

**Faculty of Arts and Science**

The Department of Computer Science

# CIS1702

# Programming 1

Level 4

Coursework 2 (Group Project)

2025/2026

**Module Leader:** **Saad Saihi**

**Email: saihis@edgehill.ac.uk**

***Administrators:***

🕿 01695 65 7603

**Coursework**

* **Coursework Two: Group Project**
* **Weighting: 60% of the total module mark**
* **Submission Deadline: 6 January 2026 at 23:59**

### 1.1 LEARNING OUTCOMES

* **LO1.** Demonstrate understanding and competence of the basic concepts in procedural programming.
* **LO2.** Reflect on the approaches and techniques selected and used by the group to accomplish their outcomes.
* **LO3.** Analyse a problem, and produce a computer program as a solution using a process of design, implementation and testing.
* **LO4.** Demonstrate an ability to use a simple development environment effectively to produce viable program code.

**Introduction**

This assignment is the final major part of the formal assessment of CIS1702 and is compulsory. This group project is designed to test your ability to work as part of a team to construct a more complex application than would be possible individually. It synthesises the design, implementation, testing, and collaboration skills developed throughout the module.

### THE TASK

Working in a team of 4-6 students, you will design, develop, test, and document a complete Python application based on a project brief provided at the start of the module. To facilitate effective planning and collaboration, teams will be formed by Week 6. The project will require you to combine your individual coding skills and manage your work collaboratively using a shared GitHub repository. The goal is to produce a single, functional, and well-documented software solution.

**Project Briefs**

### TECHNICAL SPECIFICATIONS

### Language: Your project must be written in Python 3.8 or newer.

### Libraries: You are restricted to the Python Standard Library. You may not use any third-party libraries (e.g., packages installed via PyPI) unless explicitly approved by the module leader. This ensures a consistent environment for marking.

### Your team must choose one of the following three projects to complete.

#### Project Brief 1: Command-Line Inventory Management System

* **Project Goal:** Create a command-line application to help a small business owner track product inventory. The system must allow the user to add, view, update, and remove stock items, and save the data between sessions.
* **Key Functional Requirements:**
  + The application must store inventory data in a file (e.g., inventory.json or inventory.csv).
  + On startup, the program must load existing data from the file.
  + The user must be presented with a clear menu to perform actions (e.g., Add Item, View Stock, Update Item, Search, Save & Exit).
  + **Add Item:** Allow the user to add a new product with a unique ID, name, price, and quantity.
  + **View Stock:** Display a formatted table of all products in the inventory.
  + **Update Item:** Allow the user to find an item by its ID and update its name, price, or quantity.
  + **Search:** Allow users to search for an item by name and display its details.
  + The program must handle errors gracefully (e.g., invalid input, item not found).
* **Possible Extensions:**
  + Generate a low-stock report for items with a quantity below a certain threshold.
  + Add functionality to track sales, reducing the quantity of an item when sold.
  + Implement a more advanced search feature (e.g., by price range).
* **Marking Focus Areas:** Robust file handling for data persistence, effective use of data structures (lists of dictionaries), clear user interface and input validation, and comprehensive testing of all core CRUD (Create, Read, Update, Delete) operations.

#### Project Brief 2: API Data Analyser & Reporter

* **Project Goal:** Develop a command-line tool that fetches data from a public API, performs a simple analysis, and generates a report file.
* **Key Functional Requirements:**
  + The application must interact with a free, public JSON API (e.g., a weather API, a public data API like REST Countries, or a fun API like the Pokémon API).
  + The user should be able to input a query (e.g., a city name, a country, a Pokémon name).
  + The program must make a request to the API and parse the returned JSON data.
  + **Data Extraction:** Extract at least four key pieces of information from the JSON response.
  + **Simple Analysis:** Perform a simple calculation or data transformation (e.g., convert temperature from Kelvin to Celsius, calculate population density, list a Pokémon's abilities).
  + **Display Output:** Print a clean, formatted summary of the extracted and analysed data to the console.
  + **Generate Report:** Save the formatted summary to a text file (e.g., report.txt). Each new report should be appended to the file.
  + The program must include robust error handling for network issues, invalid user input, or missing data in the API response (using try-except blocks).
* **Possible Extensions:**
  + Allow the user to make multiple requests in a loop until they choose to exit.
  + Compare data from two different API calls (e.g., compare the populations of two countries).
  + Save report data in a structured format like CSV.
* **Marking Focus Areas:** Correct use of libraries to fetch web data, robust JSON parsing, comprehensive error handling (especially for external services), modular code (functions for fetching, parsing, and reporting), and clear output formatting.

