University of Greater Manchester

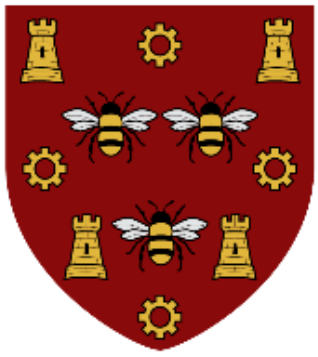
BSc Computing / Computer Science / Software Engineering / Artificial Intelligence / Cyber Security

COM4402 Programming

MODULE GUIDE FOR BLOCK DELIVERY

2025/2026

Semester 1



Level HE4

Contents

1. Introduction

Module Tutor	Aamir Abbas
Module Tutor Email	A.Abbas@greatermanchester.ac.uk
Other Contact Methods	Microsoft Teams (Click to join - use code qz0a0td)
Availability	Monday: 12:00 – 1:00 PM
Weblink to Moodle Class	https://moodle.bolton.ac.uk/course/view.php?id=36009
Weblink to Module Specification	https://modules.bolton.ac.uk/COM4402/

2. Module Communications

The Module Tutor's contact details are provided above. You must check your **University of Bolton email address** and the **Moodle** area dedicated to this module regularly as many module communications are channelled through these media.

A Microsoft Teams Group (TEAMS-COM4402-Programming-B3-Dec2025) should be joined by using the join code qz0a0td. Each cohort has an assigned channel for streamline communication and will be used regularly to broadcast class activities. Announcements will be broadcast via Moodle and/or Teams.

Your Module Tutor will normally aim to respond to your email messages within **2 full working days** of receipt. However, responses will be longer in holiday periods.

3. Module Description

This module provides a comprehensive introduction to programming, with a focus on the theory and practice of fundamental programming concepts. It is designed for students who are new to computer programming, and no prior experience is required.

The module covers the core principles of programming, including variables and assignment, primitive data types, functions, and control structures for selection and iteration. Students will also work with collections and basic data structures. Alongside coding, the module introduces essential software development practices such as problem solving, pseudocode and/or flowcharts, program design, testing, debugging, and documentation. Through practical exercises and guided activities, students will develop the skills to design, implement, and test functional software solutions that meet basic problem specifications.

The module draws on contemporary programming practices by using Python, a language widely adopted across industries such as data science, AI, software engineering, and web development. Teaching makes use of modern tools including GitHub Classroom for version control and automated feedback, Kahoot/Menti for real-time formative assessment, and industry-standard IDEs such as PyCharm and VS Code. The emphasis on coding style (PEP 8), modular design, and testing mirrors current professional software development practices.

The module is designed with the cohort's diverse backgrounds in mind, particularly as many students are complete beginners in programming. It uses a block teaching model with a strong emphasis on practical, hands-on learning to build confidence quickly. Learning activities incorporate pair programming, collaborative debugging, and scaffolded exercises that allow

students to progress at their own pace while still being supported. Automated checks and frequent feedback help students avoid falling behind and cater to different learning speeds.

The module forms a foundation for the programme by introducing the fundamental concepts and problem-solving strategies that underpin all later computing modules. It establishes essential programming skills, structured design methods, and debugging practices that students will continue to apply in areas such as algorithms, software engineering, data science, and applied computing projects. Its focus on coding conventions and modularity ensures alignment with subsequent modules that build on these skills.

The module consolidates students' learning through a portfolio-based assessment strategy that integrates design, implementation, and demonstration. The delivery plan is structured to progressively reinforce knowledge: concepts introduced in lecturettes are immediately applied in live coding and labs, while weekly self-study tasks extend understanding. Coursework tasks and the final mini-project require students to draw together variables, control structures, functions, and collections into a complete working program, ensuring that learning is embedded and transferable.

