خواندن فایل‌ها در پایتون

در پایتون، برای خواندن فایل‌ها از **تابع** `open()` همراه با حالت `"r"` (خواندن) استفاده می‌شود. چندین روش برای خواندن محتوای فایل وجود دارد که در ادامه به بررسی آن‌ها همراه با مثال می‌پردازیم.

---

۱. `read()` : خواندن کل محتوای فایل

تابع `read()` کل فایل را به‌عنوان یک رشته (String) برمی‌گرداند.

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

✅ **نکات:**

- **اگر فایل بزرگ باشد، `read()` ممکن است حافظه زیادی مصرف کند.**
- می‌توان تعداد بایت مشخصی را خواند:

```
with open("example.txt", "r") as file:
    partial_content = file.read(10)  # فقط 10 کاراکتر اول را بخواند
    print(partial_content)
```

---

۲. `readline()` : خواندن یک خط از فایل

تابع `readline()` فقط **یک خط** از فایل را برمی‌گرداند.

```
with open("example.txt", "r") as file:
    line1 = file.readline()  # خواندن خط اول
    line2 = file.readline()  # خواندن خط دوم
    print(line1.strip())  # حذف کاراکتر '\n' در انتهای خط
    print(line2.strip())
```

✅ **نکات:**

- هر بار که `readline()` فراخوانی می‌شود، خط بعدی خوانده می‌شود.
- اگر فایل دارای **خطوط زیادی** باشد، این روش **بهتر از** `read()` است چون نیازی به خواندن کل فایل نیست.

---

۳. `readlines()` : خواندن تمامی خطوط به‌صورت لیست

تابع `readlines()` تمام خطوط فایل را به‌صورت **لیستی از رشته‌ها** برمی‌گرداند.

```
with open("example.txt", "r") as file:
    lines = file.readlines()  # خواندن همه خطوط و ذخیره در یک لیست
    print(lines)
```

📌 **مثال: خواندن هر خط و پردازش جداگانه**

```python
with open("example.txt", "r") as file:
    for line in file.readlines():
        print(line.strip())  # حذف \n در انتهای هر خط
```

---

۴. خواندن فایل خط به خط با حلقه `for`

✅ بهترین روش برای پردازش **فایل‌های بزرگ** بدون مصرف زیاد حافظه:

```python
with open("example.txt", "r") as file:
    for line in file:
        print(line.strip())  # خواندن و پردازش هر خط
```

💡 **مزیت:** نیازی به ذخیره کل محتوای فایل در حافظه ندارد، بنابراین **برای فایل‌های حجیم توصیه می‌شود.**

---

۵. ترکیب خواندن داده‌ها و پردازش آن‌ها

📌 **مثال: خواندن یک فایل CSV و تبدیل داده‌ها به لیست**

```python
with open("data.csv", "r") as file:
    for line in file:
        fields = line.strip().split(",")  # تبدیل خط به لیست مقادیر
        print(fields)
```

📌 **مثال: خواندن فایل و شمارش تعداد کلمات در آن**

```python
with open("example.txt", "r") as file:
    content = file.read()
    words = content.split()
    print(f"تعداد کلمات: {len(words)}")
```

---

۶. خواندن فایل‌های حجیم بدون مصرف زیاد حافظه

📌 **اگر فایل شما بسیار بزرگ باشد، بهترین روش استفاده از `()read` با اندازه مشخص یا `for` است.**

```python
with open("large_file.txt", "r") as file:
    while chunk := file.read(1024):  # خواندن 1024 بایت در هر مرحله
        print(chunk)
```

جمع‌بندی

- 🔹 `()read` → کل فایل را می‌خواند.
- 🔹 `()readline` → فقط یک خط می‌خواند.
- 🔹 `()readlines` → همه خطوط را در یک لیست ذخیره می‌کند.
- 🔹 **بهترین روش برای فایل‌های حجیم**: استفاده از `for` یا `()read` با اندازه مشخص.

🔥 **مدیریت صحیح فایل‌ها به بهینه‌سازی عملکرد و جلوگیری از مصرف بیهوده حافظه کمک می‌کند!** 🚀