مدیریت فایل‌ها با استفاده از Context Manager (`with`)

هنگام کار با فایل‌ها، مهم است که پس از اتمام کار، فایل را ببندیم. بستن فایل باعث جلوگیری از مشکلاتی مانند **نشت حافظه، استفاده بیش از حد از منابع سیستم، و از دست رفتن داده‌ها** می‌شود.

اگرچه می‌توان از `open()` و `close()` برای کار با فایل‌ها استفاده کرد، اما روش بهتر استفاده از **Context Manager** (`with`) است که به‌طور خودکار فایل را پس از اتمام کار می‌بندد.

---

1. مزایای استفاده از `with` برای مدیریت فایل‌ها

✅ **بستن خودکار فایل:** نیازی به `file.close()` نیست.
✅ **کاهش خطاها:** اگر استثنایی رخ دهد، فایل همچنان بسته خواهد شد.
✅ **کدنویسی خواناتر و تمیزتر:** ساختار `with` باعث کاهش پیچیدگی کد می‌شود.

---

2. مقایسه `with` و `open()`

روش سنتی (بدون `with`)

```
file = open("example.txt", "r")  # باز کردن فایل
try:
    content = file.read()  # خواندن فایل
    print(content)
finally:
    file.close()  # بستن فایل
```

🔹 **مشکلات:**

- باید مطمئن شویم که `file.close()` حتماً اجرا شده است، حتی اگر خطایی رخ دهد.
- نیاز به `try-finally` برای مدیریت بستن فایل.

روش بهتر با `with`

```
with open("example.txt", "r") as file:
    content = file.read()  # خواندن فایل
    print(content)
```

🔹 **مزایا:**

- **نیازی به `file.close()` نداریم**؛ `with` این کار را خودکار انجام می‌دهد.
- اگر خطایی رخ دهد، فایل باز نخواهد ماند.
- **کد ساده‌تر و خواناتر است.**

---

3. مثال‌هایی از استفاده از `with` برای مدیریت فایل‌ها

مثال 1: خواندن از فایل

```
with open("data.txt", "r") as file:
    for line in file:
        print(line.strip())  # خواندن هر خط و حذف فاصله‌های اضافی
```

مثال 2: نوشتن در فایل

```
with open("output.txt", "w") as file:
    file.write("این یک تست برای نوشتن در فایل است.\n")
```

🔹 **نکته:** اگر فایل `output.txt` وجود نداشته باشد، ایجاد می‌شود. اگر وجود داشته باشد، محتوای آن پاک شده و جایگزین می‌شود.

مثال 3: اضافه کردن به فایل (append)

```
with open("log.txt", "a") as file:
    file.write("ورودی جدید به فایل اضافه شد.\n")
```

🔹 **نکته:** حالت `"a"` محتوا را **حذف نمی‌کند** و اطلاعات جدید را **به انتهای فایل** اضافه می‌کند.

مثال 4: خواندن و نوشتن هم‌زمان

```
with open("source.txt", "r") as source, open("destination.txt", "w") as destination:
    for line in source:
        destination.write(line)  # کپی کردن محتوا از یک فایل به فایل دیگر
```

🔹 **نکته:** می‌توان چندین فایل را هم‌زمان با `with` باز کرد.

مثال 5: استفاده از `with` برای مدیریت خطاها

```
try:
    with open("config.txt", "r") as file:
        data = file.read()
        print(data)
except FileNotFoundError:
    print("خطا: فایل موردنظر یافت نشد!")
except PermissionError:
    print("خطا: دسترسی به فایل امکان‌پذیر نیست!")
```

🔹 **مزایا:**

- مدیریت خطاها هنگام باز کردن فایل.

- جلوگیری از کرش شدن برنامه در صورت نبود فایل.

---

4. نتیجه‌گیری

✅ **استفاده از `with` یک روش استاندارد و بهینه برای کار با فایل‌ها است.**
✅ **`with` به‌طور خودکار فایل را پس از استفاده می‌بندد، حتی در صورت وقوع خطا.**
✅ **کدهای نوشته‌شده با `with` خواناتر، ایمن‌تر و ساده‌تر از روش سنتی هستند.**

🎯 تمرین:

یک برنامه بنویسید که یک فایل متنی را باز کند، تعداد خطوط آن را بشمارد و تعداد کل کلمات را نمایش دهد. از `with` برای مدیریت فایل استفاده کنید.