#### Project Brief 3: Text-Based Adventure Game Engine

* **Project Goal:** Create a reusable "engine" for a text-based adventure game. Instead of hard-coding the story, the engine should load the game's map, rooms, and interactions from a structured data file (e.g., game\_map.json).
* **Key Functional Requirements:**
  + On startup, the program must load the entire game world from a JSON file. This file will define rooms, descriptions, items, and connections between rooms (e.g., "north" from the "Hall" leads to the "Kitchen").
  + The player must be able to navigate between rooms using commands like go north, go east, etc.
  + The player must have an inventory and be able to get item and drop item.
  + The program must parse user input to understand commands and arguments (e.g., verb: "go", noun: "north").
  + The game state (player's current location, inventory) must be tracked accurately.
  + The program must handle invalid commands and impossible actions gracefully (e.g., "You can't go that way.").
  + Include a "help" command that lists available actions.
  + The game must have a clear win or lose condition defined within the data file.
* **Possible Extensions:**
  + Implement "locked" doors that require a specific item from the inventory to open.
  + Add characters (NPCs) that the player can interact with using a talk to command.
  + Allow the player to save and load their game progress to a separate file.
* **Marking Focus Areas:** Strong separation of game logic ("engine") from game data ("JSON file"), effective use of complex data structures (nested dictionaries/lists), a robust input parser, and logical state management.

**TEAM FORMATION AND MANAGEMENT**

Teams must be formed by **Week 6**. The process is as follows:

1. **Form Your Team:** Organise yourselves into groups of 4-6 students.
2. **Choose a Project:** As a team, decide which of the three project briefs you will undertake.
3. **Register Your Team:** One member of the team must post a single message in the "Team Formation" discussion tab on the VLE with the following information:
   * **Group Name:** (e.g., The Code Crusaders)
   * **Chosen Project:** (e.g., Project Brief 1: Inventory Management)
   * **Members:** (List each member's full name and student ID).

**Recommendations for Team Success:**

* **Assign Roles:** While not mandatory, consider assigning roles like a team lead (to coordinate meetings), a Git manager (to handle merges), and a documentation lead.
* **Communicate Regularly:** Set up a group chat and schedule short, regular meetings to sync up on progress and resolve blockers.
* **Resolve Conflicts Early:** Disagreements are normal. Address them constructively and respectfully. If a conflict cannot be resolved, or if a team member is not contributing, please notify your lab tutor or the module leader as early as possible.

**What will be assessed?**

Your final mark for this coursework is based on an evaluation of both your team's final product and your individual contribution to the project. The key areas are:

* **The quality and functionality of your team's final Python application.** This includes whether it meets the project brief, how robust and free of bugs it is, and the quality of the code itself (e.g., is it well-structured and commented?).
* **The quality and completeness of your group report.** This includes the clarity of your design documentation, the thoroughness of your test plan, and the insightfulness of your reflective analysis on the teamwork process.
* **Your personal contribution to the project.** This is assessed individually based on evidence from the GitHub repository's commit history, the breakdown of work in your group report, and the feedback provided by your teammates in the confidential peer review.

**What you should submit**

There are three components to your final submission, all submitted via the VLE (Blackboard):

* **Source Code Submission:** You must submit both of the following:
  1. GitHub Repository URL: A link to your team's shared GitHub repository.
  2. Source Code Archive: A single **.zip** file containing the complete, commented source code for your application. This ensures a permanent, time-stamped record of your work at the time of the deadline.
* **Group Report:** A single PDF document that includes:
  + Design documentation (e.g., flowcharts, pseudocode).
  + A comprehensive test plan with evidence of testing.
  + A clear breakdown and description of each team member's specific contributions.
  + A reflective analysis of your group's teamwork, processes, and technical decisions.
* **Peer Review Form:** Each student must individually and confidentially submit the peer review form provided on the VLE. This is a mandatory component used to help determine individual contribution marks.

**Report Guidelines**

Your group report should be a professional document that details your project's lifecycle.

