

Part 3 - Working with Files and Folders Using ChatGPT Scripts

Yahya Nazer

2026.06.18

Table of contents

Part 3 - Working with Files and Folders Using ChatGPT Scripts	3
Purpose of This Part	3
Learning Objectives	3
Course Coding Standard Reminder	3
Standard Folder Structure	4
Chapter 1 - Creating Project Folders	4
Objective	4
Start VS Code	4
ChatGPT Script	5
ChatGPT Generated Python Code	5
Run the Program	7
Improve the Script	7
Exercise	7
What You Learned	7
Chapter 2 - Writing a Text File	7
Objective	7
Python File	8
ChatGPT Script	8
ChatGPT Generated Python Code	8
Run the Program	10
Test the Program	10
Improve the Script	10
Exercise	10
What You Learned	10
Chapter 3 - Reading a Text File	11
Objective	11
Setup	11
Python File	11
ChatGPT Script	11
ChatGPT Generated Python Code	12
Improve the Script	13
Exercise	13
What You Learned	14
Chapter 4 - Copying a File Safely	14
Objective	14

Setup	14
Python File	14
ChatGPT Script	14
ChatGPT Generated Python Code	15
Improve the Script	16
Exercise	17
What You Learned	17
Chapter 5 - Scanning a Folder	17
Objective	17
Python File	17
ChatGPT Script	17
ChatGPT Generated Python Code	18
Improve the Script	20
Exercise	20
What You Learned	20
Chapter 6 - Creating a File Inventory Report	20
Objective	20
Python File	20
ChatGPT Script	20
ChatGPT Generated Python Code	21
Required Package	24
Improve the Script	24
Exercise	24
What You Learned	24
Chapter 7 - Saving a Log File	24
Objective	24
Python File	24
ChatGPT Script	25
ChatGPT Generated Python Code	25
Improve the Script	27
Exercise	27
What You Learned	27
Chapter 8 - Debugging File and Folder Errors with ChatGPT	28
Objective	28
Common Errors	28
Debugging Script	28
Example Problem	28
Improve the Script	29
Exercise	29
What You Learned	29
Chapter 9 - Part 3 Review	29
Main Workflow	29
Part 3 Skills Checklist	30
Summary	30
Looking Ahead	30

Part 3 - Working with Files and Folders Using ChatGPT Scripts

Purpose of This Part

In Part 2, you learned how to write better ChatGPT scripts and generate simple Python programs.

In Part 3, you will use the same method to create programs that work with files and folders.

You will learn how to ask ChatGPT to generate Python code that can:

- Create folders
- Read text files
- Write text files
- Copy files
- Scan folders
- Create file lists
- Save logs
- Improve file programs step by step

The main goal is not only to learn Python file commands.

The main goal is to learn how to write a clear ChatGPT script that produces useful file-management Python code.

Learning Objectives

After completing this part, you will be able to:

- Write ChatGPT scripts for file and folder programs
- Ask ChatGPT to use a standard project folder structure
- Create folders using Python
- Read text files
- Write text files
- Copy files safely
- Scan folders
- Create file inventory reports
- Save log files
- Improve scripts after testing
- Ask ChatGPT to debug file errors

Course Coding Standard Reminder

Every Python program in this course should include:

- Program name
- Purpose
- User ChatGPT script
- Expected output
- Version
- Clearly labeled steps
- Comments explaining important lines

Example:

```
# =====  
# STEP 1 - Import Libraries  
# =====
```

Standard Folder Structure

Use this structure for all programs in Part 3:

```
ChatGPT-Python-Course  
  A-Data  
  B-Engine  
  C-Results  
  D-Documentation
```

Python files go in:

```
B-Engine
```

Input files go in:

```
A-Data
```

Output files go in:

```
C-Results
```

Notes and instructions go in:

```
D-Documentation
```

Chapter 1 - Creating Project Folders

Objective

Create a Python program that creates the standard project folders automatically.

The program should create:

```
A-Data  
B-Engine  
C-Results  
D-Documentation
```

Start VS Code

1. Open VS Code.
2. Open your main course folder.
3. Open the **B-Engine** folder.
4. Create a file named:

```
Create-Project-Folders.py
```

ChatGPT Script

Copy this into ChatGPT:

Write a Python program.

Project folder structure:

Top Folder

+- A-Data

+- B-Engine

+- C-Results

+- D-Documentation

Program Name:

Create-Project-Folders.py

Program purpose:

Create the standard course folders if they do not already exist.

