Part 5D - Practical Excel Projects and Debugging Using ChatGPT Scripts

Yahya Nazer

2026.06.18

Table of contents

Part 5D - Practical Excel Projects and Debugging Using ChatGPT Scripts

Purpose of This Part

In Parts 5A, 5B, and 5C, you learned how to use ChatGPT scripts to generate Python programs for CSV and Excel files.

You learned how to:

- Read CSV files
- Write CSV files
- Read Excel files
- Write Excel files
- Validate Excel data
- Clean Excel data
- Combine multiple Excel files
- Generate Excel reports

In Part 5D, you will apply these skills to practical projects.

You will build:

- A photo inventory workbook
- An Excel analyzer GUI
- A student grade analyzer
- Debugging workflows for Excel programs
- A complete Part 5 review

The main goal is still the same:

```
Business Problem
        ↓
ChatGPT Script
        ↓
Generated Python Program
```

```
        ↓
Test
        ↓
Improve Script
        ↓
Improve Python Program
```

Learning Objectives

After completing Part 5D, you will be able to:

- ☐ Create practical Excel project scripts for ChatGPT
- ☐ Generate a photo inventory workbook
- ☐ Build a GUI Excel analyzer
- ☐ Analyze student grade data
- ☐ Save Excel reports to C-Results
- ☐ Add logs to Excel programs
- ☐ Debug Excel-related errors with ChatGPT
- ☐ Improve Python programs using better scripts
- ☐ Review the complete Part 5 workflow

Course Coding Standard

Every Python program should include:

- Program Name
- Purpose
- User ChatGPT Script
- Expected Output
- Version
- Clearly labeled steps
- Beginner-friendly comments

Example:

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

Standard Folder Structure

Use this folder structure:

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

Chapter 13 - Photo Inventory Workbook

Objective

Create a Python program that scans a folder of photos and creates an Excel inventory workbook.

The workbook should include:

- File name
- Folder path
- File extension
- File size
- Modified date

This project is useful when organizing photo libraries, camera folders, and image archives.

Folder Setup

Place images in:

```
A-Data/Photos
```

Example files:

```
IMG_0001.jpg
IMG_0002.png
Vacation-Photo.jpeg
Family-Photo.tif
```

Python File

Create:

```
B-Engine/Photo-Inventory-Workbook.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:
Photo-Inventory-Workbook.py

Program purpose:
Scan A-Data/Photos and create an Excel inventory workbook for image files.
```

```
Requirements:
1. Use pathlib.
2. Use pandas.
3. Scan A-Data/Photos and all subfolders.
4. Include only image files:
   .jpg, .jpeg, .png, .tif, .tiff, .heic
5. Create a report with:
   - file_name
   - extension
   - folder
   - full_path
   - size_bytes
   - size_mb
   - modified_date
6. Save the report to C-Results/Photo-Inventory.xlsx.
7. Print how many image files were found.
8. If the Photos folder is missing, show a friendly error message.

Coding Standards:
1. Add a detailed program header.
2. Include Program Name, Purpose, User ChatGPT Script, Expected Output, and Version.
3. Divide code into clearly labeled steps.
4. Add comments explaining every important line.
5. Make code beginner friendly.

Expected output:
An Excel photo inventory workbook is saved to C-Results.
```

