📌 مبانی شی‌گرایی (Object-Oriented Programming - OOP) در پایتون

شی‌گرایی (OOP - Object Oriented Programming) یک **الگوی برنامه‌نویسی** است که بر پایه **ایجاد اشیاء (Objects) از کلاس‌ها (Classes)** بنا شده است. در این بخش، مفاهیم پایه‌ای **شیء، کلاس و تفاوت آنها** را بررسی کرده و چندین **مثال عملی** ارائه می‌کنیم.

✅ ۱. مفهوم شیء (Object) و کلاس (Class)

🔷 شیء (Object) چیست؟

شیء (Object) یک **نمونه (Instance) از یک کلاس** است که ویژگی‌ها و رفتارهای خاصی دارد.
مثلاً، در دنیای واقعی، یک **ماشین** یک شیء است که **ویژگی‌هایی** مانند رنگ، مدل و سرعت دارد و می‌تواند **رفتارهایی** مانند حرکت یا ترمز کردن داشته باشد.

🔷 کلاس (Class) چیست؟

کلاس (Class) یک **قالب (Blueprint) یا الگو** برای ساخت اشیاء است.
به عبارت دیگر، کلاس **ساختاری را تعریف می‌کند که از روی آن می‌توان اشیاء ایجاد کرد.**

🎯 تفاوت کلاس و شیء

کلاس (Class)	شیء (Object)
یک **الگو** یا **قالب** برای ایجاد اشیاء است.	یک **نمونه (Instance)** از کلاس است.
فقط **تعریف ویژگی‌ها و رفتارها** را مشخص می‌کند.	دارای **مقدار واقعی برای ویژگی‌ها** و **اجرای رفتارها** است.
مثال: «کلاس ماشین» فقط مشخص می‌کند که ماشین **چه ویژگی‌هایی** دارد.	مثال: «یک ماشین خاص» مانند **پراید قرمز مدل ۱۴۰۰** یک شیء است.

✅ ۲. تعریف کلاس و ایجاد شیء در پایتون

در پایتون، برای تعریف یک کلاس از کلمه کلیدی `class` استفاده می‌کنیم.

📌 مثال ۱: تعریف یک کلاس و ایجاد شیء

```
# تعریف یک کلاس به نام Car
class Car:
    def __init__(self, brand, color, speed):
        self.brand = brand  # ویژگی برند خودرو
        self.color = color  # ویژگی رنگ خودرو
        self.speed = speed  # ویژگی سرعت خودرو

    def drive(self):
        print(f"{self.brand} با سرعت {self.speed} کیلومتر بر ساعت در حال حرکت است!")

# ایجاد یک شیء از کلاس Car
car1 = Car("BMW", "220 ,"مشکی)
```

```
car2 = Car("Toyota", "180 ,"سفید)

# استفاده از متد drive
car1.drive()
car2.drive()
```

✅ 🔹 **خروجی:**

```
با سرعت 220 کیلومتر بر ساعت در حال حرکت است BMW!
با سرعت 180 کیلومتر بر ساعت در حال حرکت است Toyota!
```

✅ 🔹 **نکات مهم در این مثال:**

1. **متد** `__init__` : تابع **سازنده (Constructor)** است که هنگام ایجاد شیء **ویژگی‌های آن را مقداردهی** می‌کند.
2. `self` : به شیء جاری اشاره دارد و به کمک آن می‌توان ویژگی‌های شیء را مقداردهی کرد.
3. **ایجاد اشیاء (** `car1` , `car2` **)**: هر شیء مقادیر خاص خود را دارد.

✅ ۳. استفاده از متدها (Functions in Class)

متدها همان **توابعی** هستند که در داخل کلاس تعریف می‌شوند و روی **اشیاء** کار می‌کنند.