Requirements:

1. Create A-Data.
2. Create B-Engine.
3. Create C-Results.
4. Create D-Documentation.
5. Do not delete anything.
6. If a folder already exists, do not create an error.
7. Print a message for each folder.

Coding Standards:

1. Add a detailed program header at the top of the Python file.
2. The header must include Program Name, Purpose, User ChatGPT Script, Expected Output, and Version.
3. Divide the code into clearly labeled steps.
4. Add comments explaining every important line.
5. Make the code suitable for beginners.

Expected output:

The program creates the folders and prints which folders exist.

ChatGPT Generated Python Code

```
# =====  
# Program Name:  
# Create-Project-Folders.py  
#  
# Purpose:  
# Create the standard course folders if they do not already exist.  
#  
# User ChatGPT Script:  
# Write a Python program.  
#  
# Program purpose:  
# Create A-Data, B-Engine, C-Results, and D-Documentation.  
#
```

```

# Requirements:
# 1. Create A-Data.
# 2. Create B-Engine.
# 3. Create C-Results.
# 4. Create D-Documentation.
# 5. Do not delete anything.
# 6. If a folder already exists, do not create an error.
# 7. Print a message for each folder.
#
# Expected Output:
# The program creates the folders and prints which folders exist.
#
# Version:
# 1.0
# =====

# =====
# STEP 1 - Import Libraries
# =====

# Import Path from pathlib to work with folders and files.
from pathlib import Path

# =====
# STEP 2 - Define Folder Names
# =====

# Create a list of folder names used in this course.
folder_names = [
    "A-Data",
    "B-Engine",
    "C-Results",
    "D-Documentation"
]

# =====
# STEP 3 - Create Folders
# =====

# Loop through each folder name.
for folder_name in folder_names:

    # Convert the folder name into a Path object.
    folder_path = Path(folder_name)

    # Create the folder if it does not already exist.
    folder_path.mkdir(exist_ok=True)

    # Print a message for the user.
    print("Folder ready:", folder_path)

# =====
# STEP 4 - End Program

```

```
# =====  
  
# Print completion message.  
print()  
print("Project folder setup is complete.")
```

Run the Program

In VS Code Terminal:

```
python Create-Project-Folders.py
```

On some Mac computers:

```
python3 Create-Project-Folders.py
```

Improve the Script

Ask ChatGPT:

```
Please improve the program.
```

```
New requirements:
```

1. Also create a README.txt file in D-Documentation.
 2. Write a short explanation of the folder structure into README.txt.
 3. Keep the program header.
 4. Keep the code divided into steps.
 5. Add comments for beginners.
-

Exercise

Write a ChatGPT script for a program that creates this folder structure:

```
Photo-Project  
+- A-Originals  
+- B-Edited  
+- C-Print  
+- D-Archive
```

What You Learned

You learned how to ask ChatGPT to generate Python code that creates folders safely.

Chapter 2 - Writing a Text File

Objective

Create a Python program that writes a simple note into a text file.

The output file should be saved in:

```
C-Results
```

Python File

Create:

```
B-Engine/Write-Text-File.py
```

ChatGPT Script

Copy this into ChatGPT:

```
Write a Python program.
```

```
Project folder structure:
```

```
Top Folder
```

```
+-- A-Data
```

```
+-- B-Engine
```

```
+-- C-Results
```

```
+-- D-Documentation
```

```
Program Name:
```

```
Write-Text-File.py
```

```
Program purpose:
```

```
Create a text file in C-Results and write a short message into it.
```

```
Requirements:
```

1. Create C-Results if it does not exist.
2. Create a file named My-First-Note.txt.
3. Write three lines of text into the file.
4. Use UTF-8 encoding.
5. Print the path of the saved file.

```
Coding Standards:
```

1. Add a detailed program header at the top of the Python file.
2. The header must include Program Name, Purpose, User ChatGPT Script, Expected Output, and Version.
3. Divide the code into clearly labeled steps.
4. Add comments explaining every important line.
5. Make the code suitable for beginners.

```
Expected output:
```

```
A text file is created in C-Results.
```