ChatGPT Generated Python Code

```
# ============================================================
# Program Name:
# Photo-Inventory-Workbook.py
#
# Purpose:
# Scan A-Data/Photos and create an Excel inventory workbook
# for image files.
#
# User ChatGPT Script:
# Write a Python program that scans A-Data/Photos and all
# subfolders, finds image files, and saves an Excel inventory
# workbook to C-Results.
#
# Expected Output:
# An Excel photo inventory workbook 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 dates.
from datetime import datetime

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

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

# Define the photo folder.
photo_folder = Path("A-Data") / "Photos"

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

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

# Define the output Excel file.
output_file = results_folder / "Photo-Inventory.xlsx"

# ============================================================
# STEP 3 - Define Image Extensions
# ============================================================

# Create a set of image file extensions.
image_extensions = {
    ".jpg",
    ".jpeg",
    ".png",
    ".tif",
    ".tiff",
    ".heic"
}

# ============================================================
# STEP 4 - Check Photo Folder
# ============================================================

# If photo folder does not exist, show a friendly message.
if not photo_folder.exists():

    print("ERROR: Photo folder was not found.")
    print("Expected folder:", photo_folder)

else:
```

```
    # ============================================================
    # STEP 5 - Scan Photo Files
    # ============================================================

    # Create an empty list for records.
    records = []

    # Scan all files in the photo folder and subfolders.
    for file_path in photo_folder.rglob("*"):

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

            # Check if file extension is an image extension.
            if file_path.suffix.lower() in image_extensions:

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

                # Add one record to the list.
                records.append({
                    "file_name": file_path.name,
                    "extension": file_path.suffix.lower(),
                    "folder": str(file_path.parent),
                    "full_path": str(file_path),
                    "size_bytes": stat.st_size,
                    "size_mb": round(stat.st_size / (1024 * 1024), 2),
                    "modified_date": datetime.fromtimestamp(stat.st_mtime)
                })

    # ============================================================
    # STEP 6 - Create Excel Inventory
    # ============================================================

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

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

    # ============================================================
    # STEP 7 - Display Summary
    # ============================================================

    # Print summary information.
    print("Image files found:", len(records))
    print("Photo inventory saved to:", output_file)

# ============================================================
# STEP 8 - End Program
# ============================================================

# Print completion message.
```

```
print("Program finished.")
```

Run the Program

```
python Photo-Inventory-Workbook.py
```

On some Mac computers:

```
python3 Photo-Inventory-Workbook.py
```

Improve the Script

Ask ChatGPT:

```
Please improve the program.

New requirements:
1. Add image width and height using Pillow.
2. Add a column called orientation with values Landscape, Portrait, or Square.
3. Add a timestamp to the output filename.
4. Auto-adjust Excel column widths using openpyxl.
5. Keep the program header and step comments.
```

Exercise

Write a ChatGPT script for a program that creates a camera file inventory.

The program should include:

- File name
- Extension
- Size
- Modified date
- Camera folder
- Possible RAW file extensions

What You Learned

You learned how to ask ChatGPT to generate a practical photo inventory workbook.

Chapter 14 - Excel Analyzer GUI

Objective

Create a GUI program that lets the user select an Excel file and analyze it.

The GUI should:

- Select an Excel file
- Display selected file path
- Analyze rows and columns
- Count missing values
- Count duplicate rows
- Display results in a log window
- Save a report to C-Results

Python File

Create:

```
B-Engine/Excel-Analyzer-GUI.py
```

ChatGPT Script

Copy this into ChatGPT:

```
Write a Python tkinter GUI program.

Project folder structure:
Top Folder
+- A-Data
+- B-Engine
+- C-Results
+- D-Documentation

Program Name:
Excel-Analyzer-GUI.py

Program purpose:
Create a GUI tool that selects an Excel file and analyzes it.

Requirements:
1. Use tkinter.
2. Use pandas.
3. Add a button to select an Excel file.
4. Display the selected file path.
5. Add a button to analyze the file.
6. Analyze:
   - row count
   - column count
   - column names
   - missing values by column
   - duplicate row count
7. Display results in a scrolling log window.
8. Save analysis report to C-Results/Excel-Analysis-Report.txt.
9. Add a Clear button.
10. Add an Exit button.
11. Use color-coded buttons:
    - green for next action
    - yellow while working
```

```
    - blue when complete
12. Use UTF-8 when saving reports.

Coding Standards:
1. Add a detailed program header.
2. Include Program Name, Purpose, User ChatGPT Script, Expected Output, and Version.
3. Divide code into clearly labeled steps.
4. Add comments explaining every important line.
5. Make code beginner friendly.

Expected output:
A GUI appears and creates an Excel analysis report.
```

