متدهای ارثی و بازنویسی متدها (Overriding Methods)

در برنامه‌نویسی شی‌گرا، یکی از ویژگی‌های مهم وراثت، امکان **بازنویسی متدها (Overriding)** است. این ویژگی به شما اجازه می‌دهد که متدهای کلاس پایه را در کلاس‌های فرزند تغییر دهید تا رفتار خاصی برای آن‌ها پیاده‌سازی کنید.

تفاوت بین ارث‌بری و بازنویسی متدها (Overriding)

- **ارث‌بری (Inheritance):**
  در ارث‌بری، کلاس فرزند از ویژگی‌ها و متدهای کلاس والد استفاده می‌کند. این فرآیند به شما این امکان را می‌دهد که متدها و ویژگی‌های کلاس پایه را در کلاس‌های فرزند به ارث ببرید.
- **بازنویسی متد (Overriding):**
  بازنویسی به این معناست که در کلاس فرزند، متدهای کلاس پایه تغییر می‌کنند یا **عملکرد جدیدی** برای آن‌ها تعریف می‌شود. یعنی کلاس فرزند می‌تواند متدهای کلاس والد را دوباره تعریف کند تا رفتار متفاوتی داشته باشند.

در واقع، زمانی که متدی در کلاس فرزند تعریف می‌کنید که با نام متد کلاس والد مشابه است، پایتون آن متد را **بازنویسی** می‌کند.

نحوه بازنویسی متدهای کلاس پایه در کلاس‌های فرزند

برای بازنویسی یک متد در کلاس فرزند، کافی است که یک متد با همان نام و پارامترها در کلاس فرزند تعریف کنید. این متد در کلاس فرزند **جایگزین** متد کلاس والد خواهد شد.

مثال بازنویسی متد

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

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

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

    def speak(self):  # بازنویسی متد speak در کلاس فرزند
        print(f"{self.name} پارس می‌کند.")

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

    def speak(self):  # بازنویسی متد speak در کلاس فرزند
        print(f"{self.name} میو میو می‌کند.")

# ایجاد اشیاء
```

```
dog = Dog("پودل" ,"سگ")
cat = Cat("سفید" ,"گربه")

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

در این مثال، هر دو کلاس `Dog` و `Cat` متد `speak` را از کلاس پایه `Animal` ارث می‌برند، اما در هر یک از کلاس‌های فرزند، این متد بازنویسی می‌شود تا رفتار خاص خود را داشته باشد.

تفاوت بین ارث‌بری و بازنویسی متدها

- **ارث‌بری:**
  کلاس فرزند به‌طور پیش‌فرض ویژگی‌ها و متدهای کلاس پایه را **دریافت می‌کند**.
- **بازنویسی متد (Overriding):**
  زمانی که شما بخواهید رفتار یک متد را در کلاس فرزند **تغییر دهید** یا **بازنویسی کنید**، آن متد را در کلاس فرزند دوباره تعریف می‌کنید. در این صورت، متد کلاس فرزند به‌جای متد کلاس پایه اجرا خواهد شد.

استفاده از متد `super()` برای فراخوانی متدهای کلاس پایه

در هنگام بازنویسی متدها، اگر بخواهید به متدهای کلاس پایه دسترسی داشته باشید (برای مثال برای فراخوانی آن‌ها یا استفاده از منطق کلاس پایه)، می‌توانید از متد `super()` استفاده کنید. این متد به شما امکان می‌دهد که به **کلاس والد** دسترسی داشته باشید.

مثال استفاده از `super()`

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

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

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

    def speak(self):  # بازنویسی متد speak
        super().speak()  # فراخوانی متد speak از کلاس والد
        print(f"{self.name} پارس می‌کند.")

# ایجاد شیء از کلاس Dog
dog = Dog("پودل" ,"سگ")
dog.speak()
```

خروجی:

```
سگ صدای خاصی تولید می‌کند.
سگ پارس می‌کند.
```

در این مثال:

- متد `speak()` در کلاس `Dog` بازنویسی شده است.
- با استفاده از `super().speak()`، ابتدا متد `speak()` کلاس پایه فراخوانی می‌شود و سپس رفتار خاص کلاس فرزند (یعنی پارس کردن سگ) اجرا می‌شود.

نکات مهم در بازنویسی متدها

- **بازنویسی متدها** معمولاً برای تغییر یا افزودن رفتارهای جدید به متدهای کلاس پایه استفاده می‌شود.
- `super()` به شما این امکان را می‌دهد که بدون نیاز به ارجاع مستقیم به نام کلاس والد، از متدهای آن استفاده کنید. این به‌ویژه در **وراثت چندگانه** بسیار مفید است.
- اگر شما یک متد را در کلاس فرزند بازنویسی نکنید، متد کلاس پایه به‌طور پیش‌فرض فراخوانی می‌شود.

تمرین برای شما:

یک کلاس `Shape` ایجاد کنید که دارای یک متد `area` برای محاسبه مساحت باشد. سپس دو کلاس `Circle` و `Rectangle` ایجاد کنید که متد `area` را بازنویسی کنند تا مساحت دایره و مستطیل را محاسبه کنند.