ChatGPT Generated Python Code

```
# =====  
# Program Name:  
# Write-Text-File.py
```

```

#
# Purpose:
# Create a text file in C-Results and write a short message into it.
#
# User ChatGPT Script:
# Write a Python program that creates My-First-Note.txt
# in C-Results and writes three lines using UTF-8 encoding.
#
# Expected Output:
# A text file is created in C-Results.
#
# Version:
# 1.0
# =====

# =====
# STEP 1 - Import Libraries
# =====

# Import Path to work with files and folders.
from pathlib import Path

# =====
# STEP 2 - Create Results Folder
# =====

# Define the results folder.
results_folder = Path("C-Results")

# Create the results folder if it does not exist.
results_folder.mkdir(exist_ok=True)

# =====
# STEP 3 - Create Text Content
# =====

# Create the text that will be written to the file.
note_text = """My First Python Note
This file was created by a Python program.
I used ChatGPT to help generate this code.
"""

# =====
# STEP 4 - Save Text File
# =====

# Define the output file path.
output_file = results_folder / "My-First-Note.txt"

# Write the text to the file using UTF-8 encoding.
output_file.write_text(note_text, encoding="utf-8")

# =====

```

```
# STEP 5 - Display Result
# =====

# Print where the file was saved.
print("File saved to:", output_file)

# Print completion message.
print("Program finished.")
```

Run the Program

```
python Write-Text-File.py
```

or:

```
python3 Write-Text-File.py
```

Test the Program

After running the program:

1. Open C-Results.
 2. Find My-First-Note.txt.
 3. Open the file.
 4. Confirm the text was written.
-

Improve the Script

Ask ChatGPT:

```
Please improve the program.
```

```
New requirements:
```

1. Ask the user to type a note.
 2. Save the user's note into C-Results/User-Note.txt.
 3. Add the current date and time at the top of the note.
 4. Keep the program header.
 5. Keep the step-by-step comments.
-

Exercise

Write a ChatGPT script for a program that saves a shopping list into a text file.

What You Learned

You learned how to ask ChatGPT to create a Python program that writes text files.

Chapter 3 - Reading a Text File

Objective

Create a Python program that reads a text file from A-Data and prints the content.

Setup

Create this file manually:

```
A-Data/Sample-Input.txt
```

Add this text:

```
This is a sample input file.  
Python can read this file.  
ChatGPT helped create the program.
```

Python File

Create:

```
B-Engine/Read-Text-File.py
```

ChatGPT Script

```
Write a Python program.
```

```
Project folder structure:
```

```
Top Folder  
+- A-Data  
+- B-Engine  
+- C-Results  
+- D-Documentation
```

```
Program Name:  
Read-Text-File.py
```

```
Program purpose:  
Read a text file from A-Data and display its content.
```

```
Requirements:  
1. Read A-Data/Sample-Input.txt.  
2. Use UTF-8 encoding.  
3. Print the file content.  
4. If the file is missing, show a friendly error message.  
5. Do not crash if the file is missing.
```

```
Coding Standards:  
1. Add a detailed program header at the top of the Python file.  
2. The header must include Program Name, Purpose, User ChatGPT Script, Expected Output, and Version.
```

3. Divide the code into clearly labeled steps.
4. Add comments explaining every important line.
5. Make the code suitable for beginners.

Expected output:

The program prints the content of Sample-Input.txt.

ChatGPT Generated Python Code

```
# =====
# Program Name:
# Read-Text-File.py
#
# Purpose:
# Read a text file from A-Data and display its content.
#
# User ChatGPT Script:
# Write a Python program that reads A-Data/Sample-Input.txt
# using UTF-8 and prints the content. If missing, show a
# friendly error message.
#
# Expected Output:
# The program prints the content of Sample-Input.txt.
#
# Version:
# 1.0
# =====

# =====
# STEP 1 - Import Libraries
# =====

# Import Path to work with file paths.
from pathlib import Path

# =====
# STEP 2 - Define Input File
# =====

# Define the input file path.
input_file = Path("A-Data") / "Sample-Input.txt"

# =====
# STEP 3 - Check if File Exists
# =====

# Check whether the input file exists.
if not input_file.exists():

    # Show a friendly error message.
    print("ERROR: The input file was not found.")
```

```

print("Expected file:", input_file)

# If the file exists, read it.
else:

    # =====
    # STEP 4 - Read File Content
    # =====

    # Read the text file using UTF-8 encoding.
    content = input_file.read_text(encoding="utf-8")

    # =====
    # STEP 5 - Display File Content
    # =====

    # Print title.
    print("File Content")
    print("-----")

    # Print the content.
    print(content)

# =====
# STEP 6 - End Program
# =====

# Print completion message.
print("Program finished.")