ChatGPT Generated Python Code

```
# ============================================================
# Program Name:
# Excel-Analyzer-GUI.py
#
# Purpose:
# Create a GUI tool that selects an Excel file and analyzes it.
#
# User ChatGPT Script:
# Write a tkinter GUI program that selects an Excel file,
# analyzes rows, columns, missing values, duplicate rows,
# displays results in a log window, and saves a report.
#
# Expected Output:
# A GUI appears and creates an Excel analysis report.
#
# Version:
# 1.0
# ============================================================

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

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

# Import datetime for timestamped reports.
from datetime import datetime

# Import tkinter for GUI.
import tkinter as tk

# Import filedialog for selecting files.
from tkinter import filedialog

# Import messagebox for showing messages.
```

```
from tkinter import messagebox

# Import scrolledtext for the log window.
from tkinter import scrolledtext

# Import pandas for Excel analysis.
import pandas as pd

# ============================================================
# STEP 2 - Define Colors
# ============================================================

# Define button colors.
COLOR_NEXT = "#90EE90"
COLOR_WORKING = "#FFFF99"
COLOR_DONE = "#ADD8E6"
COLOR_EXIT = "#F4CCCC"

# ============================================================
# STEP 3 - Define Global Variables
# ============================================================

# Store selected Excel file path.
selected_file = None

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

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

# ============================================================
# STEP 4 - Define Helper Functions
# ============================================================

def log(message):
    """
    Add a message to the log window.
    """

    # Add message to log box.
    log_box.insert(tk.END, message + "\n")

    # Scroll to bottom.
    log_box.see(tk.END)


def select_excel_file():
    """
    Let the user select an Excel file.
    """

    global selected_file
```

```
    # Open file selection dialog.
    file_path = filedialog.askopenfilename(
        title="Select Excel File",
        filetypes=[
            ("Excel Files", "*.xlsx *.xls"),
            ("All Files", "*.*")
        ]
    )

    # If user selected a file, save it.
    if file_path:

        selected_file = Path(file_path)

        file_label.config(text=str(selected_file))

        log("Selected file: " + str(selected_file))

        analyze_button.config(bg=COLOR_NEXT)


def analyze_excel_file():
    """
    Analyze the selected Excel file.
    """

    # Check whether a file was selected.
    if selected_file is None:

        messagebox.showerror("Error", "Please select an Excel file first.")
        return

    try:

        # Show working color.
        analyze_button.config(bg=COLOR_WORKING)

        log("Reading Excel file...")

        # Read Excel file.
        data = pd.read_excel(selected_file)

        # Calculate summary.
        row_count = len(data)
        column_count = len(data.columns)
        column_names = list(data.columns)
        missing_values = data.isna().sum()
        duplicate_count = data.duplicated().sum()

        # Build report.
        report_lines = []
        report_lines.append("Excel Analysis Report")
        report_lines.append("====================")
```

```
        report_lines.append("")
        report_lines.append(f"File: {selected_file}")
        report_lines.append(f"Rows: {row_count}")
        report_lines.append(f"Columns: {column_count}")
        report_lines.append("")
        report_lines.append("Column Names:")
        for column in column_names:
            report_lines.append(f"- {column}")
        report_lines.append("")
        report_lines.append("Missing Values:")
        report_lines.append(str(missing_values))
        report_lines.append("")
        report_lines.append(f"Duplicate Rows: {duplicate_count}")

        report_text = "\n".join(report_lines)

        # Display report in log.
        log("")
        log(report_text)

        # Save report.
        timestamp = datetime.now().strftime("%Y-%m-%d--%H-%M")
        report_file = results_folder / f"Excel-Analysis-Report-{timestamp}.txt"
        report_file.write_text(report_text, encoding="utf-8")

        log("")
        log("Report saved to: " + str(report_file))

        # Show done color.
        analyze_button.config(bg=COLOR_DONE)

    except Exception as e:

        # Show error.
        messagebox.showerror("Error", str(e))

        log("ERROR: " + str(e))

        analyze_button.config(bg=COLOR_NEXT)


def clear_log():
    """
    Clear the log window.
    """

    log_box.delete("1.0", tk.END)

# ============================================================
# STEP 5 - Create GUI Window
# ============================================================

# Create main window.
```

```
window = tk.Tk()

# Set window title.
window.title("Excel Analyzer GUI")

# Set window size.
window.geometry("1000x700")

# ============================================================
# STEP 6 - Create GUI Widgets
# ============================================================

# Create title label.
title_label = tk.Label(
    window,
    text="Excel Analyzer GUI",
    font=("Arial", 18, "bold")
)
title_label.pack(pady=10)

# Create select file button.
select_button = tk.Button(
    window,
    text="1) Select Excel File",
    bg=COLOR_NEXT,
    width=25,
    command=select_excel_file
)
select_button.pack(pady=5)

# Create file path label.
file_label = tk.Label(
    window,
    text="No file selected",
    wraplength=900
)
file_label.pack(pady=5)

# Create analyze button.
analyze_button = tk.Button(
    window,
    text="2) Analyze Excel File",
    bg="#D3D3D3",
    width=25,
    command=analyze_excel_file
)
analyze_button.pack(pady=5)

# Create clear button.
clear_button = tk.Button(
    window,
    text="Clear Log",
    width=25,
```

```
    command=clear_log
)
clear_button.pack(pady=5)

# Create exit button.
exit_button = tk.Button(
    window,
    text="Exit",
    bg=COLOR_EXIT,
    width=25,
    command=window.destroy
)
exit_button.pack(pady=5)

# Create log window.
log_box = scrolledtext.ScrolledText(
    window,
    width=110,
    height=25,
    font=("Courier New", 10)
)
log_box.pack(padx=10, pady=10, fill="both", expand=True)

# ============================================================
# STEP 7 - Start GUI
# ============================================================

# Start the tkinter event loop.
window.mainloop()
```

