

CSI 3680 – Script Programming

Final Project Requirement

1. Overview

The final project is a comprehensive exercise designed to assess your ability to apply Python programming skills to a practical problem. You are expected to design, implement, and present a Python-based application that meets all requirements listed below.

This project will be completed in **teams of 2–3 students**. Each team will submit one project and give a short presentation during the final session.

2. General Requirements

Your project must satisfy **all** of the following criteria:

1. **Python-based:** The entire project must be implemented in Python.
2. **Practical Relevance:** The topic should have clear real-world or potential practical usage.
3. **Functions:** Define and use at least **two functions** in meaningful ways (e.g., computation, data processing, feature extraction, etc.).
4. **Class:** Define at least **one class** that contains multiple attributes and methods. The class should represent an object or concept relevant to your project.
5. **Third-Party Package:** Import and utilize at least **one third-party library/package** (e.g., `requests`, `pandas`, `matplotlib`, `numpy`, `opencv`, `beautifulsoup4`, etc.) to support your implementation.
6. **File I/O:** Perform at least one file read or write operation (text, CSV, JSON, image, etc.).
7. **Exception Handling:** Use exception handling to ensure each file is opened and accessed without errors.
8. **Visualization:** Present your results in a visual form—such as a plot, image, GUI, or web page. Console-only output is not acceptable.

3. Deliverables

- Python source code files (`.py` or `.ipynb`)
- Input and/or output data files (if applicable)

- A short README file describing:
 - The purpose of your project
 - Instructions on how to run the code
 - Dependencies (required packages)
 - Explanation of key functions and classes
 - Team members and their major contributions

4. Final Presentation

Each team will deliver a short in-class presentation on their project during the final week:

- **Date:** December 11, 2025
- **Format:** 7–10 minutes per team (including Q&A)
- **Content:**
 - Project goal and motivation
 - System design and key functions/classes
 - Demonstration of the program (screenshots, demo video, or live run)
 - Discussion of results and visualization
 - Lessons learned and team contribution
- Each member should have a speaking role.

5. Suggested Project Types

5.1 Data Crawler

Goal: Collect, store, and analyze data from multiple online sources.

Example Objects: city weather data, movie information, traffic reports, or IT job postings.

Requirements:

- Crawl data from multiple websites (not just one simple source).
- Save data into structured files (e.g., CSV, JSON).
- Implement subsequent programs to process and visualize the data.

Notes:

- The crawler source code must not be overly simple.
- Avoid trivial topics such as “Top 100 IMDb Movies.”
- Ensure the entire data-processing pipeline is demonstrated (crawl → store → process → visualize).

5.2 Data Processing & Analysis

Goal: Process and analyze structured or unstructured data to extract meaningful insights.

Example Data: stock prices, sales data, text, images, or audio files.

Recommended Data Sources:

- Kaggle: <https://www.kaggle.com/>
- Papers with Code: <https://paperswithcode.com/datasets>

Suggested Tasks:

- Clean and preprocess real-world data.
- Analyze or model data using machine learning or simple deep learning.
- Visualize the results using graphs, charts, or images.

Examples:

- Predict future stock prices based on historical data.
- Analyze sentiment in product reviews.
- Visualize and classify images using a simple CNN.

5.3 Practical Tool or Software

Goal: Develop a Python-based tool that serves a practical purpose.

Example Ideas:

- **Encryption/Decryption Tool:** Implement RSA (1024 bits) to encrypt and decrypt files.

- **Compression Tool:** Implement a basic data compression and decompression algorithm.
- **Music Player:** Display lyrics or audio spectrum while playing music.
- **Mini Photoshop:** Implement image filters, transformations, or letter outlining.

5.4 Game Development

Goal: Design and implement an interactive game with meaningful logic and features.

Example Ideas:

- **Sudoku Puzzle:** Automatically generate puzzles, provide hints, and compute solutions (GUI recommended).
- **Board Games:** Chess, Go, or Checkers with human-machine interaction, move history, and time control.
- **Other Games:** Small character-based games or GUI games with logical rules and scoring systems.

Notes:

- Simple remakes of classic games (e.g., Snake, Flappy Bird) are not acceptable.
- If web data is involved, the full chain (crawl-store-use) must be implemented.

6. Evaluation Criteria

Your final grade will be based on both the implementation and presentation components:

1. **Code correctness and completeness** (25%)
2. **Creativity and practical value** (20%)
3. **Code structure, readability, and documentation** (15%)
4. **Proper use of functions, classes, libraries, and file I/O** (15%)
5. **Visualization and user interaction** (10%)
6. **Final presentation and teamwork contribution** (15%)

Detailed Rubric:

Criterion	Excellent (A)	Good (B)	Fair (C)	Poor (D/F)
Code Correctness	Runs flawlessly with full features	Minor bugs or incomplete edge cases	Partially functional, frequent issues	Fails to run or compile
Creativity / Practical Value	Original, meaningful, real-world value	Interesting but moderately creative	Basic or repetitive idea	Trivial or copied idea
Structure & Documentation	Well-organized, clear comments	Some comments and structure	Poor organization or unclear naming	No comments or unclear structure
Functions / Classes / Libraries	Well-designed, reusable, correctly applied	Meets all requirements with small issues	Meets minimum requirement only	Missing or misused elements
Visualization	Informative and visually appealing	Functional but simple	Minimal or unclear visualization	None or console-only output
Presentation / Teamwork	Clear, engaging, balanced team effort	Organized with minor delivery issues	Uneven participation or unclear delivery	Unprepared or no team coordination

7. Submission Instructions

- Submit your complete project folder as a compressed file (.zip or .tar.gz).
- Include your Python source files, data files, and README.
- Make sure your project runs smoothly on Google Colab or a standard Python 3.9+ environment.
- Clearly list team members and their major contributions in the README.

8. Important Reminder

Academic integrity is expected at all times. You may refer to online examples or tutorials, but you must not copy code directly. All work must be your own. Any violation will result in a zero grade.

“Think creatively. Code responsibly. Build something meaningful.”