استثناهای رایج در کار با فایل‌ها و نحوه مدیریت آن‌ها

هنگام کار با فایل‌ها، ممکن است با خطاهای مختلفی روبه‌رو شویم. اگر این خطاها مدیریت نشوند، برنامه کرش می‌کند. پایتون خطاهای مخصوصی برای مشکلات فایل‌ها دارد که باید به‌درستی کنترل شوند.

---

1. استثناهای رایج در کار با فایل‌ها

🔷 `FileNotFoundError`

وقتی فایلی که می‌خواهید باز کنید وجود نداشته باشد، این خطا رخ می‌دهد.

🔷 مثال:

```
try:
    with open("non_existent_file.txt", "r") as file:
        content = file.read()
except FileNotFoundError:
    print("خطا: فایل موردنظر یافت نشد!")
```

✅ **راه‌حل:** قبل از باز کردن فایل، بررسی کنید که آیا فایل وجود دارد یا نه.

🔷 `PermissionError`

اگر مجوزهای لازم برای دسترسی به فایل نداشته باشید، این خطا رخ می‌دهد.

🔷 مثال:

```
try:
    with open("/root/protected_file.txt", "w") as file:
        file.write("داده‌های جدید")
except PermissionError:
    print("خطا: شما مجوز لازم برای تغییر این فایل را ندارید!")
```

✅ **راه‌حل:** بررسی کنید که آیا دسترسی نوشتن/خواندن دارید. اگر نیاز به مجوز دارید، آن را تنظیم کنید.

🔷 `IsADirectoryError`

اگر به‌جای فایل، یک **دایرکتوری** را باز کنید، این خطا رخ می‌دهد.

🔷 مثال:

```
try:
    with open("some_folder/", "r") as file:
        content = file.read()
except IsADirectoryError:
    print("خطا: مسیر مشخص شده یک پوشه است، نه یک فایل!")
```

✅ **راه‌حل:** بررسی کنید که مسیر یک **فایل متنی** باشد، نه یک دایرکتوری.

🔷 `IOError`

خطای عمومی ورودی/خروجی (I/O) است که شامل مشکلاتی مانند خطای دیسک، قطع شدن ارتباط شبکه هنگام خواندن فایل از سرور، و... می‌شود.

🔷 مثال:

```
try:
    with open("file.txt", "r") as file:
        content = file.read()
except IOError:
    print("خطا: مشکلی در خواندن فایل رخ داده است!")
```

✅ **راه‌حل:** هنگام کار با فایل‌های خارجی، وضعیت ارتباط را بررسی کنید.

🔷 `UnicodeDecodeError`

اگر یک فایل **غیرفارسی** را با انکدینگ اشتباه (مثلاً UTF-8) باز کنید، این خطا رخ می‌دهد.

🔷 مثال:

```
try:
    with open("data.txt", "r", encoding="utf-8") as file:
        content = file.read()
except UnicodeDecodeError:
    print("خطا: انکدینگ فایل نامعتبر است! لطفاً انکدینگ صحیح را مشخص کنید.")
```

✅ راه‌حل:

1. مطمئن شوید که فایل از انکدینگ **درست** استفاده می‌کند.
2. اگر نمی‌دانید فایل چه انکدینگی دارد، از `"encoding="utf-8` یا `"encoding="latin-1` تست کنید.

---

2. مدیریت چندین نوع استثنا با `except`

اگر بخواهیم چندین نوع خطا را مدیریت کنیم، می‌توانیم چندین `except` داشته باشیم.

🔷 مثال:

```
try:
    with open("data.txt", "r") as file:
        content = file.read()
except FileNotFoundError:
    print("خطا: فایل پیدا نشد!")
except PermissionError:
    print("خطا: شما مجوز لازم برای این فایل را ندارید!")
except UnicodeDecodeError:
    print("خطا: مشکل در انکدینگ فایل!")
except Exception as e:
    print(f"یک خطای ناشناخته رخ داد: {e}")
```

✅ **نکته:**

- `Exception as e` خطاهای غیرمنتظره را مدیریت می‌کند.
- **همیشه خطاهای خاص را قبل از `Exception` قرار دهید!** چون `Exception` همه‌ی خطاها را می‌گیرد و دیگر `except` های بعدی اجرا نمی‌شوند.

---

3. ترکیب `try-except` با `finally` برای اطمینان از بسته شدن فایل

حتی اگر خطایی رخ دهد، باید فایل بسته شود. می‌توان از `finally` برای این کار استفاده کرد.

🔷 **مثال:**

```
try:
    file = open("data.txt", "r")
    content = file.read()
except FileNotFoundError:
    print("خطا: فایل پیدا نشد!")
finally:
    file.close()  # بسته شدن فایل حتی در صورت وقوع خطا
```

✅ **اما بهتر است `with` استفاده شود، زیرا `finally` را خودکار مدیریت می‌کند!**

---

4. استفاده از `os` برای بررسی وجود فایل قبل از باز کردن

پیش از باز کردن فایل، می‌توان با استفاده از `os.path.exists()` بررسی کرد که آیا فایل وجود دارد یا نه.

🔷 **مثال:**

```
import os

file_path = "data.txt"

if os.path.exists(file_path):
    with open(file_path, "r") as file:
        content = file.read()
        print(content)
else:
    print("خطا: فایل موردنظر وجود ندارد!")
```

✅ **مزیت:**

- از بروز `FileNotFoundError` جلوگیری می‌شود.

---

5. خلاصه و نتیجه‌گیری

✅ **مدیریت خطاها مهم است، زیرا برنامه را از کرش شدن حفظ می‌کند.**
✅ **`try-except` به ما کمک می‌کند که استثناهای رایج مانند `PermissionError`، `FileNotFoundError` و `UnicodeDecodeError` را مدیریت کنیم.**
✅ **استفاده از `with` و `os.path.exists()` روش بهتری برای جلوگیری از خطاهای رایج است.**

🎯 تمرین:

یک برنامه بنویسید که نام یک فایل را از کاربر بگیرد و محتوای آن را نمایش دهد. برنامه باید:

1. بررسی کند که فایل وجود دارد.
2. اگر وجود نداشت، پیام مناسب نشان دهد.
3. اگر دسترسی به فایل نداشت، پیام مناسب نشان دهد.
4. انکدینگ فایل را بررسی کند تا در صورت مشکل، خطای مناسبی نمایش دهد.