

Fundamentals of Computer Programming

Final Project/Report Assignment (Python)

Research Institute of Economics and Management

Dr. L Zhang

October 28, 2026

Contents

1	Introduction	1
2	Project Requirements	2
2.1	General Requirements	2
2.2	Technical Specifications	2
2.3	Self-Study Module: Using Classes	3
2.4	Academic Integrity: Use of Generative AI	3
3	Deliverables	3
3.1	Python Project Code Archive	3
3.2	Project Report	4
4	Submission Guidelines	4
5	Grading Criteria	5
6	Python Sample Project	5

1 Introduction

This document is the final assignment for the "Fundamentals of Computer Programming" course. The main purpose of this assignment is to assess your ability to apply the knowledge learned in class to solve a practical problem. You will work in a small group to design, implement, and document a complete, console-based Python application that addresses a real-world business or logic-based scenario.

The primary assessment focus of this project is on your proficiency in core course topics—specifically **functions**, **if statements**, **loops**, **collections** (e.g., lists and dictionaries), and **modular programming**—as well as adhering to good programming practices.

2 Project Requirements

2.1 General Requirements

- Form a group of **no more than 4 students** (1 to 4 students). Each group must choose a unique team name.
- Design and implement a complete Python program that addresses a practical scenario. Examples include a simple inventory management system, a student record analysis tool, a personal finance tracker, or a small text-based game. **Note:** These examples are for illustration only; you are **not allowed** to use the scenarios listed above for your project.
- The program must be a standard **console application**. The following are **strictly prohibited**:
 1. **GUI programming** (e.g., Tkinter, PyQt, etc.).
 2. **Jupyter Notebook** (`.ipynb` files); all code must be written in standard `.py` files.
 3. **Network programming** (e.g., sockets, HTTP requests, web scraping, or web frameworks).
- The primary purpose of this assignment is to evaluate your practical application of classroom concepts, specifically **functions**, **if statements**, **loops**, **collections**, and **modularization**.
- Your code must be well-commented, especially for complex business logic.

2.2 Technical Specifications

Your Python project must demonstrate proficiency in all the following areas:

1. Program Structure (Key Assessment Focus):

- The program must be organized into **multiple modules** (i.e., multiple standard `.py` script files; **Jupyter Notebook `.ipynb` files are not permitted**).
- You must make effective use of `import` statements to share functions and data between your modules. For example, you might have a `main.py` for the main program flow, a `data_source.py` for storing data, and a `logic.py` for business calculations.

2. Data Management:

- Supporting data for your application (e.g., initial inventory, user lists) may be stored within a dedicated Python module (e.g., `data_source.py`) using Python data structures such as **lists** and **dictionaries**, or organized using external `.txt` and `.csv` files.
- If you choose to use `.txt` or `.csv` files, you must **self-study** how to:
 - (a) Read data from `.txt` and `.csv` files.
 - (b) Store (write) data to `.txt` and `.csv` files.
- External databases (e.g., SQLite, MySQL) are **not permitted**.

3. **Functions:** The program logic must be well-encapsulated in functions with clear purposes, parameters, and return values.
4. **Data Structures:** Effective use of **lists** and **dictionaries** to manage and manipulate collections of data during program execution.
5. **Control Flow:** Correct and logical implementation of conditional statements (`if/elif/else`) and loops (`for/while`).

2.3 Self-Study Module: Using Classes

Only the basics of `class` (Object-Oriented Programming, OOP) were introduced during lectures. You are encouraged to study this topic in greater depth on your own.

- You **may** use classes in your project if you feel they are appropriate, for example, to create data objects and encapsulate related operations.
- The use of Object-Oriented Programming is **not required**; however, reasonable and effective use of OOP **will earn bonus points**. The primary evaluation will still focus on the modular structure of your program as described in Section 2.2.

2.4 Academic Integrity: Use of Generative AI

You may use Generative AI tools (such as ChatGPT, DeepSeek, GitHub Copilot, etc.) **only to discuss problems and clarify concepts**. However, **using AI tools to generate code or write the project report is strictly prohibited**.

The goal of this assignment is to assess your personal understanding and problem-solving skills. **Submitting work (either code or report) generated by AI is a violation of academic integrity and will result in a score of zero for the project and may lead to further disciplinary action.** All submissions may be subject to review for originality.

3 Deliverables

Each group must submit two items: an archive of the Python project files and a detailed project report in PDF.

3.1 Python Project Code Archive

- Your project should be contained within a single folder.
- All source code must be written in standard `.py` files. **The use of Jupyter Notebook (`.ipynb`) is strictly prohibited and will not be accepted.**
- The folder should be named in the format: `Python2026_TeamName_ProjectName`. Please do not use spaces or special characters in the names.
- This folder must be compressed into a single `.zip` file for submission.
- **Note:** Failure to follow the required directory structure or `.zip` archive format will result in point deductions.

3.2 Project Report

Use the Word template for your report. Add the cover provided in the template file to your report (missing the cover page will result in a 10-point deduction from the total score). The report must be compiled and submitted as a PDF document (submitting a Word file rather than a PDF will result in a 10-point deduction).

It should be written in **English** and be clear, concise, and professional. The report must include (but is not limited to) the following sections:

1. Scenario Description:

- What is the problem you are solving?
- What are the key features and functionalities of your program?
- A detailed description of each module and function designed to implement these features.

2. Program Design:

- A description of the overall program structure, including a list of all .py files (modules) and the purpose of each.
- Diagrams drawn based on your own understanding of the system—such as system architecture diagrams, system module diagrams, and flowcharts—along with corresponding descriptions of the main business logic and program flows.

3. Member Contributions:

- A clear statement detailing the percentage of work contributed by each group member. This is a crucial factor for individual grading.

The report must be submitted as a **PDF document**. A cover page template will be provided in a separate file.

4 Submission Guidelines

- **Deadline:** **November 15, 2026, 23:59 (11:59 PM) Peking Time.**

- **Files to Prepare:**

1. Project Code ZIP File: `Python2026_TeamName_ProjectName.zip`
2. Project Report PDF File: `Python2026_TeamName_ProjectName.pdf`

- **Submission Method:**

- The group leader must send an email with the two files attached.
- **To:** `nguzlp01@gmail.com`
- **CC:** `ms_zlp@hotmail.com`
- **Email Subject:** `Python2026 TeamName Final Project`
- **Email Body:** Must include the Team Leader's name, student ID, and contact information (including a **phone number** for emergency contact).

5 Grading Criteria

Projects will be evaluated based on the following criteria:

- **Program Functionality (40%):** Does the program run correctly and implement the described features?
- **Code Quality and Structure (40%):** Is the code well-organized into modules and functions? Is it properly formatted, readable, well-commented, and compliant with standard Python conventions (PEP 8)?
- **Project Report (20%):** Is the report well-written, clear, comprehensive, and properly formatted?
- Individual scores will be adjusted based on the declared member contributions.

6 Python Sample Project

A Python sample project is provided separately as a `.zip` archive. This project contains a simple but complete example demonstrating the required modular structure. It shows how to separate data, business logic, and the main program execution into different `.py` files. You are strongly encouraged to study its structure before starting your own project.