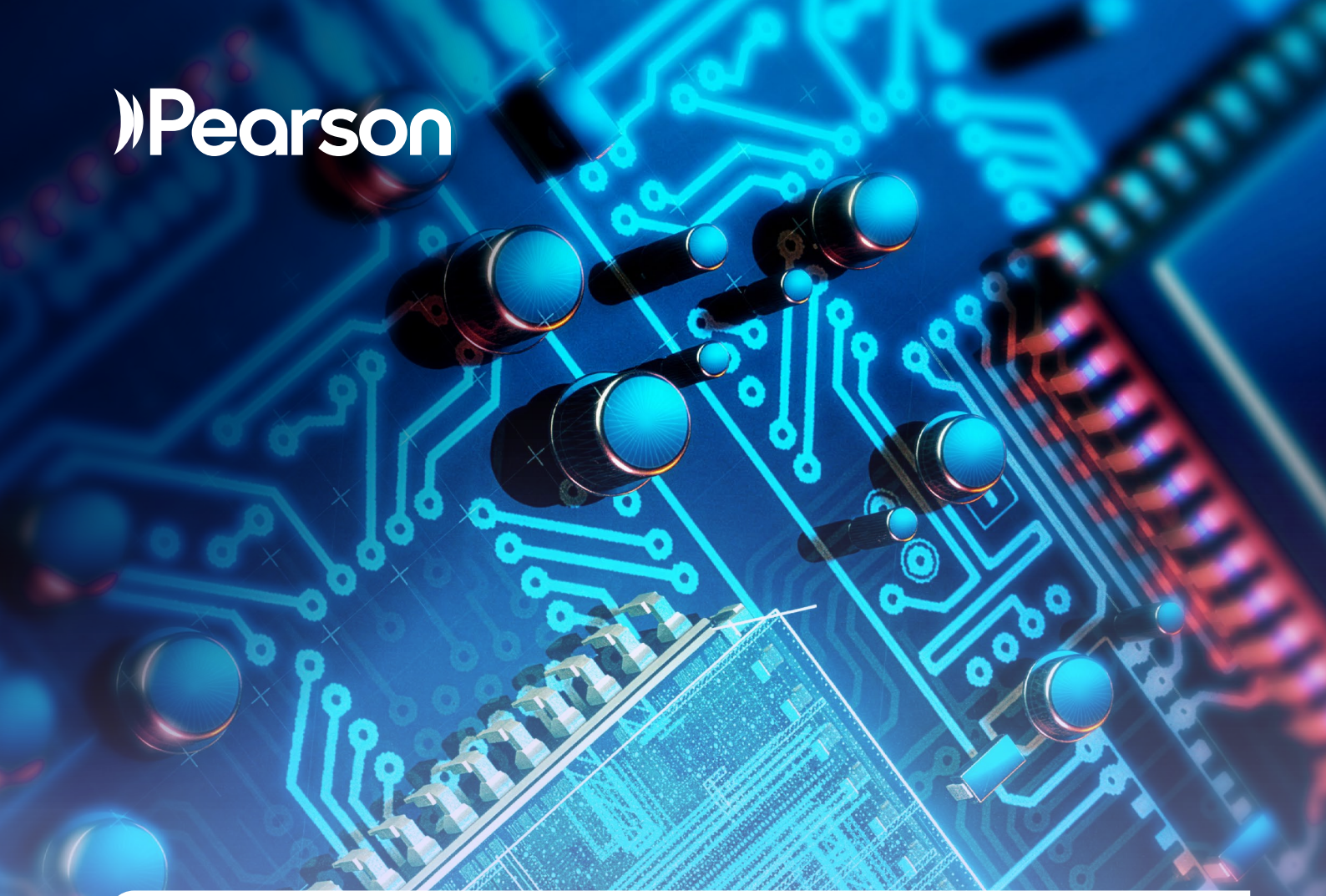


International Advanced Level

Computer Science

Getting Started Guide

Pearson Edexcel International Advanced Subsidiary in Computer Science (XCP01)
Pearson Edexcel International Advanced Level in Computer Science (YCP01)

First teaching September 2026 | First examination June 2027

First certification from June 2027 (International Advanced Subsidiary)
and August 2028 (International Advanced Level)

Issue 1

Getting Started Guide

1. Introduction

This Getting Started Guide provides an overview of the structure, content and assessment of our new Pearson International Advanced Subsidiary (IAS) and International Advanced Level (IAL) Computer Science qualifications

You should refer to the specification and sample assessment materials for more in-depth information about requirements.