📌 مثال ۲: اضافه کردن متد تغییر سرعت

```
class Car:
    def __init__(self, brand, color, speed):
        self.brand = brand
        self.color = color
        self.speed = speed

    def drive(self):
        print(f"{self.brand} با سرعت {self.speed} کیلومتر بر ساعت در حال حرکت است!")

    def change_speed(self, new_speed):
        self.speed = new_speed
        print(f"سرعت جدید {self.brand}: {self.speed} کیلومتر بر ساعت")

# ایجاد شیء و استفاده از متدها
car1 = Car("BMW", "220 ,"مشکی)
car1.drive()
car1.change_speed(250)  # تغییر سرعت
```

✅ 🔹 **خروجی:**

```
با سرعت 220 کیلومتر بر ساعت در حال حرکت است BMW!
سرعت جدید BMW: 250 کیلومتر بر ساعت
```

✅ 🔹 **نکات مهم:**

- متد `change_speed` مقدار **سرعت خودرو** را تغییر می‌دهد.
- هر متد برای دسترسی به ویژگی‌های شیء از `self` استفاده می‌کند.

✅ ۴. دسترسی به ویژگی‌های شیء (Attributes)

در پایتون، می‌توان **ویژگی‌های یک شیء را مستقیماً تغییر داد**، اما بهتر است از **متدهای کلاس** برای این کار استفاده کنیم.

📌 مثال ۳: تغییر مستقیم ویژگی‌های شیء

```
car1.speed = 300  # تغییر مقدار ویژگی شیء
print(car1.speed)  # خروجی: 300
```

✅ 🔹 نکته:

- در پایتون، می‌توان ویژگی‌های شیء را **مستقیماً تغییر داد**، اما در زبان‌هایی مانند جاوا و سی‌پلاس‌پلاس، باید این ویژگی‌ها را **خصوصی** (`private`) تعریف کرد و از **متدهای خاصی (Getter و Setter)** برای تغییر آنها استفاده کرد.

✅ ۵. چندین شیء از یک کلاس (Multiple Objects)

می‌توان **چندین شیء از یک کلاس** ایجاد کرد که هر کدام مقدارهای مخصوص به خود را دارند.

📌 مثال ۴: ایجاد چندین شیء از کلاس Car

```
car1 = Car("BMW", "220 ,"مشکی")
car2 = Car("Toyota", "180 ,"سفید")
car3 = Car("Mercedes", "240 ,"قرمز")

car1.drive()
car2.drive()
car3.drive()
```

✅ 🔹 خروجی:

```
BMW با سرعت 220 کیلومتر بر ساعت در حال حرکت است!
Toyota با سرعت 180 کیلومتر بر ساعت در حال حرکت است!
Mercedes با سرعت 240 کیلومتر بر ساعت در حال حرکت است!
```

✅ ۶. استفاده از مقدار پیش‌فرض در کلاس‌ها

می‌توان **مقدار پیش‌فرض** برای ویژگی‌های کلاس تعیین کرد.

📌 مثال ۵: مقدار پیش‌فرض برای رنگ خودرو

```
class Car:
    def __init__(self, brand, speed, color="مشکی"):  # مقدار پیش‌فرض برای رنگ
        self.brand = brand
        self.color = color
        self.speed = speed

car1 = Car("BMW", 220)  # رنگ مشخص نشده -> مشکی
car2 = Car("Toyota", 180, "سفید")

print(f"{car1.brand} رنگ: {car1.color}")
print(f"{car2.brand} رنگ: {car2.color}")
```

✅ 🔹 خروجی:

```
BMW رنگ: مشکی
Toyota رنگ: سفید
```

🎯 جمع‌بندی نهایی

✅ **شیء (Object)** نمونه‌ای از یک کلاس است.
✅ **کلاس (Class)** قالبی برای ایجاد اشیاء است.
✅ ویژگی‌ها (**Attributes**) مشخصات یک شیء هستند.
✅ متدها (**Methods**) رفتارهای یک شیء را تعریف می‌کنند.
✅ از `self` برای دسترسی به ویژگی‌های شیء استفاده می‌شود.
✅ می‌توان **چندین شیء از یک کلاس** ایجاد کرد.
✅ مقدار پیش‌فرض برای ویژگی‌های کلاس قابل تنظیم است.

📌 تمرین برای شما:

یک کلاس **Student** تعریف کنید که ویژگی‌های **نام، سن و معدل** را داشته باشد. متدی برای **افزایش معدل** و متدی برای **نمایش اطلاعات دانشجو** اضافه کنید. 🚀