```

Improve the Script

Ask ChatGPT:

Please improve the program.

New requirements:

1. Count the number of lines in the file.
2. Count the number of words in the file.
3. Display the counts after the content.
4. Save the counts to C-Results/File-Summary.txt.
5. Keep the program header and step comments.

Exercise

Write a ChatGPT script for a program that reads:

A-Data/Notes.txt

and prints the number of characters.

What You Learned

You learned how to ask ChatGPT to generate Python code that reads text files safely.

Chapter 4 - Copying a File Safely

Objective

Create a Python program that copies a file from **A-Data** to **C-Results**.

The program should not delete the original file.

Setup

Create this file:

```
A-Data/Sample-Input.txt
```

Python File

Create:

```
B-Engine/Copy-File.py
```

ChatGPT Script

```
Write a Python program.
```

```
Project folder structure:
```

```
Top Folder
```

```
+ - A-Data
```

```
+ - B-Engine
```

```
+ - C-Results
```

```
+ - D-Documentation
```

```
Program Name:
```

```
Copy-File.py
```

```
Program purpose:
```

```
Copy A-Data/Sample-Input.txt to C-Results.
```

```
Requirements:
```

1. Use `pathlib`.
2. Use `shutil.copy2` so the file date is preserved.
3. Create `C-Results` if it does not exist.
4. Do not delete the original file.
5. If the source file is missing, show a friendly error message.
6. Print what happened.

Coding Standards:

1. Add a detailed program header at the top of the Python file.
2. The header must include Program Name, Purpose, User ChatGPT Script, Expected Output, and Version.
3. Divide the code into clearly labeled steps.
4. Add comments explaining every important line.
5. Make the code suitable for beginners.

Expected output:

The file is copied from A-Data to C-Results.

ChatGPT Generated Python Code

```
# =====
# Program Name:
# Copy-File.py
#
# Purpose:
# Copy A-Data/Sample-Input.txt to C-Results.
#
# User ChatGPT Script:
# Write a Python program that copies A-Data/Sample-Input.txt
# to C-Results using pathlib and shutil.copy2.
#
# Expected Output:
# The file is copied from A-Data to C-Results.
#
# Version:
# 1.0
# =====

# =====
# STEP 1 - Import Libraries
# =====

# Import Path for file paths.
from pathlib import Path

# Import shutil for file copying.
import shutil

# =====
# STEP 2 - Define Source and Destination
# =====

# Define the source file.
source_file = Path("A-Data") / "Sample-Input.txt"

# Define the destination folder.
destination_folder = Path("C-Results")

# Create destination folder if it does not exist.
```

```

destination_folder.mkdir(exist_ok=True)

# Define destination file path.
destination_file = destination_folder / source_file.name

# =====
# STEP 3 - Check Source File
# =====

# Check whether the source file exists.
if not source_file.exists():

    # Print friendly error message.
    print("ERROR: Source file was not found.")
    print("Expected file:", source_file)

else:

    # =====
    # STEP 4 - Copy File
    # =====

    # Copy the file and preserve metadata such as modified date.
    shutil.copy2(source_file, destination_file)

    # =====
    # STEP 5 - Display Result
    # =====

    # Print what happened.
    print("Copied from:", source_file)
    print("Copied to  :", destination_file)

# =====
# STEP 6 - End Program
# =====

# Print completion message.
print("Program finished.")

```

Improve the Script

Ask ChatGPT:

Please improve the program.

New requirements:

1. Add a timestamp to the copied filename.
 2. Save the copy as Sample-Input-yyyy-mm-dd--hh-mm.txt.
 3. Keep the original file unchanged.
 4. Keep the program header and step comments.
-

Exercise

Write a ChatGPT script for a program that copies all `.txt` files from `A-Data` to `C-Results`.

What You Learned

You learned how to ask ChatGPT to create safe copy programs.

Chapter 5 - Scanning a Folder

Objective

Create a Python program that scans `A-Data` and lists all files.

Python File

Create:

```
B-Engine/List-Files.py
```

ChatGPT Script

```
Write a Python program.
```

```
Project folder structure:
```

```
Top Folder  
+- A-Data  
+- B-Engine  
+- C-Results  
+- D-Documentation
```

```
Program Name:
```

```
List-Files.py
```

```
Program purpose:
```

```
Scan A-Data and list all files.
```

```
Requirements:
```

1. Use `pathlib`.
2. Scan `A-Data` and its subfolders.
3. Print each file path.
4. Count the number of files.
5. Save the file list to `C-Results/File-List.txt`.
6. Create `C-Results` if missing.

```
Coding Standards:
```

1. Add a detailed program header at the top of the Python file.
2. The header must include Program Name, Purpose, User ChatGPT Script, Expected Output, and Version.

