File Handling

Day 15 - Python Basics

Shaida Muhammad

ShaidaSherpao@gmail.com

Friday, March 21, 2025

Agenda

1 What & Why file handling?

2 Types of files (text vs. binary)

3 Opening and closing files

4 Reading from files

5 Writing to files

6 Deleting files

7 Working with CSV files

8 Working with JSON files

9 Error handling in file operations

10 Hands-on practice

What & Why File Handling?

- **Definition:** Creating, reading, updating, and deleting files on a storage system using program.
- **Real-World Analogy:**
  - Think of files as digital notebooks.
  - Just like you write and read from a notebook, programs read from and write to files.
- **Why Learn File Handling?**
  - **Persistence:** Save data even after the program ends.
  - **Data Sharing:** Exchange data between programs.
  - **Automation:** Process large datasets or logs efficiently.

Types of Files

- **Text Files:**
  - Human-readable (e.g., `.txt`, `.csv`, `.json`).
  - Store plain text or structured data.
- **Binary Files:**
  - Non-human-readable (e.g., `.jpg`, `.exe`, `.dat`).
  - Store data in binary format (e.g., images, executables).

Opening and Closing Files

- **Opening a File:**
  - Use the `open()` function.
  - Syntax:

```
file = open("filename.txt", mode)
```

  - Modes:
    - `"r"`: Read (default).
    - `"w"`: Write (overwrites existing content).
    - `"a"`: Append (adds to existing content).
    - `"b"`: Binary mode (e.g., `"rb"`, `"wb"`).
- **Closing a File:**
  - Use the `close()` method.
  - Always close files to free up resources.
  - Example:

```
file = open("example.txt", "r")
# Perform file operations
file.close()
```

Reading from Files

- **Read Entire File:**

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

- **Read Line by Line:**

```
with open("example.txt", "r") as file:
    for line in file:
        print(line.strip())  # strip() removes newline characters
```

- **Read Specific Lines:**

```
with open("example.txt", "r") as file:
    lines = file.readlines()  # Returns a list of lines
    print(lines[0])  # Print the first line
```

Writing to Files

- **Write to a File:**

```
with open("example.txt", "w") as file:
    file.write("Starry Mashy!\n")
    file.write("Pa Khair Raaghly!\n")
```

- **Append to a File:**

```
with open("example.txt", "a") as file:
    file.write("This is a new line.\n")
```

- **Key Points:**
  - "w" mode overwrites the file.
  - "a" mode appends to the file.

Deleting Files

- **Using `os.remove()`:**
  - Deletes a file.
  - Example:

```
import os
os.remove("example.txt")
```

- **Using `os.unlink()`:**
  - Same as `os.remove()`.
  - Example:

```
import os
os.unlink("example.txt")
```

- **Handling Errors:**
  - Check if the file exists before deleting.
  - Example:

```
import os
if os.path.exists("example.txt"):
    os.remove("example.txt")
else:
    print("File does not exist.")
```

Working with CSV Files

- **What is a CSV File?**
  - Comma-Separated Values (e.g., `data.csv`).
  - Used for storing tabular data.
- **Reading CSV Files:**

```
import csv

with open("data.csv", "r") as file:
    reader = csv.reader(file)
    for row in reader:
        print(row)  # Each row is a list
```

- **Writing to CSV Files:**

```
import csv

with open("data.csv", "w", newline="") as file:
    writer = csv.writer(file)
    writer.writerow(["Name", "Age"])
    writer.writerow(["Ali", 17])
```

Working with JSON Files

- **What is JSON?**
  - JavaScript Object Notation (e.g., data.json).
  - Used for storing structured data (key-value pairs).
- **Reading JSON Files:**

```
import json

with open("data.json", "r") as file:
    data = json.load(file)
    print(data)
```

- **Writing to JSON Files:**

```
import json

data = {
    "name": "Ali",
    "age": 17,
    "city": "Islamabad"
}

with open("data.json", "w") as file:
    json.dump(data, file, indent=4)
```

Error Handling in File Operations

- **Common Errors:**
  - `FileNotFoundError`: File doesn't exist.
  - `PermissionError`: No permission to access the file.
- **Handling Errors:**

```
try:
    with open("nonexistent.txt", "r") as file:
        content = file.read()
except FileNotFoundError:
    print("File not found!")
except PermissionError:
    print("Permission denied!")
```

Hands-On Practice

- **Task 1:** Read and display the contents of a text file.

```
with open("example.txt", "r") as file:
    print(file.read())
```

- **Task 2:** Write a list of names to a text file.

```
names = ["Ali", "Bilal", "Catherine"]
with open("names.txt", "w") as file:
    for name in names:
        file.write(name + "\n")
```

- **Task 3:** Delete a file after checking if it exists.

```
import os
if os.path.exists("example.txt"):
    os.remove("example.txt")
else:
    print("File does not exist.")
```

- **Task 4:** Read and process a CSV file.

```
import csv
with open("data.csv", "r") as file:
    reader = csv.reader(file)
    for row in reader:
        print(row)
```

- **Task 5:** Write a dictionary to a JSON file.

```
import json
data = {"name": "Ali", "age": 17}
with open("data.json", "w") as file:
    json.dump(data, file)
```

Recap

- Files are used for data persistence and sharing.
- Use `open()` to open files and `close()` to close them.
- Read files using `read()`, `readline()`, or `readlines()`.
- Write files using `write()` or `writelines()`.
- Delete files using `os.remove()` or `os.unlink()`.
- CSV and JSON files are commonly used for structured data.
- Handle file-related errors using `try-except` blocks.

Homework

1. Write a program that reads a text file and writes its content in reverse to a new file.
2. Write a program that reads a CSV file and calculates the average of a numeric column.
3. Write a program that reads a JSON file, modifies its content, and saves it back.
4. Write a program that deletes all `.txt` files in a directory after confirming with the user.

Q&A

- Do you have any questions?
- Share your thoughts.

Closing

Next class: Object-Oriented Programming (OOP)