Improve the Script

Ask ChatGPT:

```
Please improve the Excel Analyzer GUI.

New requirements:
1. Save an Excel report workbook instead of only a text report.
2. Add worksheets named Summary, Missing Values, and Data Preview.
3. Add a progress label.
4. Add a Save Log button.
5. Add a status bar at the bottom.
6. Keep the program header and step comments.
```

Exercise

Write a ChatGPT script for a GUI that selects a CSV file and creates a CSV analysis report.

What You Learned

You learned how to ask ChatGPT to create a practical Excel analyzer GUI.

Chapter 15 - Practical Project: Student Grade Analyzer

Objective

Create a complete Excel-based project that reads student grades and creates a grade report.

The program should:

- Read an Excel file
- Calculate each student's average
- Identify pass or fail
- Calculate class average
- Find highest average
- Find lowest average
- Save a report workbook

Input File

Create:

```
A-Data/Student-Grades.xlsx
```

Example columns:

```
Student_ID
Student_Name
Math
Science
English
History
```

Python File

Create:

```
B-Engine/Student-Grade-Analyzer.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:
Student-Grade-Analyzer.py

Program purpose:
Read A-Data/Student-Grades.xlsx and create a student grade analysis report.

Requirements:
1. Use pandas.
2. Read A-Data/Student-Grades.xlsx.
3. Assume columns:
   - Student_ID
   - Student_Name
   - Math
   - Science
   - English
   - History
4. Calculate each student's average grade.
5. Add a column named Status:
   - Pass if average >= 60
   - Fail if average < 60
6. Calculate:
   - class average
   - highest average
   - lowest average
   - number of passing students
   - number of failing students
7. Save report workbook to C-Results/Student-Grade-Report.xlsx.
8. Workbook should include:
   - Student Results worksheet
   - Class Summary worksheet
9. Print summary to screen.

Coding Standards:
1. Add a detailed program header.
2. Include Program Name, Purpose, User ChatGPT Script, Expected Output, and Version.
3. Divide code into clearly labeled steps.
4. Add comments explaining every important line.
5. Make code beginner friendly.

Expected output:
A student grade analysis report is saved to C-Results.
```