4. Learning Outcomes and Assessments

Module Learning Outcomes	Assessment No. and Type
LO1: Evaluate problem requirements and design software solutions using appropriate structured techniques.	Assessment 1
LO2: Implement code solutions using accepted programming conventions, language constructs, and data types.	Assessment 2
LO3: Design and apply effective testing strategies to detect and correct programming faults during development.	Assessment 2

5. Assessment Deadlines

Assessment item		Due Date	Weight
1	**Part 1 Design:** Flowchart and Pseudocode Reports	21 December 2025, 03:00 PM	30%
2	**Part 2 Implementation:** Program Code and Report	25 January 2026, 03:00 PM	70%

6. Assessment Feedback

Feedback on items of assessment can be formal (such as on a signed feedback form) or informal (such as advice from a tutor in a tutorial). Feedback is therefore not just your grade or the comments written on your feedback form, it is advice you get from your tutor and sometimes your peers about how your work is progressing, how well you have done, what further actions you might take.

We recognise the value of prompt feedback on work submitted. Other than in exceptional circumstances (such as might be caused by staff illness), **you can expect your work to be marked and feedback provided not more than 5 working days** from the deadline date. However, please note that such feedback will be provisional and not agreed until the Assessment Board has met and may therefore be subject to change.

Please take time to read or listen to your assessment feedback. This can be very useful in determining your strengths and key areas for development, and can therefore help you improve on future grades.

7. Module Calendar

Session No.	Week Commencing	Topics Covered	Planned Delivery Method(s)
1 Week 1	8 Dec 25	Introduction, PyCharm IDE and 1st Python Program	On campus
2 Week 1	8 Dec 25	Flowchart's introduction	On campus
3 Week 1	8 Dec 25	Flowchart's continued	On campus
4 Week 1	8 Dec 25	Pseudocode + A1 Released	On campus
5 Week 2	15 Dec 25	I/O in Python, Variables, Primitive Data Types	On campus
6 Week 2	15 Dec 25	Type conversion, Formatting and String Operations	On campus
7 Week 2	15 Dec 25	Decision making with if/else statements	On campus
8 Week 2	15 Dec 25	Switch Statements + A1 Submission	On campus
9 Week 3	05 Jan 26	For loops	On campus
10 Week 3	05 Jan 26	While and do-while loops	On campus
11 Week 3	05 Jan 26	Loop applications	On campus
12 Week 3	05 Jan 26	Combining selection and iteration + A2 Released	On campus
13 Week 4	12 Jan 26	Functions and methods introduction	On campus
14 Week 4	12 Jan 26	Scope and lifetime of variables	On campus
15 Week 4	12 Jan 26	Lists (Introduction to collections)	On campus
16 Week 4	12 Jan 26	Iterating over arrays	On campus
17 Week 5	19 Jan 26	Dictionaries and Sets	On campus
18 Week 5	19 Jan 26	Mini project workshop	On campus
19 Week 5	19 Jan 26	Review and support session	On campus
20 Week 5	19 Jan 26	Project showcase and Viva + A2 Deadline	On campus

8. Continuous Assessment Strategy

This module will be assessed through a portfolio of activities, including a design documents report, a software implementation and testing report, and a practical demonstration. The design documents report will assess the learner's ability to analyse a given problem and present structured solutions using pseudocode and program logic. The implementation and testing report will enable learners to apply standard programming constructs and conventions to develop a working solution and explain their testing and debugging strategies. The practical

demonstration will give students the opportunity to present their completed program, describe its functionality, and discuss how it meets the given problem requirements.

9. Indicative Reading – Essential and Recommended

Essential Reading

Russ Miles & Kim Hamilton (2023) Learning UML 2.0: A Pragmatic Introduction to UML. Sebastopol, CA: O'Reilly Media.
Severance, C.R. (2023) Python for Everybody: Exploring Data in Python 3. 2nd ed. Ann Arbor, MI: Charles Severance.

