دنیای اسرارآمیز کلاس‌ها(Magic Methods)

تا به حال با متدهایی که نام‌شان با دو خط پایین و بالا (__) مشخص می‌شود برخورد کرده‌اید؟ احتمالاً این شکل خاص باعث شده تا سوالات زیادی در ذهن شما ایجاد شود که این متدها دقیقاً چه کارهایی انجام می‌دهند. در این بخش قصد داریم به دنیای جذاب این متدها که به نام **Dunder Methods** یا **Magic Methods** شناخته می‌شوند، بپردازیم.

شروع با یک مثال ساده

کلاسی به نام `Movie` داریم که اطلاعاتی مانند نام فیلم و سال انتشار آن را نگهداری می‌کند:

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

حالا اگر یک شیء از این کلاس بسازیم و از تابع `dir()` استفاده کنیم، به لیستی از ویژگی‌ها و متدها دست پیدا می‌کنیم:

```
favorite_movie = Movie('Inception', 2010)
print(dir(favorite_movie))
```

خروجی مشابه این خواهد بود:

```
['__class__', '__delattr__', '__dict__', '__dir__', '__doc__', '__eq__',
```

همانطور که مشاهده می‌کنید، علاوه بر ویژگی‌هایی که خودمان تعریف کرده‌ایم (`name` و `release_year`)، متدهای زیادی در لیست وجود دارند که به‌طور پیش‌فرض در کلاس‌های پایتون تعریف شده‌اند. حالا سوال این است که این متدها چه کارهایی انجام می‌دهند؟ اجازه دهید با چند مورد از این متدها آشنا شویم.

متد str: چاپ اطلاعات به شیوه‌ای دلپذیر

وقتی شیء خود را چاپ می‌کنید، پایتون به‌طور پیش‌فرض آدرسی از شیء را نمایش می‌دهد که شاید برای شما خیلی مفید نباشد. با استفاده از متد `__str__` می‌توانیم نحوه نمایش یک شیء را سفارشی کنیم:

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

    def __str__(self):
        return f"Movie: {self.name} ({self.release_year})"
```

حال اگر شیء `favorite_movie` را چاپ کنیم:

```
favorite_movie = Movie('Inception', 2010)
print(favorite_movie)
```

خروجی به این صورت خواهد بود:

```
Movie: Inception (2010)
```

متد lt: مقایسه شیءها با استفاده از عملگرها

فرض کنید می‌خواهید دو فیلم را با هم مقایسه کنید تا ببینید کدام فیلم قدیمی‌تر است. به‌طور پیش‌فرض، پایتون نمی‌داند که چگونه این مقایسه را انجام دهد و خطای زیر را به نمایش می‌گذارد:

```
movie1 = Movie('The Matrix', 1999)
movie2 = Movie('The Dark Knight', 2008)
print(movie1 < movie2)  # Error!
```

برای اینکه بتوانیم از عملگر `<` استفاده کنیم، باید متد `__lt__` را بازنویسی کنیم:

```python
class Movie:
    def __init__(self, name, release_year):
        self.name = name
        self.release_year = release_year

    def __lt__(self, other):
        return self.release_year < other.release_year
```

اکنون می‌توانیم فیلم‌ها را بر اساس سال انتشار مقایسه کنیم:

```python
movie1 = Movie('The Matrix', 1999)
movie2 = Movie('The Dark Knight', 2008)
print(movie1 < movie2)  # True
```

متد call: امکان فراخوانی شیءها مانند توابع

گاهی اوقات ممکن است بخواهید با استفاده از پرانتز (مثل توابع) یک شیء را فراخوانی کنید. برای این کار باید متد `__call__` را تعریف کنید:

```python
class Movie:
    def __init__(self, name, release_year):
        self.name = name
        self.release_year = release_year

    def __call__(self):
        return f"Welcome to {self.name}!"
```

اکنون می‌توانید شیء `movie` را مانند یک تابع فراخوانی کنید:

```python
movie = Movie('The Matrix', 1999)
print(movie())  # Welcome to The Matrix!
```

در این درسنامه با تعدادی از متدهای جادویی پایتون آشنا شدیم که به شما این امکان را می‌دهند تا شیءها را بهتر مدیریت کرده و تجربه‌ی برنامه‌نویسی‌تان را ارتقا دهید. در درس‌های بعدی، با دیگر متدهای جادویی

کاربردی آشنا خواهیم شد.