ChatGPT Generated Python Code

```
# ============================================================
# Program Name:
# Student-Grade-Analyzer.py
#
```

```
# Purpose:
# Read Student-Grades.xlsx and create a student grade analysis report.
#
# User ChatGPT Script:
# Read A-Data/Student-Grades.xlsx, calculate student averages,
# assign Pass/Fail status, create class summary, and save report.
#
# Expected Output:
# A student grade analysis report is saved to C-Results.
#
# Version:
# 1.0
# ============================================================

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

from pathlib import Path
import pandas as pd

# ============================================================
# STEP 2 - Define Files and Folders
# ============================================================

input_file = Path("A-Data") / "Student-Grades.xlsx"
results_folder = Path("C-Results")
results_folder.mkdir(exist_ok=True)
output_file = results_folder / "Student-Grade-Report.xlsx"

# ============================================================
# STEP 3 - Read Student Grades
# ============================================================

grades = pd.read_excel(input_file)

# ============================================================
# STEP 4 - Define Grade Columns
# ============================================================

grade_columns = [
    "Math",
    "Science",
    "English",
    "History"
]

# ============================================================
# STEP 5 - Calculate Student Averages
# ============================================================

grades["Average"] = grades[grade_columns].mean(axis=1)
```

```
# ============================================================
# STEP 6 - Assign Pass or Fail Status
# ============================================================

grades["Status"] = grades["Average"].apply(
    lambda average: "Pass" if average >= 60 else "Fail"
)

# ============================================================
# STEP 7 - Create Class Summary
# ============================================================

class_summary = pd.DataFrame({
    "Metric": [
        "Class Average",
        "Highest Average",
        "Lowest Average",
        "Passing Students",
        "Failing Students"
    ],
    "Value": [
        grades["Average"].mean(),
        grades["Average"].max(),
        grades["Average"].min(),
        (grades["Status"] == "Pass").sum(),
        (grades["Status"] == "Fail").sum()
    ]
})

# ============================================================
# STEP 8 - Save Excel Report
# ============================================================

with pd.ExcelWriter(output_file, engine="openpyxl") as writer:

    grades.to_excel(
        writer,
        sheet_name="Student Results",
        index=False
    )

    class_summary.to_excel(
        writer,
        sheet_name="Class Summary",
        index=False
    )

# ============================================================
# STEP 9 - Display Summary
# ============================================================

print("Student Grade Report")
print("--------------------")
```

```
print("Students:", len(grades))
print("Class Average:", round(grades["Average"].mean(), 2))
print("Passing Students:", (grades["Status"] == "Pass").sum())
print("Failing Students:", (grades["Status"] == "Fail").sum())
print("Report saved to:", output_file)

# ============================================================
# STEP 10 - End Program
# ============================================================

print("Program finished.")
```

Improve the Script

Ask ChatGPT:

```
Please improve the student grade analyzer.

New requirements:
1. Add letter grades:
   A >= 90
   B >= 80
   C >= 70
   D >= 60
   F < 60
2. Add a worksheet named Letter Grade Summary.
3. Add a bar chart of average grades.
4. Auto-adjust Excel column widths.
5. Add timestamp to output filename.
6. Keep the program header and step comments.
```

Exercise

Write a ChatGPT script for an employee performance analyzer.

