**File Handling in Python**

File handling in Python allows reading and writing data to files stored on disk. Python provides built-in functions to manage files efficiently.

**1. Opening and Closing Files**

Python provides the open() function to open a file and close() method to close it.

**Opening a File**

file = open("example.txt", "r") # Opens file in read mode

Syntax:

open("filename", "mode")

Where **mode** can be:

* "r" – Read (default)
* "w" – Write (creates a new file if it doesn't exist)
* "a" – Append (adds to the file without overwriting)
* "r+" – Read and write (file must exist)
* "w+" – Write and read (overwrites file)
* "a+" – Append and read

**Closing a File**

file.close()

Alternatively, use the with statement to automatically close the file:

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

content = file.read()

# File is automatically closed outside the `with` block

**2. Different Read Functions**

Python provides several ways to read file contents.

**Reading Entire File**

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

content = file.read() # Reads the entire file

print(content)

**Reading Line by Line**

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

line = file.readline() # Reads one line at a time

print(line)

**Reading All Lines as a List**

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

lines = file.readlines() # Returns a list of lines

print(lines)

**Looping Through a File**

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

for line in file:

print(line.strip()) # Removes newline characters

**3. Writing to a File**

Python provides different modes to write data into a file.

**Writing with "w" Mode (Overwrites Content)**

with open("example.txt", "w") as file:

file.write("Hello, Python!")

**Appending with "a" Mode (Adds Content)**

with open("example.txt", "a") as file:

file.write("\nAppending new line!")

**4. Different Modes - r+ and a+**

**r+ Mode (Read and Write)**

* Reads and writes from the same file.
* The file must exist before using r+.

with open("example.txt", "r+") as file:

print(file.read()) # Read content

file.write("\nWriting in r+ mode") # Write content

**a+ Mode (Append and Read)**

* Appends new content and allows reading.
* If the file doesn’t exist, it creates a new file.

with open("example.txt", "a+") as file:

file.write("\nAppending in a+ mode")

file.seek(0) # Move to the beginning of the file

print(file.read()) # Read entire content

**5. Reading and Writing JSON Data**

Python provides the json module for handling JSON data.

**Reading JSON from a File**

import json

with open("data.json", "r") as file:

data = json.load(file) # Converts JSON to Python dictionary

print(data)

**Writing JSON to a File**

data = {"name": "Alice", "age": 25}

with open("data.json", "w") as file:

json.dump(data, file, indent=4) # Saves dictionary as JSON

**Transforming Python Objects into JSON**

Convert Python objects to JSON using json.dumps():

python\_dict = {"name": "John", "age": 30}

json\_string = json.dumps(python\_dict, indent=2)

print(json\_string)

**6. Reading and Writing Excel and CSV Data**

Python provides pandas and csv modules to handle Excel and CSV files.

**Reading CSV File**

import csv

with open("data.csv", "r") as file:

reader = csv.reader(file)

for row in reader:

print(row)

**Writing to CSV File**

data = [["Name", "Age"], ["Alice", 25], ["Bob", 30]]

with open("data.csv", "w", newline="") as file:

writer = csv.writer(file)

writer.writerows(data)

**Reading Excel File**

import pandas as pd

df = pd.read\_excel("data.xlsx")

print(df)

**Writing to Excel File**

df.to\_excel("output.xlsx", index=False)

**7. Transforming Python Objects into CSV**

Python dictionaries and lists can be converted into CSV files.

import csv

data = [{"Name": "Alice", "Age": 25}, {"Name": "Bob", "Age": 30}]

with open("output.csv", "w", newline="") as file:

fieldnames = ["Name", "Age"]

writer = csv.DictWriter(file, fieldnames=fieldnames)

writer.writeheader()

writer.writerows(data)

**8. Different CSV Dialects**

CSV files can have different delimiters (comma, semicolon, tab).

**Registering a New CSV Dialect**

csv.register\_dialect("semicolon", delimiter=";")

with open("data.csv", "r") as file:

reader = csv.reader(file, dialect="semicolon")

for row in reader:

print(row)

**9. Object Serialization - YAML and JSON**

Serialization converts Python objects to formats like JSON or YAML.

**Using JSON**

import json

data = {"name": "Alice", "age": 25}

json\_string = json.dumps(data)

print(json\_string)

**Using YAML**

import yaml

data = {"name": "Alice", "age": 25}

with open("data.yaml", "w") as file:

yaml.dump(data, file)

with open("data.yaml", "r") as file:

loaded\_data = yaml.safe\_load(file)

print(loaded\_data)

**Summary**

| **Topic** | **Description** |
| --- | --- |
| **Opening & Closing Files** | open() and close() methods |
| **Reading Files** | .read(), .readline(), .readlines() |
| **Writing to Files** | .write(), w, a modes |
| **Modes r+ & a+** | r+ allows read-write; a+ allows appending and reading |
| **JSON Handling** | json.load(), json.dump(), json.dumps() |
| **CSV Handling** | csv.reader(), csv.writer() |
| **Excel Handling** | pandas.read\_excel(), to\_excel() |
| **CSV Dialects** | csv.register\_dialect() for custom formats |
| **Serialization** | JSON (json.dump()), YAML (yaml.dump()) |