Unit 3 Software Development 2025 Outcome 1 Software development: programming – Template for developing an assessment task – Blank		
Outcome 1 On completion of this unit the student should be able to interpret teacher-provided solution requirements and designs and use appropriate features of an object-oriented programming language to develop working software modules.		**Assessment task development**
Key knowledge	**Key skills**	
• emerging trends in programming using artificial intelligence, including: – using prompts to generate code – automated debugging and testing of modules – code optimisation – responsible and ethical use of artificial intelligence tools • characteristics of functional and non-functional requirements, constraints and scope • design tools for representing modules, including: – data dictionaries – mock-ups – object descriptions – input-process-output (IPO) charts – pseudocode	• interpret solution requirements and designs	
• characteristics of data types, including: – text (character, string) – numeric (integer, floating point, date/time) – Boolean • characteristics of data structures, including: – one-dimensional arrays – two-dimensional arrays – records (varying data types, field index) • characteristics of data sources (plain text (TXT), delimited (CSV) and XML files), including: – structure – reasons for use	• use a range of data types, data structures and data sources	
• principles of OOP, including: – abstraction – encapsulation – generalisation – inheritance • features of a programming language, including: – local and global variables, and constants – data types – instructions and control structures (sequence, selection, iteration/repetition)	• use and justify appropriate features of an OOP language to develop working software modules	

Unit 3 Software Development 2025 Outcome 1 Software development: programming – Template for developing an assessment task – Blank		
– arithmetic, logical and conditional operators – graphical user interfaces (GUIs) – functions and methods – classes and objects • algorithms for sorting and searching, including: – selection sort – quick sort – binary search – linear search		
• purposes and features of naming conventions for solution elements (variables, interface controls, code structures), including: – Hungarian notation – camel casing – snake casing • validation techniques for data, including: – existence checking – type checking – range checking	• develop and apply suitable naming conventions and validation techniques within modules	
• purposes of internal documentation, including: – explaining and justifying data and code structures – code maintenance – placeholder comments for future development (stubs)	• document the functioning of modules using internal documentation	
• types of errors, including: – syntax – logic – runtime (overflow, index out of range, type mismatch, divide by zero) • debugging and testing techniques for checking modules function correctly, including: – use of breakpoints – use of debugging statements – construction of relevant test data – test cases comparing expected and actual output in testing tables	• develop and apply suitable debugging and testing techniques using appropriate test data	