2. Key features

Up-to-date and engaging content endorsed by industry and higher education experts, that develops both theoretical knowledge and practical programming, providing a solid foundation for further study of computer science and progression into employment.

A straight-forward modular structure comprising two units for the IAS and four for the IAL – the two IAS units plus two additional IA2 units.

Examination-only assessment consisting of four equally weighted papers – one for each unit – set and marked by Pearson Edexcel, with two exam series a year.

Python 3 used as a user-friendly introduction to programming, with a clearly defined program language subset that sets out the specific elements of Python learners need to know and be able to use.

No need to learn pseudocode, freeing up time for learners to engage with a real programming language.

A choice of on-screen or paper answer booklet for the two theory papers, plus an **on-screen exam** for the two programming papers in which learners use Python and an Integrated Development Environment (IDE) of their choice to demonstrate their ability to write, debug and refine code.

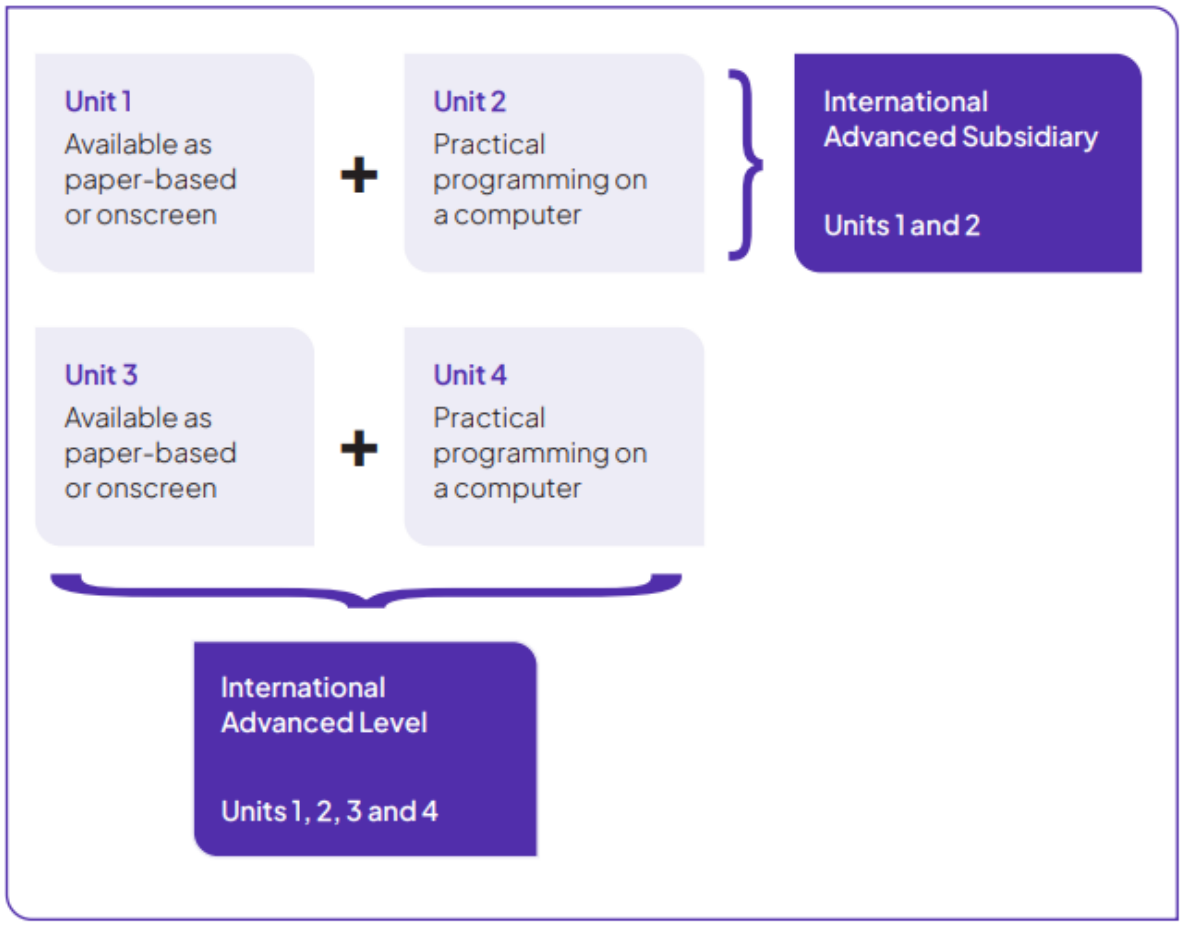
Comprehensive support – helping teachers to plan and implement the course successfully.