Recommended Reading

Al Sweigart (2024) The Big Book of Small Python Projects. San Francisco, CA: No Starch Press.
Chaudhuri, A. B. (2020) Flowchart and Algorithm Basics: The Art of Programming. Mercury Learning and Information
Henney, K. (2020) 97 Things Every Programmer Should Know. O'Reilly.
Luciano Ramalho (2023) Fluent Python: Clear, Concise, and Effective Programming. 2nd ed. Sebastopol, CA: O'Reilly Media.
Martin, R.C. (2008) Clean Code: A Handbook of Agile Software Craftsmanship. Pearson.
Reuven M. Lerner (2024) Python Workout: 50 Essential Exercises. 2nd ed. Shelter Island, NY: Manning Publications.

10. Guidelines for the Preparation and Submission of Written Assessments

1. Written assessments should be word-processed in Arial or Calibri Light font size 12. There should be double-spacing and each page should be numbered.

2. There should be a title page identifying the programme name, module title, assessment title, your student number, your marking tutor and the date of submission.

3. You should include a word-count (excluding reference list/bibliography, figures, tables and appendices).
   Where a word limit is specified, the following penalty systems applies (For HE5, HE6 & HE7 Modules only):
   - Up to 10% over the specified word length = no penalty
   - 10 – 20% over the specified indicative word length = 5 marks subtracted (but if the assessment would normally gain a pass mark, then the final mark to be no lower than the pass mark for the assessment).
   - More than 20% over the indicative word length = if the assessment would normally gain a pass mark or more, then the final mark will be capped at the pass mark for the assessment.

4. At the end of the assessment you should include a declaration of any software tools including Generative AI (GAI) applications that you used in developing and completing the assessment. The assessment brief will specify if and how you can use GAI applications in the assessment.
5. All written work should be referenced using the standard University of Bolton's Harvard referencing style – see: https://libguides.bolton.ac.uk/resources/referencing/

6. Unless otherwise notified by your Module Tutor, electronic copies of assignments should be saved as word documents and uploaded into Turnitin via the Moodle class area. If you experience problems in uploading your work, then you must send an electronic copy of your assessment to your Module Tutor via email BEFORE the due date/time.

7. Please note that when you submit your work to Moodle, it will automatically be checked for matches against other electronic information, as well as for hidden text characters and GAI generated text. You will be able to see similarity matches but not currently flags for hidden characters and AI-generated text. The outcomes of Turnitin reports may be used as evidence in an academic misconduct investigation (see Section 14).

11 Late work

Late work will be subject to the following penalties:

- Up to 5 calendar days late = 10 marks (If HE5 & HE6) subtracted but if the assignment would normally gain a pass mark, then the final mark to be no lower than the pass mark for the assignment.
- More than 5 calendar days late **=** This will be counted as non-submission and no marks will be recorded unless this has been agreed with the Programme Leader, Module tutor or Personal Tutor.

Late submission of assessments on refer and those which are graded Pass/Fail only, is not permitted unless an extension is approved. See below.

12. Extensions

In the case of exceptional and unforeseen circumstances, an extension of up to 5 days after the assessment deadline may be requested using the standard University Extension Request Form. For approval there would need to be an explanation and evidence of relevant circumstances.

Longer extensions for individual assessments, projects and artefacts may be granted, at the discretion of the Programme Leader.

Requests for extensions which take a submission date past the end of the due date must be made using the Special Circumstances procedure.

Some students with registered disabilities will be eligible for revised submission deadlines. Revised submission deadlines for disability adjustments do not require the completion extension request paperwork. However, students should request these in writing in advance.

Please note that the failure of data storage systems is not considered to be a valid reason for an extension. It is therefore important that you keep multiple copies of your work on different storage devices before submitting it.

13. Academic Misconduct

Academic misconduct may be defined as any attempt by a student to gain an unfair advantage in any assessment. This includes plagiarism, collusion, commissioning (contract cheating and unauthorised use of GAI) amongst other offences.

