Unit 4061: Programming for Engineers

Unit Code: **A/650/2923**

Level: **4**

Credits: **15**

Introduction

With the increasing programmability of devices, it is essential that engineers can define and develop software artefacts. Engineers are often involved in developing programs for a wide variety of projects, such as creating firmware, automating robots and machines, modelling conceptual designs, processing data, and developing machine-learning models. By acquiring programming competencies, engineers can meet these challenges, reap the benefits of customised designs, and develop solutions to solve future engineering problems, thus enhancing their career prospects.

This unit provides engineering students with a comprehensive introduction to programming. Students will be able to investigate different software development platforms, programming paradigms, programming languages (e.g. Python, C or C++), and their engineering applications. They will gain the experience of going through a standard development process; from setting requirements through to design, implementation, testing and maintenance. The unit also covers program design, structure, and syntax through project activities. Students will be assessed on creating programs that are efficient, functional, reliable, and maintainable.

On completion of this unit, students will have acquired essential knowledge and skills in programming using a popular language that can be utilised in Level 5 units such as Machine Learning and Embedded Systems.

Learning Outcomes

By the end of this unit students will be able to:

LO1 Discuss key aspects of software evolution and development in the context of engineering applications

LO2 Design a programming solution for an engineering problem

LO3 Implement a programming solution for an engineering problem

LO4 Perform testing of the programming solution to meet defined requirements and to ensure high-quality outputs.

Essential Content

LO1 **Discuss key aspects of software evolution and development in the context of engineering applications**

Programming in engineering:

Evolution of programming concepts; paradigms (e.g. object-oriented, event-driven, procedural, functional); development platforms including IDEs; current/future trends

Processes/components of programming environment (i.e. microcomputer hardware: CPU, arithmetic logic unit (ALU), registers, memory; fetch-execute cycles)

Devices/systems that can be programmed (e.g. computers, mobile phones, tablets, industrial controllers, field-programmable gate arrays (FPGAs))

Programming languages and platforms with which to program devices/systems (e.g. Python, C, C++, C#, ADA, Java and MATLAB); comparisons (e.g. compiled versus interpreted languages).

Engineering applications and practical skills:

Project-based learning (PBL) approach for understanding planning, development and delivery of small/medium-sized engineering applications

Software engineering principles, software development life cycle, methodologies (e.g. agile, waterfall), roles and responsibilities of a development team (e.g. analyst, programmer, tester, Scrum master, product owner), modelling and prototyping. Overview of Engineering project management techniques for programmers (e.g. SWOT, stakeholder matrices, risk mapping, radar chart and summary risk profiles).

Edit, execute and test example engineering applications

Developer attributes: responsibility towards planning and prioritisation of development activities in meeting business needs, ability to work independently, pro-active, initiative, communicative, keen to analyse root cause of problems, contextual knowledge and skills for practice, solve and develop efficient and ethical solutions

Programming case studies:

Embedded systems, automation, Industry 4.0, machine learning (AI), networking, Internet of Things (e.g. smart factories), cloud computing, cybersecurity; concepts, purpose and application

Industry relevance (e.g. manufacturing, defence, medical, automotive, aeronautics, space technologies, utilities, consumer goods)

Occupation-centric: programming tools for diagnostics (e.g. web-based diagnostics for network devices and other software tools such as PROFITrace), interconnected occupational competencies (e.g. network engineers to bring together programming skills and network installation and management skills to solve problems).

Best practices:

Coding standards, secure programming, green coding, programmer ethics, accessibility.

LO2 **Design a programming solution for an engineering problem**

Program design, structure and maintenance:

Requirements analysis and specification, flow and function charts, pseudocode, selection and application of design methodology, design for testing and maintenance, occupational role and relevance in designing maintainable software solutions (e.g. use of software tools/techniques for troubleshooting network issues, securely isolate and debug faults, automate different aspects of network maintenance)

Documentation of design (e.g. project name, description, version control such as Git and commentary); reading, extracting and interpreting technical, business related and other relevant documentation.

Programming features:

Data types and operators (i.e. integers, floating point, strings, characters, Boolean, arithmetic, relational, logical, bitwise, assignment)

Data type qualifiers (e.g. mutable and immutable)

Classes and object-oriented programming (OOP) concepts (i.e. abstraction, polymorphism, encapsulation, inheritance)

Data structures (i.e. arrays, lists, sets)

Control structures (i.e. decision, selection, and iterative statements)

Input/output (i.e. file reading and writing, standard I/O, databases)

Libraries (i.e. GUI, networking, logging)

Data management: cleaning data, producing statistical analysis of data.

Algorithmic design and development:

Example algorithms for engineering problems (e.g., path finding)

Design algorithms for a range of small engineering applications

Complexity analysis, Big-O notation.

LO3 **Implement a programming solution for an engineering problem**

Benefits of modular design:

Development efficiency, maintainability, testability, reusability and debugging.

Declaring, defining and calling functions:

Naming, return type and arguments (parameters), function body

Passing data to and receiving data from functions, call functions by value, and call by reference

Life cycle of variables in functions (e.g. global versus local, class versus instance)

Recursive functions.