3. Divide the code into clearly labeled steps.
4. Add comments explaining every important line.
5. Make the code suitable for beginners.

Expected output:

A file list is printed and saved.

ChatGPT Generated Python Code

```
# =====  
# Program Name:  
# List-Files.py  
#  
# Purpose:  
# Scan A-Data and list all files.  
#  
# User ChatGPT Script:  
# Write a Python program that scans A-Data and subfolders,  
# prints each file path, counts files, and saves the list.  
#  
# Expected Output:  
# A file list is printed and saved.  
#  
# Version:  
# 1.0  
# =====  
  
# =====  
# STEP 1 - Import Libraries  
# =====  
  
# Import Path to work with folders and files.  
from pathlib import Path  
  
# =====  
# STEP 2 - Define Folders  
# =====  
  
# Define the source folder.  
source_folder = Path("A-Data")  
  
# Define the results folder.  
results_folder = Path("C-Results")  
  
# Create results folder if needed.  
results_folder.mkdir(exist_ok=True)  
  
# =====  
# STEP 3 - Scan Files  
# =====
```

```

# Create an empty list to store file paths.
file_list = []

# Scan all files in A-Data and subfolders.
for file_path in source_folder.rglob("*"):

    # Only process files, not folders.
    if file_path.is_file():

        # Add file path to list.
        file_list.append(str(file_path))

# =====
# STEP 4 - Display Results
# =====

# Print report title.
print("File List")
print("-----")

# Print each file.
for file_name in file_list:
    print(file_name)

# Print total count.
print()
print("Total files:", len(file_list))

# =====
# STEP 5 - Save Results
# =====

# Define output file.
output_file = results_folder / "File-List.txt"

# Create report text.
report_text = "\n".join(file_list)
report_text += f"\n\nTotal files: {len(file_list)}\n"

# Save report.
output_file.write_text(report_text, encoding="utf-8")

# =====
# STEP 6 - End Program
# =====

# Print saved location.
print("File list saved to:", output_file)

# Print completion message.
print("Program finished.")

```

Improve the Script

Ask ChatGPT:

```
Please improve the program.
```

```
New requirements:
```

1. Ignore folders starting with Z-.
 2. Ignore folders starting with underscore.
 3. Ignore hidden folders starting with dot.
 4. Save the results as CSV.
 5. Include file name, folder, extension, and size.
 6. Keep the header and step comments.
-

Exercise

Write a ChatGPT script for a program that lists only image files.

What You Learned

You learned how to ask ChatGPT to create folder scanning programs.

Chapter 6 - Creating a File Inventory Report

Objective

Create a Python program that scans A-Data and creates an Excel file inventory report.

This chapter introduces the idea of a useful report program.

Python File

Create:

```
B-Engine/File-Inventory-Report.py
```

ChatGPT Script

```
Write a Python program.
```

```
Project folder structure:
```

```
Top Folder  
+- A-Data  
+- B-Engine  
+- C-Results  
+- D-Documentation
```

```
Program Name:
```

File-Inventory-Report.py

Program purpose:

Scan A-Data and create an Excel file inventory report.

Requirements:

1. Use pathlib.
2. Use pandas.
3. Scan A-Data and subfolders.
4. Ignore folders starting with Z-.
5. Ignore folders starting with underscore.
6. Ignore hidden folders starting with dot.
7. Create a report with file name, extension, folder, size in bytes, and modified date.
8. Save the report to C-Results/File-Inventory.xlsx.
9. Add clear print messages.

Coding Standards:

1. Add a detailed program header at the top of the Python file.
2. The header must include Program Name, Purpose, User ChatGPT Script, Expected Output, and Version.
3. Divide the code into clearly labeled steps.
4. Add comments explaining every important line.
5. Make the code suitable for beginners.

Expected output:

An Excel file inventory report is saved to C-Results.