In order to avoid these types of academic misconduct, you should **ensure that all your work is your own and that sources and software applications are attributed.** You can also check originality through *Turnitin.*

Please note that penalties apply if academic misconduct is proven. See the following link for further details:
https://greatermanchester.ac.uk/student-policy-zone/student-policies-2025-26/academic-misconduct-regulations-and-procedures-2025-26

14. Assessments

Background: The Digital Transition of Holton College

Holton College is embarking on an ambitious digital transformation initiative to modernise its teaching and learning practices. A key pain point identified by the faculty is the administration of summative quizzes. Currently, these are conducted on paper, a process that is time-consuming for teachers to create, administer, and mark, and slow for students to receive feedback on.

The Challenge:

The College's IT Department has outlined a need for a basic **Digital Quizzing System**. This system aims to digitise the process, allowing teachers to create quizzes and students to attempt them on a computer, with automatic marking and immediate score calculation. This will free up valuable teaching time and provide students with faster feedback to aid their learning.

Your Role:

Our software company, SuperDuper Software Maker Ltd., has been hired to develop a **Proof of Concept (PoC)** application. The goal of this PoC is not to build the full, complex system with databases and user logins, but to demonstrate the core functionality in a simple, working prototype. This prototype will be a console-based application written in Python, proving that the fundamental logic of a digital quiz is sound and viable.

The Proof of Concept Goal:

Your task is to develop a program that models one specific quiz. The program must:

1. **Store** a set of questions, multiple-choice options, and the correct answers.
2. **Present** the quiz to a student one question at a time.
3. **Accept** the student's answers.
4. **Calculate** and **display** the student's final score automatically.

Successfully demonstrating this PoC will be a crucial first step in securing further funding from the college for the development of a full-scale application.

Assessment 1 Brief

Assessment Number	001
Assessment Type (and weighting)	Portfolio component – 30%
Assessment Name	Part 1 Design
Assessment Submission Date	Sunday, 21 December 2025, 03:00 PM
Submission Links	Assessment 1: Design - FLOWCHARTS Assignment 1: Design - PSEUDOCODE

Learning Outcomes Assessed:

LO1: Evaluate problem requirements and design software solutions using appropriate structured techniques.

Use of Generative Artificial Intelligence (GAI) Applications in this Assessment

AI Status	Application	Notes
Category A	No GAI tool is permitted.	While grammar and/or spell checkers may be used to correct individual words and sentences, the use of GAI is not allowed. This is because the learning outcomes require you to produce original assessment work without any GAI assistance Any GAI outputs which are presented as your own original work/skills and are not authorised and acknowledged will be assessed for academic misconduct.

Assessment Task:

Our company, SuperDuper Software Maker Ltd., is developing a Proof of Concept (PoC) for Holton College. The goal is to create a simple console-based quiz application in Python that demonstrates the core functionality of a digital assessment tool. The PoC must store quiz questions, present them, accept answers, and calculate a score.

Your Task: The Design Phase

Before writing any code, you must plan your solution using structured techniques. You are required to produce two design documents that outline the complete logic and flow of your program. This demonstrates your ability to analyse a problem and design a structured software solution, as outlined in Learning Outcome 1.

Your submission will consist of two parts:

1. Program Flowchart (15%)

Objective: To visually represent the flow of control and logic within your quiz program.

Requirements:

- Your flowchart must be a clear, professional diagram created using standard flowchart symbols (e.g., oval pills for start/end, parallelograms for input/output, rectangles for processes, diamonds for decisions).
- It must model the entire program flow, from start to finish.

