📌 ترکیب (Composition)

ترکیب (Composition) یکی از اصول طراحی در شی‌گرایی است که به جای استفاده از وراثت (Inheritance) برای ایجاد روابط بین کلاس‌ها، به ما این امکان را می‌دهد که کلاس‌ها را با استفاده از **اجزای داخلی** به هم متصل کنیم. به عبارت ساده، در ترکیب، یک کلاس می‌تواند به دیگر کلاس‌ها به عنوان **اجزاء داخلی** (subcomponents) وابسته باشد و از آن‌ها برای انجام وظایف خود استفاده کند.

ترکیب به طور خاص زمانی مفید است که:

- نیازی به ایجاد سلسله‌مراتب پیچیده از کلاس‌ها نباشد.
- کلاس‌ها به گونه‌ای با یکدیگر تعامل دارند که به جای ارث‌بری بهتر است که از ارتباطات ساده‌تری استفاده شود.
- ایجاد وابستگی‌های پیچیده بین کلاس‌ها که در وراثت ایجاد می‌شود، مشکل‌ساز باشد.

✅ تفاوت بین وراثت و ترکیب

1. **وراثت (Inheritance):**
   - در وراثت، یک کلاس (کلاس فرزند) ویژگی‌ها و متدهای یک کلاس دیگر (کلاس پایه) را به ارث می‌برد.
   - وراثت معمولاً زمانی استفاده می‌شود که بین کلاس‌ها یک **رابطه IS-A** وجود دارد، یعنی کلاس فرزند نوعی از کلاس پایه است.
2. **ترکیب (Composition):**
   - در ترکیب، یک کلاس (کلاس اصلی) از کلاس‌های دیگر به عنوان **اجزاء داخلی** استفاده می‌کند.
   - ترکیب معمولاً زمانی استفاده می‌شود که بین کلاس‌ها یک **رابطه HAS-A** وجود دارد، یعنی یک کلاس دارای اجزای دیگری است.

به عنوان مثال، یک **ماشین** ممکن است دارای **چرخ** باشد، اما چرخ‌ها نوعی از ماشین نیستند؛ بلکه جزء آن هستند.

✅ نحوه استفاده از ترکیب در پایتون

در ترکیب، به جای اینکه یک کلاس از کلاس دیگری ارث‌بری کند، از اشیاء کلاس‌های دیگر به عنوان ویژگی‌های خود استفاده می‌کند. این به این معنی است که یک کلاس می‌تواند درون خود از دیگر کلاس‌ها به عنوان اجزاء استفاده کند.

مثال ساده از ترکیب:

فرض کنید می‌خواهیم یک کلاس `Car` بسازیم که از چندین جزء مانند `Engine` و `Wheel` استفاده کند. در اینجا، ماشین `Car` به اجزای داخلی خود مانند **موتور** و **چرخ‌ها** نیاز دارد، اما این اجزاء به طور مستقل از کلاس ماشین وجود دارند.

```
class Engine:
    def start(self):
        print("Engine is starting")

class Wheel:
    def rotate(self):
        print("Wheel is rotating")
```

```
# ترکیب کلاس‌ها برای ایجاد کلاس ماشین
class Car:
    def __init__(self):
        self.engine = Engine()  # موتور ماشین
        self.wheels = [Wheel() for _ in range(4)]  # چهار چرخ برای ماشین

    def drive(self):
        self.engine.start()  # شروع موتور
        for wheel in self.wheels:
            wheel.rotate()  # چرخ‌ها شروع به چرخش می‌کنند

# ایجاد شیء از کلاس Car
my_car = Car()
my_car.drive()
```

خروجی:

```
Engine is starting
Wheel is rotating
Wheel is rotating
Wheel is rotating
Wheel is rotating
```

✅ مزایای ترکیب:

1. **انعطاف‌پذیری بیشتر:** ترکیب به شما اجازه می‌دهد که اجزای مختلف را به راحتی جایگزین یا تغییر دهید. این برخلاف وراثت است که گاهی اوقات تغییرات در یک کلاس پایه می‌تواند اثرات زیادی در کلاس‌های فرزند داشته باشد.
2. **کاهش پیچیدگی‌های وارثت:** ترکیب به کاهش سلسله‌مراتب پیچیده کمک می‌کند. در سیستم‌های پیچیده، تعداد زیادی کلاس پایه ممکن است منجر به مشکلاتی در کدنویسی و نگهداری کد شود. ترکیب می‌تواند ساده‌تر باشد.
3. **قابلیت استفاده مجدد بیشتر:** با ترکیب، می‌توانید از کلاس‌های مختلف به راحتی در سایر بخش‌های پروژه استفاده کنید، بدون اینکه مجبور باشید تغییرات گسترده‌ای در ساختار وراثت ایجاد کنید.
4. **کاهش وابستگی‌ها:** در ترکیب، یک کلاس به جای وابستگی به سلسله‌مراتب کلاس‌ها، به اجزاء دیگر وابسته است که باعث کاهش وابستگی‌های پیچیده در کد می‌شود.

✅ مثال‌های بیشتر از ترکیب در پایتون:

1. سیستم مدرسه (School System)

در این مثال، یک **مدرسه** می‌تواند از اجزای مختلفی مانند **کلاس‌ها**، **دانش‌آموزان** و **معلمان** استفاده کند. به جای استفاده از وراثت، می‌توانیم این اجزاء را ترکیب کنیم.

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

    def study(self):
```

```
        print(f"{self.name} is studying.")

class Teacher:
    def __init__(self, name):
        self.name = name

    def teach(self):
        print(f"{self.name} is teaching.")

class School:
    def __init__(self, name):
        self.name = name
        self.students = []
        self.teachers = []

    def add_student(self, student):
        self.students.append(student)

    def add_teacher(self, teacher):
        self.teachers.append(teacher)

    def start_classes(self):
        for teacher in self.teachers:
            teacher.teach()
        for student in self.students:
            student.study()

# استفاده از ترکیب برای ایجاد سیستم مدرسه
school = School("Sunshine High School")
student1 = Student("Alice")
student2 = Student("Bob")
teacher1 = Teacher("Mr. Smith")

school.add_student(student1)
school.add_student(student2)
school.add_teacher(teacher1)

school.start_classes()
```

خروجی:

```
Mr. Smith is teaching.
Alice is studying.
Bob is studying.
```

✅ نتیجه‌گیری:

ترکیب یک تکنیک قدرتمند برای ایجاد روابط بین کلاس‌ها است که بر اساس اجزاء داخلی بنا شده است. به جای استفاده از وراثت، که می‌تواند پیچیدگی‌های بیشتری ایجاد کند، ترکیب به ما این امکان را می‌دهد که کلاس‌ها را از اجزاء مختلف بسازیم و از ویژگی‌های آن‌ها به صورت جداگانه استفاده کنیم. این باعث انعطاف‌پذیری و سادگی بیشتر در طراحی سیستم‌ها می‌شود.