انتزاع (Abstraction) در پایتون

انتزاع در شی‌گرایی به معنای **مخفی کردن پیچیدگی‌ها** و **نمایش تنها ویژگی‌های ضروری** به کاربر است. در این روش، جزئیات پیاده‌سازی از دید کاربر پنهان می‌ماند و فقط رابط‌ها و ویژگی‌های اصلی برای استفاده باقی می‌ماند. هدف از انتزاع این است که پیچیدگی‌های سیستم کاهش یابد و کاربران تنها با بخش‌های حیاتی و مورد نیاز ارتباط برقرار کنند.

در پایتون، برای پیاده‌سازی انتزاع، می‌توان از **کلاس‌های انتزاعی** (Abstract Classes) و **متدهای انتزاعی** (Abstract Methods) استفاده کرد. این قابلیت از ماژول (Abstract Base Class) `abc` پشتیبانی می‌شود.

تعریف انتزاع به عنوان مخفی کردن پیچیدگی‌ها

انتزاع باعث می‌شود که جزئیات پیاده‌سازی از کاربر پنهان شود و تنها اطلاعات و ویژگی‌های ضروری برای استفاده در دسترس باشند. این کار به مدیریت بهتر کدها، کاهش پیچیدگی و افزایش قابلیت نگهداری برنامه کمک می‌کند.

برای مثال، اگر بخواهیم یک سیستم مدیریت وسایل نقلیه بسازیم، می‌توانیم کلاس‌های مختلف برای انواع وسایل نقلیه مثل **ماشین** و **موتور سیکلت** ایجاد کنیم. این کلاس‌ها می‌توانند برخی ویژگی‌های مشترک مثل **حرکت** و **ترمز** را از کلاس انتزاعی به ارث ببرند، بدون اینکه جزئیات پیاده‌سازی برای هر وسیله نقلیه به کاربر نشان داده شود.

استفاده از کلاس‌های انتزاعی (Abstract Classes) و متدهای انتزاعی (Abstract Methods)

کلاس‌های انتزاعی در پایتون با استفاده از ماژول `abc` ایجاد می‌شوند. یک کلاس انتزاعی می‌تواند حاوی متدهایی باشد که **بازنویسی** باید در کلاس‌های فرزند انجام شود. این متدها به عنوان **متدهای انتزاعی** شناخته می‌شوند و در کلاس پایه پیاده‌سازی نمی‌شوند.

برای ایجاد یک کلاس انتزاعی باید از **کلمه‌کلیدی** `ABC` به همراه `abstractmethod` استفاده کرد.

نحوه تعریف کلاس انتزاعی و متدهای انتزاعی

1. **ایجاد کلاس انتزاعی:**
   برای ایجاد یک کلاس انتزاعی باید از کلمه‌کلیدی `ABC` ارث‌بری کرده و از `abstractmethod` برای متدهای انتزاعی استفاده کنید.
2. **تعریف متدهای انتزاعی:**
   متدهای انتزاعی متدهایی هستند که در کلاس انتزاعی تعریف می‌شوند، اما پیاده‌سازی آن‌ها در کلاس‌های فرزند انجام می‌شود.

مثال پیاده‌سازی انتزاع در پایتون

در این مثال، یک کلاس انتزاعی به نام `Vehicle` ایجاد کرده‌ایم که دارای یک متد انتزاعی `start_engine` است. سپس این متد را در کلاس‌های `Car` و `Bike` بازنویسی می‌کنیم.

```
from abc import ABC, abstractmethod

class Vehicle(ABC):  # ایجاد کلاس انتزاعی
    @abstractmethod
    def start_engine(self):  # متد انتزاعی که باید در کلاس‌های فرزند بازنویسی شود
        pass
```

```python
class Car(Vehicle):  # کلاس فرزند از Vehicle
    def start_engine(self):  # بازنویسی متد start_engine
        print("ماشین در حال حرکت است!")

class Bike(Vehicle):  # کلاس فرزند از Vehicle
    def start_engine(self):  # بازنویسی متد start_engine
        print("موتور در حال حرکت است!")

# ایجاد اشیاء از کلاس‌های فرزند
car = Car()
bike = Bike()

# استفاده از متد انتزاعی در کلاس‌های فرزند
car.start_engine()  # خروجی: ماشین در حال حرکت است!
bike.start_engine()  # خروجی: موتور در حال حرکت است!
```

توضیحات:

- **کلاس** `Vehicle` یک کلاس انتزاعی است که متد `start_engine` را به صورت انتزاعی تعریف کرده است.
- کلاس‌های `Car` و `Bike` این متد را به طور خاص برای خود بازنویسی کرده‌اند.
- وقتی شیء از این کلاس‌ها ساخته می‌شود، متد `start_engine` به صورت خاص برای هر نوع وسیله نقلیه اجرا می‌شود.

مزایای استفاده از انتزاع

1. **مخفی کردن پیچیدگی‌ها:**
   با استفاده از انتزاع، می‌توان جزئیات پیچیده را از کاربر پنهان کرد و تنها رابط‌های ساده و کاربردی را نمایش داد.
2. **افزایش قابلیت نگهداری و گسترش کد:**
   وقتی متدهای انتزاعی در کلاس‌های پایه تعریف می‌شوند، تغییرات در جزئیات پیاده‌سازی در کلاس‌های فرزند امکان‌پذیر است بدون اینکه کد اصلی تحت تاثیر قرار گیرد.
3. **استانداردسازی کد:**
   انتزاع به ما این امکان را می‌دهد که کدهای مشابهی را در کلاس‌های مختلف پیاده‌سازی کنیم و آن‌ها را با یک رابط مشترک به کار ببریم.

📝 تمرین برای شما:

یک سیستم بانکداری بسازید که از کلاس انتزاعی `Account` استفاده کند. در این سیستم، یک متد انتزاعی به نام `withdraw` تعریف کنید. سپس کلاس‌های فرزند مانند `SavingAccount` و `CheckingAccount` را ایجاد کنید که این متد را برای عملیات برداشت به طور خاص پیاده‌سازی کنند.

این مفهوم به شما کمک می‌کند تا بتوانید سیستم‌های پیچیده را با طراحی‌هایی ساده و قابل گسترش پیاده‌سازی کنید.