**Python File Handling: Complete Guide with Real-World Examples**

### **📄 Introduction**

This guide covers Python's file handling concepts including reading, writing, appending, reading CSVs, using readline(), and more. Every question and example is explained in a practical, project-friendly way.

### **🔧 Basic File Modes in Python**

| **Mode** | **One-Liner Explanation** |
| --- | --- |
| 'r' | Read-only mode (file must exist). |
| 'w' | Write-only mode, overwrites existing file or creates new. |
| 'a' | Append mode, adds content to end of file if it exists. |
| 'r+' | Read and write mode (file must exist). |
| 'w+' | Write and read mode, overwrites or creates new file. |
| 'a+' | Append and read mode, creates file if it doesn’t exist. |
| 'rb' | Read binary file (e.g., images, PDFs). |
| 'wb' | Write binary file, overwrites or creates. |
| 'ab' | Append binary file. |
| 'rb+' | Read and write binary file. |
| 'wb+' | Write and read binary file, overwrites or creates. |
| 'ab+' | Append and read binary file. |

### **✅ Write to a File (w mode)**

file = open("notes.txt", "w")

file.write("Welcome to Python File Handling!\n")

file.write("This is a new file.\n")

file.close()

### **✅ Read a File (r mode)**

file = open("notes.txt", "r")

content = file.read()

print("File Content:\n", content)

file.close()

### **✅ Append to a File (a mode)**

file = open("notes.txt", "a")

file.write("Adding a new line.\n")

file.close()

### **✅ Using with Block (Best Practice)**

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

for line in file:

print(line.strip())

### **✅ User Feedback Logger Example**

feedback = input("Enter your feedback: ")

with open("feedback\_log.txt", "a") as log:

log.write(feedback + "\n")

print("Thanks! Your feedback is saved.")

### **✅ readline() Example**

**Sample data.txt:**

Apple

Banana

Cherry

**Code:**

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

print(file.readline().strip())

print(file.readline().strip())

print(file.readline().strip())

**Loop Version:**

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

while True:

line = file.readline()

if not line:

break

print(line.strip())

**Use Cases for readline()**

* Read specific lines
* Control read position
* Paginated display
* Efficient processing of huge files

### **✅ Difference Between readline() vs for line in file:**

* for line in file: is Pythonic and uses readline() internally
* Use readline() when you want manual control (e.g., skip lines, stop early)

### **✅ What is for \_ in range(n)?**

* \_ is a throwaway variable: you don’t care about the index

for \_ in range(3):

print("Repeat this 3 times")

### **✅ Reading a CSV and Writing to New File**

**Sample input\_file.csv:**

id,name,age

1,John,25

2,Alice,30

3,Bob,22

**Code:**

with open("input\_file.csv", "r") as infile, open("output\_file.csv", "w") as outfile:

for line in infile:

print(line.strip())

outfile.write(line)

### **✅ Read Only the Age Column**

**Using csv.DictReader:**

import csv

with open("input\_file.csv", "r") as file:

reader = csv.DictReader(file)

for row in reader:

print(row["age"])

**Using Indexing with split():**

with open("input\_file.csv", "r") as file:

lines = file.readlines()

for line in lines[1:]:

columns = line.strip().split(",")

print(columns[2])

### **✅ readline() Use Case: Error Filter**

**Sample file.txt:**

INFO: Application started successfully

INFO: User login completed

WARNING: Disk space low

ERROR: Failed to connect to database

INFO: Scheduled task started

ERROR: Null pointer exception occurred

INFO: Application shutdown

**Code:**

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

while True:

line = f.readline()

if not line:

break

if "ERROR" in line:

print("Found error:", line.strip())

### **🔄 Summary Table: Reading Options**

| **Method** | **Use Case** |
| --- | --- |
| read() | Read entire file at once (small files) |
| readlines() | Get list of lines |
| readline() | Read line-by-line manually |
| for line in f | Loop over lines efficiently |
| with block | Safe, auto file closing |

## **✅ 1. Mini Project Idea: Python File Handling Project**

### **🏗️ Project Title:**

**“Feedback Logger with CSV Reporting”**

### **📘 Project Description:**

This project demonstrates practical use of file handling in Python. It allows users to enter feedback via CLI, stores it in a text log file, and generates a summary CSV file from the collected inputs. It showcases read, write, append, readline(), and working with CSV data.

