چند ریختی (Polymorphism)

زمانی ما از چند ریختی‌ها استفاده می‌کنیم که بخواهیم یک تابع در کلاس‌های مختلف کار متفاوتی انجام دهد. برای مثال اگر ما یک کلاس به نام شکل داشته باشیم و کلاس‌های فرزند آن کلاس مستطیل، دایره و مثلث باشند، و درون هریک از کلاس‌های فرزندمان یک تابع با نام رسم شکل وجود داشته باشد، اتفاقی که میافتد این است که نام تابع در همه کلاس‌ها یکسان است اما در هر یک از کلاس‌ها متفاوت عمل می‌کند.
در اصطلاح اگر تابع جدید با همان پارامتر های ورودی تابع قبلی باشد میگوییم این تابع override شده و اگر در کنار همنام بودن ، پارامترهای ورودی تغییر کند و متفاوت باشد میگوییم overload اتفاق افتاده است که هر دو قسمتی از پلی مورفیزم هستند.
در حقیقت چند ریختی به معنی استفاده از توابع با نام های یکسان و فرم های متفاوت است.

در مثال زیر مشاهده میکنیم که متد flight در کلاس والد به همراه متد intro تعریف شده است. اما دو فرزندی که از کلاس Bird استفاده میکنند با حفظ نام همان متد والد اما عملیات متفاوتی را انجام میدهند که به این فرآیند پلی مورفیزم یا چندریختی میگویند.

```
class Bird:
    def intro(self):
        print("There are many types of birds.")

    def flight(self):
        print("Most of the birds can fly but some cannot.")


class Sparrow(Bird):
    def flight(self):
        print("Sparrows can fly.")


class Hen(Bird):
    def flight(self):
        print("Hens cannot fly.")
```

انتزاع (Abstraction)

کلاسی که شامل یک یا چند متد abstract باشد را کلاس abstract می گویند. یک متد abstract متدی است که اعلان (declaration) می شود ولی شامل بدنه پیاده سازی شده نیست. کلاس abstract معمولا به عنوان base کلاس مورد استفاده قرار می گیرند. abstract ها زمانی استفاده می شوند که بخواهیم برای یک متد چندین پیاده سازی مختلف انجام دهیم یا کلاس هایی وجود دارند که متد خاصی را به شیوه های مختلف پیاده سازی می کنند.
در این ترم با این تعریف انتزاع کاری نداریم و صرفا این نکته حائز اهمیت است که در محیط واقعی کار و برنامه نویسی ما لزوما از پیاده سازی همه چیز با خبر نیستیم و صرفا از بعضی از متدها یا دستورات استفاده میکنیم. برای مثال دستور len در پایتون. ما میدانیم که این دستور چه کاری انجام میدهد اما از جزییات پیاده سازی آن با خبر نیستیم.
در فضای کاری هم این اتفاق میافتد که تیم های مختلف در حال پیاده سازی بخش های مختلف پروژه هستند و صرفا نیازهای همدیگر را از طریق بعضی متدها برطرف میکنند ولی تیم A از جزییات پیاده سازی تیم B که مسئولیت تهیه زیرساخت را برعهده داشته ، باخبر نیست.

Enum

در پایتون ماژول enum برای ایجاد کردن مجموعه ای از متغیرها است. این متغیرها ویژگی های خاصی خواهند داشت مثل منحصر به فرد بودن و غیرقابل تغییر بودن.
به منظور پیاده سازی مفهوم Enum در پایتون از ماژول enum استفاده می کنیم. این ماژول به صورت پیشفرض در پایتون موجود است.

در ابتدا ماژول enum را وارد محیط برنامه می نماییم:

```
from enum import Enum
```

حال متغیرهایی که میخواهیم در برنامه خود بسازیم را در یک کلاسی که از Enum ارث بری میکند تعریف میکنیم. نکته مهم این است که این متغیرها در تعریف میبایست یک مقدار اولیه هم به آنها داده شود.
بصورت قراردادی درنظر میگیریم که متغیرهای تعریف شده به این شکل را با حروف بزرگ تعریف کنیم.
برای مثال میخواهیم متغیرهای جهت برای استفاده در پروژه را تعریف کنیم:

```
class Direction(Enum):
    VERTICAL = 1
    HORIZONTAL = 2
```

در زبان برنامه نویسی پایتون به جای اینکه به این متغیرها بصورت دستی مقدار دهیم میتوانیم از دستور auto() استفاده کنیم که در ابتدا لازم است ماژول آن را بالای برنامه اضافه کنیم:

```
from enum import Enum, auto


class Direction(Enum):
    VERTICAL = auto()
    HORIZONTAL = auto()
```