Reflection

بازتاب (Reflection) در برنامه‌نویسی

بازتاب (Reflection) در برنامه‌نویسی شیءگرا به قابلیتی گفته می‌شود که برنامه می‌تواند در زمان اجرا، ساختار خودش را بررسی کرده و حتی تغییر دهد. این ویژگی به توسعه‌دهندگان اجازه می‌دهد تا اطلاعاتی درباره‌ی کلاس‌ها، اشیاء، متدها و ویژگی‌های آن‌ها استخراج کنند و حتی ویژگی‌های جدیدی به اشیاء اضافه کنند. در پایتون، این قابلیت به کمک توابع داخلی مانند `getattr` ، `setattr` و `hasattr` پیاده‌سازی می‌شود.

دسترسی به ویژگی‌های شیء با استفاده از `getattr`

تابع `getattr` به ما این امکان را می‌دهد که مقدار یک ویژگی را از یک شیء دریافت کنیم، بدون اینکه به‌طور مستقیم نام آن را بدانیم. این تابع معمولاً زمانی کاربرد دارد که نام ویژگی به‌صورت رشته‌ای ذخیره شده باشد و در زمان اجرا مشخص شود.

مثال: استفاده از `getattr` برای دریافت مقدار یک ویژگی

Copy Python

```python
class Student:
    def __init__(self, name, grade):
        self.name = name
        self.grade = grade

student = Student("Ali", "A")
attribute = input("Enter the attribute name:")
print(getattr(student, attribute, "Attribute not found"))
```

توضیح:

- این کد از ورودی کاربر نام یک ویژگی را دریافت می‌کند.
- با استفاده از `getattr` مقدار ویژگی خوانده می‌شود. اگر ویژگی وجود نداشته باشد، مقدار پیش‌فرض "ویژگی موردنظر یافت نشد" برگردانده می‌شود.

افزودن یا تغییر مقدار ویژگی با استفاده از `setattr`

تابع `setattr` به ما این امکان را می‌دهد که مقدار یک ویژگی را تغییر دهیم یا ویژگی جدیدی را به یک شیء اضافه کنیم.

مثال: افزودن ویژگی جدید به یک شیء

```
class Car:
    def __init__(self, brand):
        self.brand = brand

car = Car("Toyota")
setattr(car, "color", "Red")
print(f"Brand: {car.brand}, Color: {car.color}")
```

توضیح:

- ابتدا یک شیء از کلاس `Car` ساخته می‌شود.
- سپس با `setattr` ویژگی `color` اضافه می‌شود.
- در نهایت مقدار ویژگی‌های شیء چاپ می‌شود.

تعریف پویای متدها با `__getattr__`

گاهی اوقات می‌خواهیم بدون اینکه مستقیماً یک متد را تعریف کنیم، امکان دسترسی به آن را ایجاد کنیم. این کار با بازنویسی متد `__getattr__` امکان‌پذیر است.

مثال: شبیه‌سازی متدها در زمان اجرا

```
class Data:
    def __init__(self):
        self.values = {"height": 180, "weight": 75, "age": 25}

    def __getattr__(self, attr):
        if attr.startswith("get_"):
```

```
            key = attr[4:]
            return self.values.get(key, "Attribute not found")
        raise AttributeError(f"'{attr}' Not present in the class")

person = Data()
print(person.get_height())  # 180
print(person.get_weight())  # 75
print(person.get_age())     # 25
print(person.get_salary())  # Attribute not found
```

توضیح:

- کلاس `Data` دارای یک دیکشنری داخلی از ویژگی‌ها است.
- متد `__getattr__` بررسی می‌کند که اگر نام ویژگی با `get_` شروع شود، مقدار مربوطه از دیکشنری برگردانده شود.
- اگر ویژگی نامعتبر باشد، پیام مناسبی نمایش داده می‌شود.

جمع‌بندی

- `getattr` امکان دریافت مقدار ویژگی را به‌صورت پویا فراهم می‌کند.
- `setattr` به ما اجازه می‌دهد ویژگی جدیدی اضافه کنیم یا مقدار ویژگی موجود را تغییر دهیم.
- با بازنویسی `__getattr__` می‌توان متدهایی را در زمان اجرا شبیه‌سازی کرد.

این تکنیک‌ها در توسعه‌ی فریمورک‌های پویا، تحلیل داده‌ها و برنامه‌هایی که نیاز به پردازش انعطاف‌پذیر دارند بسیار مفید هستند.