- It must address all key business requirements from the scenario:
    - **Initialisation:** How the quiz questions and answers are stored in the program's memory (e.g., using lists/tuples).
    - **Presentation:** The process of displaying a question and its multiple-choice options to the user.
    - **Input:** How the program accepts the user's (student's) answer.
    - **Validation:** The decision process for checking if the user's answer is correct or incorrect.
    - **Scoring:** The process of tracking the user's score after each question.
    - **Iteration:** The loop that controls moving to the next question.
    - **Termination:** The process that occurs after the last question: displaying the final score and ending the program.
- The flowchart must be exported and submitted as a single **Word (*.docx)** document with page size set to A2.

2. Pseudocode (15%)

Objective: To describe the logic and structure of your program using a simplified, language-independent textual format.

Requirements:

- Your pseudocode must be a detailed, step-by-step description of your program's algorithm.
- It must be written in plain English, structured clearly with indentation to show loops, conditional statements, and program blocks (e.g., IF, THEN, ELSE, FOR, WHILE).
- It must **explicitly address all components** of the problem statement:
    - The **data structures** used to store questions, options, and answers (e.g., LIST, TUPLE).
    - The initialisation of a **score variable**.
    - The mechanism for **iterating** through all quiz questions.
    - The **output** of each question and its choices.
    - The **input** of the user's answer.
    - The **conditional logic** (decision-making) to compare the user's answer against the correct answer.
    - The **operation** of updating the score if the answer is correct.
    - The **output** of the final score at the end of the quiz.
- The pseudocode should be submitted as a **Text document (*.txt)**.

Submission Instructions

Please adhere to the following instructions carefully. Submissions that do not meet these requirements may be subject to penalties.

1. **Submission Points:** You will be provided with two separate submission links on Moodle:
    - **One link for your Flowchart**
    - **One link for your Pseudocode**
2. **File Formats and Requirements:**
    - **Flowchart:**
        - Must be submitted as a **single Microsoft Word document (.docx)**.
        - The document page size must be set to **A2**. This ensures that large flowchart diagrams are clear, legible, and fit onto a single page per sheet.

        - You may be provided with a Word template to use for this purpose. It is recommended to download and edit this template.
        - Ensure the flowchart images are embedded correctly in the Word document and is not an external link.
        - Use different pages for multiple images.
    - **Pseudocode:**
        - Must be submitted as a **plain text file (.txt)**.
        - The pseudocode must be written in a clear, structured format using indentation to denote code blocks (e.g., for loops, if statements).
        - Do not submit pseudocode as a Word document or PDF for this part.
3. **File Naming Convention:**
    - Flowchart File Name: STUDENTNUMBER_COM4402_Flowchart.docx
        - Example: 2533125_COM4402_Flowchart.docx
    - Pseudocode File Name: STUDENTNUMBER_COM4402_Pseudocode.txt
        - Example: 2533125_COM4402_Pseudocode.txt
    - Replace STUDENTNUMBER with your own student ID number.
4. **Document Contents:**
    - The first page of your Flowchart Word document and the first few lines of your pseudocode **must** contain the following information:
        - Student Number
        - Student Name
        - Module Code (COM4402)
        - Module Title (Programming)
    - Please ensure that the **document size is set to A2**.
5. **Template:**
    - A Microsoft Word template for the flowchart, pre-configured with the correct A2 page size and a cover page for your details, may be provided by your lecturer. It is your responsibility to download and use this template if available.

Failure to follow these submission instructions may result in a delay in marking or the application of a penalty.

Marking Criteria

Your submission will be assessed based on your ability to:

- **Evaluate:** The problem requirements should be evaluated accurately and completely.
- **Design:** A concise, logical and efficient software solution should be designed.
- **Application:** Appropriate structured techniques (flowcharts, pseudocode) to represent your design must be applied.
- **Completeness:** Address all specified business and functional requirements in your design.

Important Note: This assessment evaluates your design skills. A successful design that fully addresses the scenario will form a solid foundation for the implementation in Assessment 001 (Part 2 - Implementation).

