Unit 1 Applied Computing 2025

Outcome 2 Programming – Template for developing an assessment task – Blank

Outcome 2

On completion of this unit the student should be able to interpret teacher-provided solution requirements to design and develop a software solution using an object-oriented programming language.

Key knowledge	Key skills
• emerging trends in programming languages and artificial intelligence-based (AI) code generators for the development of software solutions, such as: – low-code development approaches – readability and/or maintainability improvements • characteristics of functional and non-functional requirements, constraints and scope • key legal requirements relating to intellectual property and copyright while designing and developing software	• interpret solution requirements to develop a software solution
• design tools for representing the functionality and appearance of solution designs, such as: – mock-ups – input-process-output (IPO) charts – flowcharts/pseudocode	• select and use appropriate design tools to represent solution designs
• characteristics of data types, such as: – text (character, string) – numeric (integer, floating point, date/time) – Boolean • types of data structures, such as: – one-dimensional arrays – lists – records (varying data types, field index)	• use a range of data types and data structures
• principles of OOP, such as: – abstraction – encapsulation • features of a programming language, such as: – variables, and initialising, accessing and storing data in variables – control structures (sequence, selection and iteration/repetition) – arithmetic, logical and conditional operators – procedures, functions and methods	• develop a software solution using appropriate features of an OOP language

Assessment task development

Unit 1 Applied Computing 2025 Outcome 2 Programming – Template for developing an assessment task – Blank		
• naming conventions for solution elements, such as: – Hungarian notation – camel casing		
• purposes of internal documentation, such as: – explaining data and code structures – code maintenance	• document the functioning of a software solution through internal documentation	
• validation techniques for data, such as: – existence checking – type checking – range checking • debugging and testing techniques for checking software solutions function correctly, such as: – test tables to compare expected and actual output – construction of relevant test data – breakpoints – debugging output statements	• design and apply suitable validation, debugging and testing techniques	