وراثت و تجرید در پایتون
(Inheritance and Abstraction)

وراثت یکی از اصول اصلی برنامه‌نویسی شیءگرا (OOP) است که به امکان ارث‌بری کلاس‌ها از یکدیگر اشاره دارد. در وراثت، کلاس‌ها می‌توانند ویژگی‌ها (Attribute) و روش‌های (Method) کلاس‌های دیگر را به ارث ببرند و از آن‌ها استفاده کنند. این امر باعث می‌شود که تکرار کد کاهش یابد و ساختار کد مناسب‌تر و قابل مدیریت‌تر شود.

وراثت در پایتون به کمک کلاس‌های زیرکلاس (subclass) از وجود کلاس‌های پایه (base class) یا سوپرکلاس (superclass) پیاده‌سازی می‌شود. کلاس زیرکلاس، کلاسی است که از کلاس پایه ارث‌بری می‌کند و به ویژگی‌ها و روش‌های آن دسترسی دارد.

یک مثال ساده را در زبان برنامه‌نویسی پایتون در نظر بگیرید:

```
class BirdFamily:
    def __init__(self, name):
        self.name = name
    def fly(self):
        print(self.name, "can fly.")
###
class Eagle(BirdFamily):
    pass
class Ostrich(BirdFamily):
    def fly(self):
        print(self.name, "cannot fly.")
```

در این مثال، کلاس BirdFamily کلاس پایه است که یک متد و روش به نام fly دارد. کلاس‌های Eagle و Ostrich از کلاس BirdFamily ارث‌بری می‌کنند و به روش fly دسترسی دارند. در این مثال کلس Eagle کلس است که کاملا و بدون هیچ تغییر اضافی، از کلس والد خود ارث‌بری میکند و در واقع هیچ چیز جدیدی برای ارائه دادن ندارد! و کلاس Ostrich نیز روش fly را به صورت متفاوتی پیاده‌سازی می‌کند.

بنابراین اگر یک شیء از کلاس عقاب (Eagle) یا شترمرغ (Ostrich) بسازیم، می‌توانیم از روش پرواز (fly) استفاده کنیم:

```
eagle1 = Eagle("Golden")

ostrich1 = Ostrich("Pencil Neck")

###

eagle1.fly()    # Output: Golden can fly

ostrich1.fly()  # Output: Pencil Neck cannot fly
```

این مثال نشان می‌دهد که چگونه از وراثت در برنامه‌نویسی شیءگرا بهره می بریم و کد را تمیزتر و قابل مدیریت‌تر نگه داریم. همچنین وراثت امکان استفاده از مفهوم چند ریختی (Polymorphism) را در برنامه‌نویسی شیءگرا فراهم می‌آورد که به معنای دارا بودن شکل‌های متعدد است. در مثال بالا، کلاس شترمرغ متد و روش fly را به شکل خودش پیاده‌سازی کرد، یعنی با این حال که فرض شد تمام خانواده ی پرندگان قابلیت پرواز را داراست اما متد پرواز برای شتر مرغ درون آن هم تعریف شده است تا مختص خودش باشد (و متدش overridden می شود) که این یک نمونه از چندریختی یا Polymorphism است.

پس به طور کلی: در پایتون، ارث‌بری یک مکانیزم است که به یک کلاس اجازه می‌دهد ویژگی‌ها و متدهای یک کلاس دیگر را به ارث ببرد. کلاسی که ویژگی‌ها و متدها از آن به ارث برده می‌شود، به عنوان کلاس والد یا سوپرکلاس شناخته می‌شود، در حالی که کلاسی که ارث می‌برد، به عنوان کلاس فرزند یا ساب‌کلاس شناخته می‌شود.

با استفاده از ارث‌بری، ما می‌توانیم سلسله‌مراتبی از کلاس‌ها ایجاد کنیم که هر زیرکلاس قابلیت ارث‌بری و گسترش تابعیت‌های کلاس والد خود را دارد. این کار به افزایش قابلیت استفاده مجدد کد کمک می‌کند و در سازماندهی و ساختاردهی کد موثر است.