### **🔧 Key Features:**

* Collect user feedback and store each entry in feedback\_log.txt
* Read feedback line by line using readline()
* Filter entries with specific keywords like "ERROR" or "BUG"
* Write the filtered results into a new feedback\_summary.csv file
* Clean and simple console interface

### **🛠️ Tech Used:**

* Python (Built-in open(), readline(), csv module)

### **📁 Directory Structure:**

feedback\_project/

├── feedback\_log.txt

├── feedback\_summary.csv

├── feedback\_collector.py

├── feedback\_filter.py

### **💡 Stretch Goals:**

* Add timestamp to each entry
* Export feedback count report per day
* Use JSON instead of CSV for config/settings

## **🎯 2. How to Explain in an Interview**

When asked **“Have you used file handling in real projects?”** or **“Tell me about a Python mini project you did”**, answer like this:

### **🗣️ Sample Answer (Structured):**

“Yes, I built a simple CLI-based feedback logging system using Python’s file handling features. I used open() with different modes like 'w', 'a', and 'r' to collect and store user feedback in a text file. I used readline() to process large logs line-by-line and applied keyword filtering to extract only relevant lines such as error reports. Finally, I wrote those filtered results to a new CSV file using Python’s csv module. This helped me understand memory-efficient file processing and how to work with both text and structured CSV formats.”

### **🧠 Tips to Impress:**

* Emphasize real-world relevance (e.g., logs, reports, config files)
* Highlight efficiency (readline() over read())
* Mention error handling and file closing using with
* Talk about extensions like adding timestamps, analytics, or integrating with a web form

### **About the Author**

**Gowtham SB** is a **Data Engineering expert, educator,** **and content creator** with a passion for **big data technologies, as well as cloud and Gen AI** . With years of experience in the field, he has worked extensively with **cloud platforms, distributed systems, and data pipelines**, helping professionals and aspiring engineers master the art of data engineering.

Beyond his technical expertise, Gowtham is a **renowned mentor and speaker**, sharing his insights through engaging content on **YouTube and LinkedIn**. He has built one of the **largest Tamil Data Engineering communities**, guiding thousands of learners to excel in their careers.

Through his deep industry knowledge and hands-on approach, Gowtham continues to **bridge the gap between learning and real-world implementation**, empowering individuals to build **scalable, high-performance data solutions**.

𝐒𝐨𝐜𝐢𝐚𝐥𝐬

🎥𝐘𝐨𝐮𝐓𝐮𝐛𝐞 - https://www.youtube.com/@dataengineeringvideos

📸𝐈𝐧𝐬𝐭𝐚𝐠𝐫𝐚𝐦 - <https://instagram.com/dataengineeringtamil>

📸𝐈𝐧𝐬𝐭𝐚𝐠𝐫𝐚𝐦 - [https://instagram.com/](https://instagram.com/dataengineeringtamil)thedatatech.in

🤝𝐂𝐨𝐧𝐧𝐞𝐜𝐭 𝐟𝐨𝐫 𝟏:𝟏 - https://topmate.io/dataengineering/

💼𝐋𝐢𝐧𝐤𝐞𝐝𝐈𝐧 - https://www.linkedin.com/in/sbgowtham/

🌐𝐖𝐞𝐛𝐬𝐢𝐭𝐞 - https://codewithgowtham.blogspot.com

💻𝐆𝐢𝐭𝐇𝐮𝐛 - http://github.com/Gowthamdataengineer

💬𝐖𝐡𝐚𝐭𝐬 𝐀𝐩𝐩 - https://lnkd.in/g5JrHw8q

📧𝐄𝐦𝐚𝐢𝐥 - atozknowledge.com@gmail.com

📱𝐀𝐥𝐥 𝐌𝐲 𝐒𝐨𝐜𝐢𝐚𝐥𝐬 - <https://lnkd.in/gf8k3aCH>