* **Length:** Approximately 6-10 pages (excluding title page and appendices).
* **Format:** Standard academic format (e.g., 12-point font, 1.5 line spacing).
* **Structure:** The report must include the following sections:  
  1. **Introduction:** Briefly introduce your chosen project, your team, and the report's structure.
  2. **System Design:** Explain your design process. Include design artifacts like  
      **flowcharts or pseudocode** that illustrate the main logic of your application. Describe your data structures and file formats.
  3. **Implementation Summary:** Discuss key implementation details and technical decisions. Highlight any significant challenges you overcame. Do not just paste code here.
  4. **Testing:** Provide a **comprehensive test plan** with evidence of testing (e.g., a table of test cases with inputs, expected outputs, and actual outputs/screenshots).
  5. **Breakdown of Contributions:** Include a clear table or list detailing **each team member's specific contributions** to the project (e.g., "Jane Doe: Implemented file handling and the search function").
  6. **Reflection:** A **reflective analysis** of your group's teamwork, processes, and technical decisions. What went well? What would you do differently next time? How did you manage collaboration?
  7. **Conclusion:** Summarise the project and your achievements.

**Peer Review Form**

This form must be completed and submitted individually. Be honest and constructive in your feedback.

* **Your Name:** \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_
* **Your Student ID:** \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_
* **Your Group Name:** \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Instructions:** For each of your teammates (do not rate yourself), provide a score from 1 to 5 for each category based on the descriptions below. Add specific comments to justify your ratings.

**Rating Scale:**

* **5 - Outstanding:** Consistently exceeded expectations and took a leading role.
* **4 - Very Good:** Consistently fulfilled all duties to a high standard.
* **3 - Good:** Fulfilled all assigned duties reliably.
* **2 - Inconsistent:** Contributed, but inconsistently or required significant prompting.
* **1 - Poor:** Contributed very little or was actively disruptive.

**Teammate 1 Name:** \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

* **Contribution & Quality of Work (1-5):** \_\_\_\_\_\_
* **Reliability & Meeting Deadlines (1-5):** \_\_\_\_\_\_
* **Communication & Teamwork (1-5):** \_\_\_\_\_\_
* **Comments (Required):** \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Teammate 2 Name:** \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

* **Contribution & Quality of Work (1-5):** \_\_\_\_\_\_
* **Reliability & Meeting Deadlines (1-5):** \_\_\_\_\_\_
* **Communication & Teamwork (1-5):** \_\_\_\_\_\_
* **Comments (Required):** \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

***(...repeat for all other teammates)***

**Extensions and Malpractice**

### LATE SUBMISSIONS

### The standard university policy for late submissions applies to individual components of the assessment (such as the peer review form). However, for the group submission (the GitHub link and Group Report), the deadline is final. Late submissions for group work will not be accepted without prior approval through a formal extension, as a delay from one member impacts the entire team. Early submission is therefore strongly recommended.

### EXTENSIONS

If you are unable to submit by the stated deadline, you should follow the extension application guidelines below. Only in extreme cases will extensions be granted for coursework.

All applications for extensions should be submitted to the Department Administrator. Any application for an extension of a deadline should normally be made at least twenty-four hours before the due deadline for the assessment. In exceptional circumstances outside of this time scale students are permitted to submit an extension form without supporting evidence. The supporting evidence must be submitted within 7 working days or the extension request will not be approved. The timing of student feedback against published deadlines will be affected if an extension is granted.

### PERSONAL CIRCUMSTANCES

If you are affected by more serious, long-term problems that are likely to prevent you from completing work with a short-term extension, then you may apply for Personal Circumstances using the online form.

### Academic Malpractice

It is a serious academic offence to use or attempt to use unfair means to enhance your performance or influence the standard of award obtained. 'Unfair' includes all forms of cheating, including plagiarism, collusion and impersonation. When researching and writing-up your coursework you must ensure that you fully reference any elements within it that are not written or conceived by yourself. If you fail to attribute a source for the work of other people and present it as your own, you are stealing their work. Academic malpractice of this kind ('plagiarism') is seen as the most serious offence in academia.

The department of Computer science, like many departments at Edge Hill University, use the Harvard referencing system.

### AVOIDING MALPRACTICE WHEN USING ELECTRONIC RESEARCH MECHANISMS

You will be encouraged to use a range of electronic and online resources such as electronic journals, online database and the Internet. However, students need to be cautious when using these research tools. You should ensure that you search within the relevant parameters of investigation as framed by the recommended reading and research materials indicated in the module handbook. Remember, it is as easy for the department to uncover any information taken from the internet and attempted to be passed off as your own work, as it is for you to find it!