ChatGPT Generated Python Code

```
# =====  
# Program Name:  
# File-Inventory-Report.py  
#  
# Purpose:  
# Scan A-Data and create an Excel file inventory report.  
#  
# User ChatGPT Script:  
# Write a Python program that scans A-Data and subfolders,  
# ignores Z-, underscore, and hidden folders, and saves an  
# Excel file inventory report.  
#  
# Expected Output:  
# An Excel file inventory report is saved to C-Results.  
#  
# Version:  
# 1.0  
# =====  
  
# =====  
# STEP 1 - Import Libraries  
# =====
```

```

# Import Path to work with folders and files.
from pathlib import Path

# Import datetime to convert modified time.
from datetime import datetime

# Import pandas to create Excel report.
import pandas as pd

# =====
# STEP 2 - Define Folders
# =====

# Define the source folder.
source_folder = Path("A-Data")

# Define the results folder.
results_folder = Path("C-Results")

# Create results folder if needed.
results_folder.mkdir(exist_ok=True)

# =====
# STEP 3 - Define Helper Function
# =====

def should_skip_folder(folder_path):
    """
    Return True if the folder should be ignored.
    """

    # Check every part of the folder path.
    for part in folder_path.parts:

        # Skip folders starting with Z-.
        if part.startswith("Z-"):

            return True

        # Skip folders starting with underscore.
        if part.startswith("_"):

            return True

        # Skip hidden folders starting with dot.
        if part.startswith("."):

            return True

    # If none of the rules match, do not skip.
    return False

# =====

```

```

# STEP 4 - Scan Files
# =====

# Create empty list for records.
records = []

# Scan files recursively.
for file_path in source_folder.rglob("*"):

    # Skip folders we do not want.
    if should_skip_folder(file_path.parent):

        continue

    # Process only files.
    if file_path.is_file():

        # Get file information.
        stat = file_path.stat()

        # Add one record.
        records.append({
            "file_name": file_path.name,
            "extension": file_path.suffix.lower(),
            "folder": str(file_path.parent),
            "size_bytes": stat.st_size,
            "modified_date": datetime.fromtimestamp(stat.st_mtime)
        })

# =====
# STEP 5 - Create Excel Report
# =====

# Convert records to a pandas DataFrame.
report = pd.DataFrame(records)

# Define output file.
output_file = results_folder / "File-Inventory.xlsx"

# Save report to Excel.
report.to_excel(output_file, index=False)

# =====
# STEP 6 - Display Summary
# =====

# Print summary.
print("Files found:", len(records))
print("Report saved to:", output_file)

# =====
# STEP 7 - End Program
# =====

```

```
# Print completion message.  
print("Program finished.")
```

Required Package

If pandas is missing, install it:

```
python -m pip install pandas openpyxl
```

On Mac:

```
python3 -m pip install pandas openpyxl
```

Improve the Script

Ask ChatGPT:

```
Please improve the program.
```

```
New requirements:
```

1. Add file size in MB.
 2. Add relative path.
 3. Add file category based on extension.
 4. Auto-adjust Excel column widths using openpyxl.
 5. Keep the header and step comments.
-

Exercise

Write a ChatGPT script that creates a file inventory only for PDF files.

What You Learned

You learned how to ask ChatGPT to generate a useful file inventory report.

Chapter 7 - Saving a Log File

Objective

Create a Python program that saves a log file while processing files.

Logs help users understand what happened.

Python File

Create:

ChatGPT Script

Write a Python program.

Project folder structure:

Top Folder

+– A-Data

+– B-Engine

+– C-Results

+– D-Documentation

Program Name:

Save-Log-File.py

Program purpose:

Create a log file in C-Results.

Requirements:

1. Create C-Results if missing.
2. Create a timestamped log file.
3. Add log messages with time.
4. Print the same messages to the screen.
5. Save the log file using UTF-8.

Coding Standards:

1. Add a detailed program header at the top of the Python file.
2. The header must include Program Name, Purpose, User ChatGPT Script, Expected Output, and Version.
3. Divide the code into clearly labeled steps.
4. Add comments explaining every important line.
5. Make the code suitable for beginners.

Expected output:

A timestamped log file is saved to C-Results.

