Programming requirements

The following is a list of programming requirements that are studied and used, and that students are expected to be able to apply. Note that this list is not exhaustive; learning does not have to be confined to the requirements listed below.

In the development of working software modules (Unit 3 Area of Study 1) and the software solution (Unit 4 Area of Study 1), the chosen object-oriented programming (OOP) language should provide students with the ability to carry out the development stage of the problem-solving methodology within the three conceptual layers of: interface, logic and data source.

Interface

The chosen OOP language must enable students to develop a graphical user interface (GUI) for use in a digital system through one or more of the following options:

- an Integrated Development Environment (IDE) (drag and drop/WYSIWYG)
- using code (same language)
- using code (a supporting language).

Logic

Programming requirements for the logic layer:

- instructions
- program control structures (sequence, selection and iteration/repetition)
- operators
  - *arithmetic:* addition, subtraction, multiplication, division, integer division, modulus, unary plus/minus, increment/decrement
  - *logical:* AND, OR, NOT
  - *conditional/comparison:* equality, inequality, less than, less than or equal to, greater than, greater than or equal to
- functions and methods
- classes and objects.

Data source

Programming requirements for the data source layer:

- initialise, set and access local and global variables, and constants

- using relevant data types (numeric, text, Boolean)
- read data from external sources, such as files and databases (local or cloud-based)
- write data to external sources, such as files and databases (local and cloud-based).

The following file formats that can be used are:

- delimited (CSV)
- plain text (TXT)
- XML.

Study Design KK and KS

characteristics of data types

- 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:**

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)
- arithmetic, logical and conditional operators
- graphical user interfaces (GUIs)
- functions and methods
- classes and objects

access modifiers (public, protected and private)

· **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

algorithms for sorting and searching, including:

- selection sort
- quick sort
- binary search
- linear search

· **established and innovative approaches to software development, including:**

- application programming interfaces (APIs) and libraries

-

Rubric

VCE Software Development: School-assessed Task 2025							
Assessment Criteria	**Levels of Performance**						
	Indicators	**Not shown**	**1–2 (very low)**	**3–4 (low)**	**5–6 (medium)**	**7–8 (high)**	**9–10 (very high)**
Unit 4 Outcome 1 6. **Skills in using the features of the programming language.**	• Uses a range of appropriate features of the programming language.	Insufficient evidence	Develops the solution using: • instructions • arithmetic operators.	Develops the solution using: • local variables and constants • logical and conditional operators • control structures of sequence and selection • graphical user interfaces (GUIs).	Develops the solution using: • global variables • the control structure of iteration/repetition • relevant GUI controls.	Develops the solution using: • functions and methods • access modifiers.	Develops the solution using: • classes and objects. Applies all relevant OOP principles.
	• Uses a range of appropriate data types, data structures and data sources.		Identifies data types, including text, numeric and Boolean.	Uses data types for local variables. Outlines, in the internal documentation, why the selected data types were used.	Uses appropriate data types for global variables and data structures, including arrays and/or records. Identifies, in the internal documentation, why the selected data types and data structures were used.	Uses appropriate data types, data structures and data sources. Describes, in the internal documentation, why the selected data sources were used.	Uses a range of data types, data structures and data sources. Explains in the internal documentation, why the selected data types, data structures and data sources were used.
		0 □	1 □ 2 □	3 □ 4 □	5 □ 6 □	7 □ 8 □	9 □ 10 □

VCE Software Development: School-assessed Task 2025							
Assessment Criteria	**Levels of Performance**						
	Indicators	**Not shown**	**1–2 (very low)**	**3–4 (low)**	**5–6 (medium)**	**7–8 (high)**	**9–10 (very high)**
Unit 4 Outcome 1 7. **Skills in developing the software solution.**	• Applies suitable naming conventions.	Insufficient evidence	Identifies naming conventions for use within the software solution.	Applies naming conventions to variables within the software solution.	Applies naming conventions to variables and interface controls within the software solution.	Applies naming conventions to variables, interface controls and code structures within the software solution.	Applies suitable naming conventions to all solution elements within the software solution.
	• Writes comprehensive internal documentation.		Identifies the functioning of the software solution.	Outlines the functioning of the software solution using internal documentation.	Uses internal documentation to describe the: • functionality of the software solution • use of data. Displays evidence of code maintenance. Some issues with clarity exist.	Uses internal documentation to explain the: • functionality of the software solution • use of data • use of code structures. Minor issues with clarity exist.	Uses internal documentation to explain the: • functionality of the software solution • use of all data • use of all code structures. Clear and concise.
	• Applies appropriate validation techniques.		Identifies input data for validation.	Validates data using one of the following checks: • existence • type • range.	Validates data using two of the following checks: • existence • type • range. Some inconsistencies are present.	Validates data using all the following checks: • existence • type • range. Minor inconsistencies are present.	Validates all relevant input data and checks the reasonableness and completeness of all input data. No inconsistencies are present.
		0 □	1 □ 2 □	3 □ 4 □	5 □ 6 □	7 □ 8 □	9 □ 10 □