Preprocessor directives:

Include, import statements, C header files, macro definitions, sharing between multiple source files, #define, #ifndef statements

Python packages.

Program development and implementation:

Develop and implement small engineering applications using a suitable programming language; develop documentation to industry standards and style guides

Explore team approach to program development and delivery

Consider possible user-experience concerns and how these could be solved.

LO4 **Perform testing of the programming solution to meet defined requirements and to ensure high-quality outputs**

Overview of testing:

Software testing frameworks and methodologies including functional (e.g. unit testing, integration, system, acceptance) and non-functional (e.g. usability, performance, security, compatibility) methods; tools and techniques to monitor and enhance performance against requirements

Test environments

Continuous integration/continuous development (CI/CD) pipeline and continuous testing.

Approach to testing:

Relationship between test activities and program development activities; identify elements that need to be tested; consider data that should be used to fully test the program; match tests against the defined requirements (e.g. user, system); use of test harnesses

Use of relevant test procedures: test plans, test techniques (e.g. open-box, closed-box); testing documentation (e.g. reports, plans, checklists)

Overview of alpha and beta testing.

Debugging:

Use of debugger tools; documentation of the debugging process with reference to watch lists, breakpoints, and tracing

Debugging the process to fix vulnerabilities, defects and bugs in code

Understand coding standards and their benefits when writing program code in a team as well as for the individual.

Learning Outcomes and Assessment Criteria

Pass	Merit	Distinction
LO1 Discuss key aspects of software evolution and development in the context of engineering applications		
P1 Discuss the key stages of the software development life cycle, including the roles and responsibilities of team members **P2** Present a choice of programming languages and development platforms for a given engineering problem.	**M1** Analyse the suitability of any two specific software life-cycle models for a given engineering problem.	**D1** Evaluate industry-recognised best practices in using software life-cycle models for engineering problems.
LO2 Design a programming solution for an engineering problem		**LO2, LO3 and LO4**
P3 Produce an outline requirements specification for a given engineering application **P4** Design a suitable algorithmic solution for the key requirements.	**M2** Refine the requirements specification and a suitable design solution to cover the full set of requirements, including modularity and maintainability.	**D2** Reflect on the design, implementation, testing and documentation aspects of engineering programming solutions, including use of coding standards and why it is necessary in a team as well as for the individual.
LO3 Implement a programming solution for an engineering problem		
P5 Implement a given design solution for an engineering problem using an appropriate programming language **P6** Demonstrate successful execution of the developed solution in a chosen programming environment.	**M3** Refine the implemented solution for modularity and maintainability.	

Pass	Merit	Distinction
LO4 Perform testing of the programming solution to meet defined requirements and to ensure high-quality outputs.		
P7 Produce a test plan to demonstrate whether the program meets the key requirements **P8** Perform tests on the program against the key requirements, resolving any functional errors.	**M4** Analyse the effectiveness of testing, including an explanation of the choice of tests used **M5** Demonstrate the use of debugging tools to identify and correct errors in a programming solution	

Recommended Resources

This unit does not specify which programme language should be used to deliver this content – this decision can be made by the academic staff.

Examples of languages that are used in industry are Python, C, C++, C#, ADA, Java, and MATLAB but any language which will allow the student to achieve the Learning Outcomes is acceptable.

Note: See HN Global for guidance on additional resources.

Print Resources

Bradley R. (2011) *Programming for Engineers: A Foundational Approach to Learning C and MATLAB*. Springer.

Clough D.E. and Chapra S.C. (2023) *Spreadsheet Problem Solving and Programming for Engineers and Scientists (Hardback).* Taylor & Francis Ltd.

Cyganek B. (2020) *Introduction to Programming with C++ for Engineers*. Wiley/IEEE Press.

Kenan A. (2020) *Python for Mechanical & Aerospace Engineering*.

Nagar S. (2017) *Introduction to Python for Engineers and Scientists*. Apress.

Sanchez J. and Canton M.P. (2017) *Java Programming for Engineers (Hardback).* Taylor & Francis Ltd.

Sierra K., Bates B. and Gee T. (2022) *Head First Java.* 3rd Ed. O'Reilly Media.

Sola A. (2021) *Hardcore Programming For Mechanical Engineers: Build Engineering Applications from Scratch (Paperback).* No Starch Press,US.

Wei-Bing J., Aizenman H., Espinel E.M.C., Gunnerson K. and Liu J. (2022) *An Introduction to Python Programming for Scientists and Engineers (Paperback).* Cambridge University Press.

Journals

Note: Example journals listed below provide a broad range of articles related to unit content and those relevant for the qualification. Staff and students are encouraged to explore these journals and any other suitable journals to support the development of academic study skills, and subject specific knowledge and skills as part of unit level delivery.

Advances in Engineering Software

Computer Applications in Engineering Education

Journal of Computer Science and Control Systems

Programming Journal.

Links

This unit links to the following related units:

Unit 5013: Embedded Systems

Unit 5047: Computer Architecture and Interfacing

Unit 5050: Machine Learning Systems and Programming.