس والد (Base Class)** خود استفاده کند.
در پایتون، کلاس‌ها به راحتی می‌توانند از یک کلاس دیگر ارث‌بری کنند و از ویژگی‌ها و متدهای آن استفاده نمایند.

نحوه استفاده از وراثت

در پایتون، برای ارث‌بری از یک کلاس دیگر، باید نام کلاس والد را در داخل پرانتز پس از نام کلاس فرزند بنویسیم.

ساختار عمومی وراثت

```
class Parent:  # کلاس والد
    def __init__(self):
        self.name = "کلاس والد"

    def greet(self):
        print("سلام از کلاس والد!")

class Child(Parent):  # کلاس فرزند که از کلاس Parent ارث‌بری می‌کند
    def __init__(self):
        super().__init__()  # فراخوانی متد __init__ از کلاس والد
        self.name = "کلاس فرزند"

    def greet(self):  # بازنویسی متد greet در کلاس فرزند
        print("سلام از کلاس فرزند!")

# ایجاد شیء از کلاس فرزند
child = Child()
print(child.name)  # خروجی: کلاس فرزند
child.greet()  # خروجی: سلام از کلاس فرزند!
```

نکات کلیدی:

- **کلاس فرزند** از **کلاس والد** ویژگی‌ها و متدها را به ارث می‌برد.
- `super().__init__()` برای فراخوانی **متد** `__init__` کلاس والد استفاده می‌شود و به این ترتیب می‌توانیم ویژگی‌های کلاس والد را در کلاس فرزند به‌روزرسانی کنیم.
- در صورت وجود **متد مشابه** در کلاس فرزند، متد فرزند **متد والد را بازنویسی (override)** می‌کند.

مزایای استفاده از وراثت

کاهش تکرار کد:

با استفاده از وراثت، می‌توانیم کدهای مشترک را در یک کلاس پایه قرار دهیم و سپس آن‌ها را در کلاس‌های فرزند بدون نیاز به تکرار دوباره کد، استفاده کنیم. این امر باعث **کاهش پیچیدگی و افزایش خوانایی کد** می‌شود.

مدیریت بهتر و گسترش ساده‌تر:

با ایجاد سلسله‌مراتبی از کلاس‌ها، می‌توانیم کد خود را به بخش‌های کوچک‌تر تقسیم کنیم و برای افزودن ویژگی‌ها یا تغییرات جدید به راحتی **کلاس‌های فرزند را گسترش دهیم**.

مثال ساده از وراثت

مثال 1: کلاس‌های حیوانات

```
class Animal:  # کلاس والد
    def __init__(self, name):
        self.name = name

    def sound(self):
        print(f"{self.name} صدای خاصی تولید می‌کند.")

class Dog(Animal):  # کلاس فرزند از Animal
    def __init__(self, name, breed):
        super().__init__(name)  # فراخوانی متد __init__ از کلاس والد
        self.breed = breed

    def sound(self):  # بازنویسی متد sound
        print(f"{self.name} پارس می‌کند.")

class Cat(Animal):  # کلاس فرزند از Animal
    def __init__(self, name, color):
        super().__init__(name)
        self.color = color

    def sound(self):  # بازنویسی متد sound
        print(f"{self.name} میو میو می‌کند.")

# ایجاد اشیاء
dog = Dog("پودل" ,"سگ")
cat = Cat("سفید" ,"گربه")

dog.sound()  # خروجی: سگ پارس می‌کند.
cat.sound()  # خروجی: گربه میو میو می‌کند.
```

نکات:

- در این مثال، کلاس `Animal` به‌عنوان کلاس **والد** ویژگی‌های `name` و متد `sound` را تعریف می‌کند.
- کلاس‌های `Dog` و `Cat` از کلاس `Animal` ارث‌بری می‌کنند و متد `sound` را بازنویسی می‌کنند تا صدای مخصوص خود را تولید کنند.
- از `super()` برای فراخوانی متد `__init__` کلاس والد استفاده شده است.

نحوه ایجاد و استفاده از وراثت در پایتون

ارث‌بری از یک کلاس پایه (Base Class)

زمانی که می‌خواهید از ویژگی‌ها و متدهای کلاس پایه استفاده کنید، می‌توانید از کلمه کلیدی `class` به‌همراه نام کلاس پایه در داخل پرانتز استفاده کنید.

بازنویسی متدها (Method Overriding)

در صورتی که بخواهید متدهای کلاس پایه را تغییر دهید یا عملکرد متفاوتی برای آن‌ها پیاده‌سازی کنید، می‌توانید **متدهای کلاس فرزند را بازنویسی** کنید.

ارث‌بری چندگانه (Multiple Inheritance)

پایتون به شما این امکان را می‌دهد که یک کلاس فرزند از چندین کلاس پایه ارث‌بری کند. این ویژگی به‌عنوان **وراثت چندگانه** شناخته می‌شود.

مثال ارث‌بری چندگانه

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

    def speak(self):
        print(f"{self.name} صدای خاصی تولید می‌کند.")

class Swimmer:
    def swim(self):
        print(f"{self.name} در آب شنا می‌کند.")

class Fish(Animal, Swimmer):  # وراثت چندگانه از Animal و Swimmer
    def __init__(self, name):
        Animal.__init__(self, name)

# ایجاد شیء
fish = Fish("ماهی")
fish.speak()  # خروجی: ماهی صدای خاصی تولید می‌کند.
fish.swim()   # خروجی: ماهی در آب شنا می‌کند.
```

جمع‌بندی وراثت در پایتون

- **وراثت** امکان استفاده مجدد از کد و سازماندهی بهتر کدها را فراهم می‌کند.
- در پایتون، می‌توانید **کلاس فرزند** را از **کلاس پایه** ارث‌بری کنید.
- با **بازنویسی متدها**، می‌توانید رفتار متدهای کلاس پایه را در کلاس‌های فرزند تغییر دهید.
- **وراثت چندگانه** به شما این امکان را می‌دهد که از چندین کلاس پایه به‌طور هم‌زمان ارث‌بری کنید.

📝 تمرین برای شما:

یک کلاس `Vehicle` ایجاد کنید که ویژگی `speed` و متد `move` را داشته باشد. سپس از آن یک کلاس `Car` و یک کلاس `Bike` ایجاد کنید که رفتار خود را از کلاس `Vehicle` ارث‌بری کنند.