**Marking Criteria**

The following marking criteria table is used to determine the final mark for CW1 based on cumulative performance across all three in-class tests:

| **Criteria** | **Fail**  **(0-29%)** | **Narrow Fail**  **(30-39%)** | **Pass**  **(40-49%)** | **Good**  **(50-59%)** | **Very Good**  **(60-69%)** | **Excellent**  **(70-84%)** | **Outstanding**  **(85+%)** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **LO1. Demonstrate understanding and competence of the basic concepts in procedural programming. (25%)** | Answers <30% of syntax, control flow, and function questions correctly; shows little or no conceptual understanding. | Answers 30–39% correctly; significant misconceptions in loops, types, or control structures. | Answers 40–49% correctly; demonstrates basic recall of syntax and use of variables, but limited functional application. | Answers 50–59% of relevant questions with minor syntax/logical errors; applies functions or loops with basic correctness. | Answers 60–69% correctly; consistently uses core concepts like loops/functions and shows some abstraction. | Answers 70–84% correctly; integrates multiple concepts (e.g., loops in functions, conditional logic). | Near-perfect use of syntax and constructs; applies them fluently across all tasks. |
| **LO2. Reflect on the approaches and techniques selected and used by the group to accomplish their outcomes. (20%)** | The reflective analysis in the report is missing or entirely superficial, with no discussion of the group process. | The reflection describes what the group did but shows little to no analysis of the techniques used, challenges faced, or teamwork dynamics. | The reflection provides a basic description of the group's work but lacks critical insight. It may state challenges but offers no analysis of how they were overcome. | The report provides a good analysis of the team's process, identifying some strengths and weaknesses of the chosen approach. | The report provides a clear and insightful reflection on the group's techniques, decisions, and overall process, supported by examples. | The report provides a critical and deeply insightful reflection, expertly analysing the teamwork and technical decisions made throughout the project. | The report provides an exemplary reflection that not only analyses the group's process but also insightfully connects it to wider development methodologies. |
| **LO3. Analyse a problem, and produce a computer program as a solution using a process of design, implementation and testing. (30%)** | Fails to identify problem logic; incorrect or missing program structure. | Attempts problem-solving but fails in execution; misinterprets requirements. | Can solve basic logic tasks (1–2 steps) but struggles with multi-step analysis or modular design. | Successfully implements most logic tasks with minor inefficiencies or errors in reasoning. | Applies logical structures well; uses functions and modular approaches to solve moderate problems. | Demonstrates design thinking; breaks down problems logically, uses reusable code. | Synthesizes multiple concepts into robust, efficient solutions; excellent interpretation of requirements. |
| **LO4. Demonstrate an ability to use a simple development environment effectively to produce viable program code. (20%)** | No evidence of understanding environment (VS Code, Git, CLI). | Recognizes environment but misidentifies key components or commands. | Identifies tools correctly but uses few features appropriately. | Uses terminal, basic Git, and run/debug features with some accuracy. | Navigates VS Code efficiently; understands command-line use and Git basics. | Integrates environment features smoothly (e.g., Git + terminal + extensions). | Expert use of VS Code and Git; anticipates and solves environment-related issues independently. |
| **Transferable Skills - ability to communicate information/academic writing and language (5%)** | No sentence structure, many errors in spelling, grammar and vocabulary which severely affects the meaning and understanding of the discussion. | Unclear communication of ideas. Frequent significant errors in spelling, punctuation, and grammar, affecting the meaning and understanding of the discussion. | Clear communication of ideas but some significant errors in spelling, punctuation, and grammar.    Sentence structures are incorrect and affect the understanding of the discussion. | Written in generally correct English (UK), contains minor reoccurring spelling and grammatical errors.    Sentence structures are partially correct and contain some relevant vocabulary, aiding the understanding of the discussion. | Written in correct English (UK), contains minor errors that do not affect the meaning and understanding of the discussion and are not recurring.    Sentences structured correctly and contain appropriate and relevant vocabulary, which includes some technical terminology, used to support the communication and development of ideas. | Fluent writing style, written in correct English (UK). No abbreviations evident.    Effective and accurate use of a variety of appropriate vocabulary, incorporating detailed technical terminology.    A strong indication of a developing academic writing style, with each paragraph attempting to follow the [SEED](http://eshare.edgehill.ac.uk/7165/1/Paragraph%20Structure%20%28SEED%29.pdf) structure. | Fluent writing style, written in correct English (UK). No abbreviations evident.    Highly effective and accurate use of a wide variety of appropriate vocabulary, incorporating detailed technical terminology.    Adopts a professional and academic writing style and conventions, with each paragraph following the [SEED](http://eshare.edgehill.ac.uk/7165/1/Paragraph%20Structure%20%28SEED%29.pdf) structure. |