Declaration: At the end of the assessment you should also include a declaration of any software tools including Generative AI (GAI) applications that you used in developing and completing the assessment.

Minimum Secondary Research Source Requirements:

Level HE4 - It is expected that the Reference List will contain between five and ten sources. As a MINIMUM the Reference List should include one refereed academic journal and three academic books

Specific Assessment Criteria/Marking Scheme:

Exceeds Expectations

The design documents demonstrate excellence and insight. The flowchart is a comprehensive, professional-grade diagram that uses correct symbols and perfectly visualizes the program's flow, including error handling. The pseudocode is exceptionally detailed, logically structured, and could be translated into code with minimal effort. Together, they elegantly and completely address all business requirements and demonstrate a sophisticated understanding of the problem and its solution.

Pass/Meets Expectations

The design documents meet the core requirements. The flowchart correctly represents the main program flow using appropriate symbols, though it may lack some details or minor elements. The pseudocode describes the main logic of the solution. Together, they address the key functional requirements of the scenario, providing a solid foundation for implementation.

Fail / Does Not Meet Expectations

The design documents fail to meet the core requirements. The flowchart is missing, incorrect, incoherent, or uses non-standard symbols. The pseudocode is missing, is not actually pseudocode (e.g., is just a paragraph of text), or bears little relation to the problem. The submission demonstrates a significant misunderstanding of the task or provides a design that does not address the core business needs of the quiz application.

Also see Section 16: General Assessment Criteria for Written Assessments

Assessment 2 Brief

Assessment Number	002
Assessment Type (and weighting)	Portfolio component – 70%
Assessment Name	Part 2 Implementation
Assessment Submission Date	Sunday, 25 January 2026, 03:00 PM
Submission Link	Assignment 2: Implementation - REPORT

Learning Outcomes Assessed:

LO2: Implement code solutions using accepted programming conventions, language constructs, and data types.

LO3: Design and apply effective testing strategies to detect and correct programming faults during development.

Use of Generative Artificial Intelligence (GAI) Applications in this Assessment

AI Status	Application	Notes
Category A	No GAI tool is permitted.	While grammar and/or spell checkers may be used to correct individual words and sentences, the use of GAI is not allowed. This is because the learning outcomes require you to produce original assessment work without any GAI assistance Any GAI outputs which are presented as your own original work/skills and are not authorised and acknowledged will be assessed for academic misconduct.

Assessment Task:

Following the successful approval of your design documents, SuperDuper Software Maker Ltd. has now tasked you with developing the Proof of Concept (PoC) for the Holton College Digital Quiz System. This phase involves building a functional, well-tested, and professionally presented Python application.

Your Task: Implementation & Testing

You are required to develop the console-based quiz application according to the core requirements. You must use the Python programming language and adhere to standard coding conventions.

Core Functional Requirements:

Your program must demonstrate the following functionality:

1. **Data Storage:** Use appropriate data structures (e.g., lists, tuples, dictionaries) to store at least **5 quiz questions**. Each question must include:
    - The question text.
    - Multiple-choice options (e.g., 1, 2, 3, 4).
    - The correct answer.
2. **Quiz Execution:** The program must:
    - Iterate through each question one at a time.

    - Print the question and the multiple-choice options to the console.
    - Prompt the user to input their answer (e.g., by typing '1', '2', '3', or '4').
3. **Scoring Logic:** The program must:
    - Check if the user's answer matches the correct answer.
    - Keep a running total of the user's score.
4. **Results Display:** After the final question, the program must:
    - Print the user's final score (e.g., "You scored 3 out of 5").
    - Display a closing message.

Non-Functional & Development Requirements:

- **Code Quality:** Code must be clean, readable, and well-commented. It must follow Python naming conventions and PEP 8 style guide where appropriate (e.g., meaningful variable names, use of whitespace).
- **Modular Design:** Code must be written in a modular fashion. This means separating logic into functions (e.g., a run_quiz() function, a check_answer() function) rather than writing all code in a single block.
- **Version Control:** You must use Git for version control. Your entire development history must be visible in your GitHub repository, with frequent commits over the development period showing your progress.
- **Testing:** You must rigorously test your program to ensure it is robust and functions correctly.