The Excel file should include:

- Employee name
- Quality score
- Productivity score
- Attendance score
- Average score
- Performance category

What You Learned

You learned how to create a complete practical Excel analysis project using ChatGPT scripts.

Chapter 16 - Debugging Excel Programs with ChatGPT

Objective

Learn how to ask ChatGPT to debug Excel-related Python errors.

Excel programs often fail because of:

- Missing files
- Wrong sheet names
- Missing packages
- Wrong column names
- Open Excel files
- Permission problems
- Bad data types

Common Excel Errors

FileNotFoundError

This usually means Python cannot find the file.

Possible causes:

- File name is wrong
- File is in the wrong folder
- Program is running from the wrong folder

ModuleNotFoundError

This usually means a required package is not installed.

Example:

```
ModuleNotFoundError: No module named 'openpyxl'
```

Fix:

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

On Mac:

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

KeyError

This usually means a column name is missing or misspelled.

Example:

```
KeyError: 'Salary'
```

Possible cause:

- The column is named `salary`
- The column has extra spaces
- The column does not exist

Debugging Script

Use this ChatGPT script when your Excel program has an error:

```
My Python Excel 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 or Excel column names 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]
```

Script to Improve Error Handling

Ask ChatGPT:

```
Please improve my Excel program.

New requirements:
1. Check if the input file exists before reading it.
2. Print the current working directory.
3. Print the expected input file path.
4. Check whether required columns exist.
5. Show a friendly message if columns are missing.
6. Keep the program header and step comments.
```

Exercise

Create one error on purpose:

- Rename the Excel file
- Change a column name
- Remove openpyxl
- Use the wrong sheet name

Then ask ChatGPT to help debug it.

What You Learned

You learned how to use ChatGPT to debug Excel automation programs.

Chapter 17 - Part 5 Review

What You Built in Part 5

Part 5 was divided into four sections.

Part 5A

You learned:

- CSV vs Excel
- Reading CSV files
- Writing CSV files
- Reading Excel files

Part 5B

You learned:

- Writing Excel files
- Selecting Excel files with a GUI
- Viewing Excel file information
- Creating Excel statistics

Part 5C

You learned:

- Data validation
- Data cleaning
- Combining Excel files
- Creating Excel reports

Part 5D

You learned:

- Photo inventory workbook
- Excel analyzer GUI
- Student grade analyzer
- Debugging Excel programs

Part 5 Skills Checklist

By the end of Part 5, you should be able to:

☐ Write ChatGPT scripts for CSV programs

- ☐ Write ChatGPT scripts for Excel programs
- ☐ Read CSV files using pandas
- ☐ Write CSV files using pandas
- ☐ Read Excel files
- ☐ Write Excel files
- ☐ Validate Excel data
- ☐ Clean Excel data
- ☐ Combine multiple Excel files
- ☐ Generate Excel reports
- ☐ Build Excel GUI tools
- ☐ Create practical Excel projects
- ☐ Debug Excel programs using ChatGPT

Important ChatGPT Script Pattern

For Excel programs, always tell ChatGPT:

```
1. What file to read.
2. Where the file is located.
3. What columns exist.
4. What calculations are needed.
5. Where to save the result.
6. What error handling is needed.
7. How the code should be commented.
```

Summary

In Part 5, you learned how to use ChatGPT to develop Python programs for CSV and Excel automation.

This is one of the most practical skills in the course because many users already work with spreadsheets every day.

You learned the full development workflow:

```
Spreadsheet Problem
        ↓
ChatGPT Script
        ↓
Generated Python Program
        ↓
Run in VS Code
        ↓
Check Output
        ↓
Improve Script
        ↓
Improve Program
```

Looking Ahead

In Part 6, you will learn how to use ChatGPT to generate Python programs for image processing and photo automation.

You will build programs that can:

- Read image folders
- Resize images
- Convert formats
- Create contact sheets
- Add page numbers
- Prepare images for printing
- Organize photo collections