ChatGPT Generated Python Code

```
# =====  
# Program Name:  
# Save-Log-File.py  
#  
# Purpose:  
# Create a timestamped log file in C-Results.  
#  
# User ChatGPT Script:  
# Write a Python program that creates a timestamped log file,  
# adds log messages with time, prints messages, and saves UTF-8.  
#
```

```

# Expected Output:
# A timestamped log file is saved to C-Results.
#
# Version:
# 1.0
# =====

# =====
# STEP 1 - Import Libraries
# =====

# Import Path to work with folders and files.
from pathlib import Path

# Import datetime for date and time.
from datetime import datetime

# =====
# STEP 2 - Create Results Folder
# =====

# Define results folder.
results_folder = Path("C-Results")

# Create results folder if missing.
results_folder.mkdir(exist_ok=True)

# =====
# STEP 3 - Define Log Function
# =====

# Create an empty list to store log lines.
log_lines = []

def log(message):
    """
    Print a message and save it in the log list.
    """

    # Create current time.
    current_time = datetime.now().strftime("%H:%M:%S")

    # Create formatted log line.
    line = f"[{current_time}] {message}"

    # Print line to screen.
    print(line)

    # Add line to list.
    log_lines.append(line)

# =====
# STEP 4 - Write Log Messages

```

```

# =====
# Write sample log messages.
log("Program started.")
log("Checking folders.")
log("Processing files.")
log("Program finished.")

# =====
# STEP 5 - Save Log File
# =====

# Create timestamp for filename.
timestamp = datetime.now().strftime("%Y-%m-%d--%H-%M")

# Define log file path.
log_file = results_folder / f"Log-{timestamp}.txt"

# Save log lines using UTF-8.
log_file.write_text("\n".join(log_lines), encoding="utf-8")

# =====
# STEP 6 - End Program
# =====

# Print saved file path.
print()
print("Log saved to:", log_file)

```

Improve the Script

Ask ChatGPT:

Please improve this program.

New requirements:

1. Add log levels: INFO, WARNING, ERROR.
2. Save the log as CSV.
3. Include date, time, level, and message.
4. Keep the program header and step comments.

Exercise

Add a log function to one of your earlier programs.

What You Learned

You learned how to ask ChatGPT to create logging code.

Chapter 8 - Debugging File and Folder Errors with ChatGPT

Objective

Learn how to ask ChatGPT to debug file and folder errors.

Common Errors

Examples:

```
FileNotFoundError
PermissionError
ModuleNotFoundError
UnicodeDecodeError
```

Debugging Script

Use this script when a file program fails:

```
My Python file/folder program has an error.
```

```
Please help me debug it.
```

```
Explain:
```

1. What the error means.
2. Which line likely caused it.
3. How to fix it.
4. How to avoid it next time.
5. Whether the folder structure may be wrong.

```
Project folder structure:
```

```
Top Folder
+- A-Data
+- B-Engine
+- C-Results
+- D-Documentation
```

```
Here is my code:
```

```
[paste code here]
```

```
Here is the full error message:
```

```
[paste error message here]
```

Example Problem

If the program says:

```
FileNotFoundError
```

Possible causes:

- The file name is misspelled.
 - The file is in the wrong folder.
 - The program is running from the wrong working directory.
 - The path is wrong.
-

Improve the Script

Ask ChatGPT:

```
Please modify my program so it prints:  
1. Current working directory.  
2. Expected input file path.  
3. Whether the input file exists.  
4. A friendly explanation if the file is missing.
```

Exercise

Create a missing file error and ask ChatGPT to debug it.

What You Learned

You learned how to use ChatGPT to debug file and folder problems.

Chapter 9 - Part 3 Review

Main Workflow

In Part 3, every program followed this workflow:

```
Objective  
↓  
Folder Structure  
↓  
VS Code File  
↓  
ChatGPT Script  
↓  
Generated Python Header  
↓  
Step-by-Step Python Code  
↓  
Run and Test  
↓  
Improve Script
```

Part 3 Skills Checklist

By the end of Part 3, you should be able to:

- Write ChatGPT scripts for file programs
 - Create folders with Python
 - Write text files
 - Read text files
 - Copy files safely
 - Scan folders
 - Create file lists
 - Create Excel file inventory reports
 - Save log files
 - Ask ChatGPT to debug file errors
-

Summary

In Part 3, you learned how to use ChatGPT scripts to generate Python programs that work with files and folders.

You also learned an important lesson:

```
File programs must be careful.  
They should check paths.  
They should avoid deleting files.  
They should print clear messages.  
They should save logs when useful.
```

Looking Ahead

In Part 4, you will use the same method to build your first GUI applications with tkinter.

You will learn how to ask ChatGPT to generate programs with:

- Buttons
- Labels
- Text boxes
- Folder selection
- Log windows
- Color-coded workflow buttons