The Testing & Development Report

You must submit a report that documents your development process and proves the functionality of your application. The report must be submitted as a **Word document (.docx)** and contain the following sections:

1. **Introduction:** Briefly restate the project's purpose.
2. **GitHub Link:** A publicly accessible link to the repository containing your application must be provided.
3. **Development Process & Decisions:**
    - Explain key decisions you made during development. For example:
        - Why did you choose a list of dictionaries over a list of tuples to store the questions? (Or vice versa).
        - How did you structure your code into functions and why?
        - Did you encounter any problems? How did you solve them (e.g., using a debugger, print statements)?
        - How does your final implementation compare to your original flowchart/pseudocode? Explain any changes and the reasons for them.
4. **Testing Strategy:**
    - Describe how you tested your code. Mention both manual testing (running the program) and any automated checks you might have performed.
    - **Provide a minimum of 10 specific test cases.** Each test case must include:
        - Test ID (e.g., TC01)
        - Test Description (What are you testing? e.g., "Testing correct answer increases score")
        - Input Data (What input did you give?) Provide screenshot of input
        - Expected Outcome (What should happen?)
        - Actual Outcome (What actually happened?) Provide screenshot of output

- Pass/Fail

Sample Execution

Your program's output should be clear and user-friendly. Below is an example of how it might look and function:

```
Welcome to the Holton College Quiz!
Please answer with the letter (1, 2, 3, or 4) of your choice.

Question 1: What is the capital of France?
1. London
2. Paris
3. Berlin
4. Madrid
Your answer: 2
Correct!

Question 2: Which data structure uses key-value pairs?
1. List
2. Tuple
3. Dictionary
4. Set
Your answer: 3
Correct!

... (Questions 3-5) ...

Quiz Complete!
You scored 4 out of 5 correct.
Thank you for playing!
```

Submission Instructions for Part B

1. **GitHub Repository:**
   - You must create a **public** GitHub repository for your project.
   - The repository must contain all your Python source code files (e.g., quiz.py).
   - The commit history must show regular progress over the development period. A single large commit at the deadline will be penalised.
   - The repository must include a README.md file with your student number, module code, and simple instructions on how to run your code (e.g., "Ensure Python 3 is installed. Run the program by typing python quiz.py in the terminal.").
2. **Report:**
   - The report must be submitted as a **single Microsoft Word document (.docx)**.
   - The report must contain your student number, name, module name and code.
3. **File Naming Convention:**
   - **Report File Name:** STUDENTNUMBER_COM4402_Report.docx
     - Example: 2532851_COM4402_Report.docx
4. **What to Submit on Moodle:**
   - You will be provided with a submission link on Moodle.

- You must submit a single Word document report that also contains the clickable URL (web address) to your public GitHub repository. Please ensure the link is correct and accessible.

Failure to follow these submission instructions may result in a delay in marking or the application of a penalty.

Marking Criteria

Your submission will be assessed based on your ability to:

- **Implement:** A functional, robust, and runnable code solution must be implemented using accepted Python conventions, language constructs, and data types.
- **Test:** Effective and rigorous testing strategies must be designed and applied to detect and correct programming faults during development. This must be thoroughly documented.
- **Articulate:** The development process and the rationale behind key decisions (e.g., choice of data structures, modular design) must be clearly explained.
- **Professionalism:** The code must be clean, readable, and modular. The version control history must demonstrate a consistent and logical development process over time.

Important Note: This assessment evaluates your practical coding, testing, and professional development skills. A successful implementation that is well-tested and professionally presented proves the viability of the PoC for the client.

