**FILE HANDLING**

**What is File Handling?**

File handling in Python allows you to **create, read, write, append, and delete** files. It is crucial for:

* Logging
* Reading configuration files
* Parsing data (e.g., CSV, JSON, YAML)
* Working with output logs in DevOps/automation tasks

**🔹 Modes of File Operation**

| **Mode** | **Name** | **Description** |
| --- | --- | --- |
| 'r' | Read | Default mode. Opens file for reading |
| 'w' | Write | Overwrites file if exists |
| 'a' | Append | Adds to end of file |
| 'x' | Exclusive Create | Fails if file exists |
| 'b' | Binary | Used with other modes (e.g., rb) |
| 't' | Text | Default. Read/write in text mode |
| '+' | Read & Write | Combine read and write |

**🔹 Basic Syntax**

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

# Perform operations

file.close()

**✅ B. OPENING & CLOSING FILES**

**🔹 Example: Open & Read File**

f = open("example.txt", "r") # open file for reading

content = f.read()

print(content)

f.close()

🔍 Explanation:

* open() returns a **file object**
* read() reads the **entire content**
* close() is **mandatory** to free resources

**⚠️ Common Pitfall**

If you forget close(), the file may remain locked or lead to memory leaks.

✅ Use with statement instead:

**✅ C. USING with STATEMENT (CONTEXT MANAGER)**

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

content = f.read()

print(content)

🔍 Why use with?

* Automatically closes the file
* Safer for production/DevOps code
* Prevents accidental file lock/resource leak

**✅ D. READING FILES IN DETAIL**

**1️⃣ read()**

with open("log.txt", "r") as f:

data = f.read()

print(data)

* Reads **entire file**
* Suitable for **small files**

**2️⃣ readline()**

with open("log.txt", "r") as f:

line1 = f.readline()

line2 = f.readline()

* Reads one line at a time
* Useful for **stream processing**

**3️⃣ readlines()**

with open("log.txt", "r") as f:

lines = f.readlines()

for line in lines:

print(line.strip())

* Reads all lines into a **list**
* Can be memory intensive for huge files

**🔁 Iterating Over File**

with open("log.txt") as f:

for line in f:

print(line.strip())

✅ Memory-efficient, preferred for large files

**✅ E. WRITING TO FILES**

**1️⃣ write()**

with open("output.txt", "w") as f:

f.write("This is a line.\n")

f.write("This is another line.\n")

* Overwrites file if exists
* Use \n to create new lines

**2️⃣ writelines()**

lines = ["Line1\n", "Line2\n", "Line3\n"]

with open("out.txt", "w") as f:

f.writelines(lines)

* Writes a list of strings

**⚠️ CAUTION with 'w' mode**

with open("important.txt", "w") as f:

f.write("oops")

👎 This **overwrites** the file. Use 'a' to append.

**✅ F. APPENDING TO FILES**

with open("out.txt", "a") as f:

f.write("Appended line\n")

* Adds content at the **end of file**
* File is created if it doesn’t exist

**✅ G. CHECKING IF FILE EXISTS (BEFORE READ/WRITE)**

import os

if os.path.exists("data.txt"):

with open("data.txt") as f:

print(f.read())

else:

print("File not found!")

**✅ H. FILE POINTERS & CURSOR CONTROL**

**🔹 tell() → Get current cursor position**

with open("sample.txt", "r") as f:

print(f.tell()) # usually 0

f.read(10)

print(f.tell()) # 10

**🔹 seek() → Move cursor**

with open("sample.txt", "r") as f:

f.seek(5) # Move to byte 5

print(f.read(5)) # Reads 5 bytes from there

**✅ I. FILE MODES – COMBINATIONS**

**🔹 Read & Write:**

with open("sample.txt", "r+") as f:

f.write("Hello")

f.seek(0)

print(f.read()) # Read updated content

**🔹 Create File – Fail if Exists:**

with open("new.txt", "x") as f:

f.write("New file")

Raises FileExistsError if file already exists.

**✅ J. BINARY FILES**

with open("image.png", "rb") as f:

data = f.read()

**Use-cases:**

* Images, PDFs, executables
* Avoids character encoding issues

**Copy binary files:**

with open("img.png", "rb") as src:

with open("copy.png", "wb") as dst:

dst.write(src.read())

**✅ K. EXCEPTION HANDLING IN FILE OPERATIONS**

try:

with open("config.yaml", "r") as f:

print(f.read())

except FileNotFoundError:

print("The config file was not found.")

except PermissionError:

print("Permission denied.")

Always add error handling for robust automation

**✅ L. FILE DELETION**

import os

if os.path.exists("delete\_me.txt"):

os.remove("delete\_me.txt")

else:

print("File already gone.")

**✅ M. REAL-WORLD DEVOPS USE CASES**

**1. Log Reader Tool**

with open("/var/log/syslog", "r") as log:

for line in log:

if "ERROR" in line:

print(line.strip())

**2. Generate YAML from Template**

template = """

apiVersion: v1

kind: ConfigMap

metadata:

name: sample-config

data:

key: {value}

"""

with open("config.yaml", "w") as f:

f.write(template.format(value="hello-world"))

**3. File-based Secret Store Reader**

with open("/etc/secrets/db.txt", "r") as f:

db\_pass = f.read().strip()

**4. File Comparison for CI Check**

def compare\_files(f1, f2):

with open(f1) as a, open(f2) as b:

return a.read() == b.read()

**✅ N. BEST PRACTICES**

1. ✅ Always use with for safety
2. ✅ Avoid loading large files with read() — use loops
3. ✅ Use try-except to catch FileNotFoundError
4. ✅ Strip newline characters when reading (line.strip())
5. ✅ Never hardcode file paths — use os.path.join()
6. ✅ Use binary mode (rb, wb) for non-text files
7. ✅ Don’t use 'w' carelessly — it erases content

**✅ O. SUMMARY**

| **Operation** | **Method** |
| --- | --- |
| Read entire | read() |
| Read line | readline() |
| Read lines | readlines() |
| Write | write() |
| Append | a mode |
| Read+Write | r+ or w+ |
| Binary | rb, wb |
| Cursor | seek(), tell() |
| Delete | os.remove() |