برای ایجاد یک زیرکلاس، آن را با مشخص کردن نام کلاس والد درون پرانتزها بعد از نام زیرکلاس تعریف می‌کنیم. سپس زیرکلاس می‌تواند به تمام ویژگی‌ها و متدهای کلاس والد دسترسی داشته باشد. علاوه بر این، ویژگی‌ها و متدهای به ارث برده شده را میتوان در زیرکلاس تغییر یا گسترش داد.

تجرید (Abstraction)

»کلاس مجرد» کلاسی است که شامل یک یا چند «متد مجرد» باشد و «متد مجرد» متدی است که اعلان (Declare) شده ولی بدنه آن تعریف (Define) نشده است. کلاس‌های مجرد قابلیت نمونه‌سازی ندارند و نمی‌توان از آن‌ها شی ایجاد نمود؛ چرا که هدف از توسعه آن‌ها قرار گرفتن در بالاترین سطح (یا چند سطح بالایی) درخت وراثت، به عنوان کلاس پایه برای ارث‌بری کلاس‌های پایین‌تر می‌باشد. ایده طراحی کلاس مجرد در تعیین یک نقشه توسعه برای کلاس‌های فرزند آن است؛ تعیین صفات و متدهای لازم ولی واگذاردن تعریف متدها بر عهده کلاس‌های فرزند.

به عنوان نمونه سه کلاس «ماهی»، «گربه» و «کبوتر» را در نظر بگیرید. این کلاس‌ها جدا از رفتارهای خاص خود (مانند: «پرواز کردن» در کبوتر یا «شنا کردن» در ماهی)، در یک سری رفتار به مانند «نفس کشیدن»، «غذا خوردن» و... مشترک هستند. راه درستِ توسعه این کلاس‌ها تعیین یک «کلاس پایه» برای رفتارهای مشترک و ارث‌بری هر سه آن‌ها می‌باشد. ولی از آنجا که هر یک، این رفتارهای مشترک را به گونه‌ای دیگر انجام می‌دهد؛ راه درست‌تر آن است که یک «کلاس مجرد» به عنوان «کلاس پایه» آن‌ها در نظر بگیریم؛ در این حالت هر کدام از کلاس‌ها ضمن دانستن رفتارهای لازم می‌تواند آن‌ها را متناسب با خواست خود تعریف نمایند.

(source: https://python.coderz.ir/lessons/l05-object-oriented-programming.html)

مانند (مثالی دیگر در قالب کد):

```
from abc import ABC, abstractmethod

class Animal(ABC):
    def init(self, name):
        self.name = name
# متدی که در کلاس مجرد هست، به شکل زیر است:
    @abstractmethod
    def make_sound(self):
        pass
# متدی که در کلاس مجرد هست، به شکل زیر است:
    @abstractmethod
    def display_info(self):
        print(f"I am {self.name}.")
        self.make_sound()
###
class Dog(Animal):
    def make_sound(self):
        print("Woof!")
```

تابع مورد نظر (make_sound) در کلاس فرزند هست و دستورات در این تابع هستند ولی در کلاس والد فقط تعریف نام (Declare) شده است.

پس به طور کلی: انتزاع (Abstraction) یک مفهوم بنیادی در برنامه‌نویسی شیءگرا است که میتواند به معنای انتزاع/چکیده/خلاصه معنی شود. و این اجازه را می‌دهد تا بر روی ویژگی‌های ضروری یک شیء تمرکز کنیم، در حالی که جزئیات غیر ضروری را پنهان کنیم. درواقع این یک روش برای نمایش سیستم‌های پیچیده یا ایده‌های پیچیده به صورت ساده و مدیریت‌پذیر است.

در برنامه‌نویسی، انتزاع به ما امکان می‌دهد تا رفتار، خصوصیات و قابلیت‌های یک شیء را در ابتدا تعریف کنیم، بدون بیان جزئیات پیاده‌سازی داخلی آنها.

و همینطور با انتزاع جزئیات غیر ضروری پنهان می‌شوند، کد، قابلیت سازماندهی و ساختاردهی را پیدا می‌کند.