Declaration: At the end of the assessment you should also include a declaration of any software tools including Generative AI (GAI) applications that you used in developing and completing the assessment.

Minimum Secondary Research Source Requirements:

Level HE4 - It is expected that the Reference List will contain between **five and ten sources**. As a MINIMUM the Reference List should include **one refereed academic journal** and **three academic books**

Specific Assessment Criteria/Marking Scheme:

Exceeds Expectations

The submission demonstrates excellence across all criteria. The code is fully functional, robust, and handles errors gracefully. It is exceptionally well-structured, readable, and adheres to best practices. The testing report is comprehensive, containing 10+ well-documented test cases that cover valid, invalid, and edge case inputs. The development rationale is insightful and clearly explains the evolution from design to implementation. The GitHub commit history is rich and shows clear, logical progression.

Pass/Meets Expectations

The submission meets the core requirements. The code is functional and addresses the problem statement, though it may have minor bugs or lack elegance. The code is readable and shows an attempt at structure. A testing report is provided with an adequate number of test cases (at or

near 10), primarily covering the main functionalities. The development rationale describes what was done, if not deeply why. A GitHub repository with a visible commit history is present.

Fail / Does Not Meet Expectations

The submission fails to meet the core requirements. The code is non-functional, severely incomplete, or demonstrates a minimal attempt. The code is unstructured and difficult to read. The testing report is missing, wholly inadequate, or does not reflect the submitted code. Little to no explanation of the development process is provided. The GitHub repository is missing, has no meaningful commit history, or does not contain the correct code.

Also see Section 16: General Assessment Criteria for Written Assessments

16. GENERAL ASSESSMENT GUIDELINES – LEVEL HE4 (Block Modules)

	Relevance **Learning outcomes must be met for an overall pass**	Knowledge and Understanding	Analysis, Creativity and Problem-Solving	Self-awareness and Reflection	Research/ Referencing	Written English	Presentation and Structure
Pass	Work is directly relevant and effectively addresses the requirements of the brief. **Learning outcomes are met.**	Demonstrates breadth of knowledge and understanding of theory and practice to meet the learning and competency requirements for this level. Demonstrates clear understanding of key concepts in different contexts.	Presents a cohesive appraisal of findings through the analysis of information. Draws clear, justified and thoughtful conclusions. Demonstrates creative flair, originality and initiative. Demonstrates a good understanding of problem-solving approaches and applies strong problem-solving skills.	Provides insightful reflection and self-awareness in relation to the outcomes of own work and personal responsibility.	A wide range of contemporary and relevant reference sources selected and drawn upon. Sources cited accurately in both the body of text and in the Reference List/ Bibliography.	Writing style is clear and appropriate to the requirements of the assessment. A well written answer with competent spelling, grammar and punctuation. For example, paragraphs are well structured and include linking and signposting. Sentences are complete and different types are used. A wide range of appropriate vocabulary is used.	The presentational style and layouts are correct for the type of assignment. Evidence of effective planning and logically structured. Where relevant, there is effective placement of, and reference to, figures, tables and images.
Fail (refer)	Work does not address the requirements of the brief. Irrelevant and superficial content. **One or more learning outcomes have not been met.**	Demonstrates inadequate knowledge and understanding of theory and practice for this level. Demonstrates insufficient understanding of key concepts.	Presents a limited discussion of findings with little consideration of the quality of information drawn upon. Draws irrelevant conclusions. Creativity, initiative and problem-solving skills are absent.	Provides inadequate reflection and self-awareness in relation to the outcomes of own work and personal responsibility, when required.	There is an absence of relevant sources. Poor referencing technique employed.	Writing style is unclear and does not match the requirements of the assessment in question. Deficiencies in spelling, grammar and punctuation makes reading difficult and arguments unclear. Unstructured paragraphs.	For the type of assignment the presentational style, layout and/or structure are lacking. Figures, tables and images are absent when required or lack relevance/clarity.