3. Specification at a glance

The discipline of computer science encompasses a broad range of topics, some theoretical and some practical, that explore how computer systems work and how they can be programmed to solve problems. This is reflected in the design of the specification, with units organised into two distinct strands.

The theory strand (Units 1 and 3) covers the core concepts and principles of computer science, whilst the aim of the practical strand (Units 2 and 4) is to develop programming and problem-solving skills.

The knowledge and understanding of programming taught in the practical strand, is supported by some of the content of the theory strand.



There are four-equally sized units, two pitched at IAS level and two at IA2 level. Whilst each unit covers a defined area of content, there are many connections across units. For example, in Unit 1 learners are introduced to the arrangement of data elements in a stack, while in Unit 2 they learn how to implement a stack using a fixed-length list.

The knowledge and skills gained in the two IAS units provide a solid foundation for progression to the advanced IA2 units. For example, in Unit 3 learners compare and contrast the Harvard machine architecture with the classic von Neumann architecture they learnt about in Unit 1, and explore how modern computers use pipelining to overcome the bottlenecks of sequential processing.

Qualification overview

The IAS and the IAL in Computer Science are unitised qualifications, enabling learners to build knowledge, understanding and skills cumulatively, and take exams when they are ready.

Learners can sit unit assessments in any of the examination series in which they are offered. They receive a uniform mark between 0 and 100 for each unit assessment. Resits of unit assessments are permitted, with the better of the two most recent attempts contributing to the overall qualification grade.

For more details see pages 87 of the specification.

IAS
• Stand-alone qualification, or first half of the IAL • Two mandatory units (Unit 1 and Unit 2) • 180 GLH • Typically taught over one year • Two externally set and marked examinations – one for each unit • Two exam series a year (January and June) • Graded on a five-grade scale from A to E (with A highest and E lowest)
IAL
• Four mandatory units – the two IAS units, plus two further IA2 units (Unit 3 and Unit 4) • 360 GLH • Typically taught over two years, with two units studied in each year • Four externally set and marked examinations – one for each unit • Two exam series a year (January and June) • Graded on a six-point scale A* to E (with A* highest and E lowest)

Prior learning

There are no prior learning requirements. Learners do not need to have studied computer science at GCSE. However, they do need to have a strong interest in the subject and, preferably, some previous experience of coding.

Computer-related mathematics

Mathematics is a key component of computer science. Learners need a good understanding of basic mathematical concepts and must be able to apply relevant maths skills in order to engage fully with the content of each of the four units.

4. Unit overview

UNIT 1

Principles of Computer Science

In this unit learners are introduced to core concepts and principles fundamental to a basic understanding of computer science, including components of computers systems, data representation, networks and encryption, programming paradigms and emerging technologies.

They learn different ways in which data can be structured and manipulated, including relational databases, 1- and 2-dimensional data structures, and abstract data types.

The algorithms and problem-solving techniques taught in this unit underpin the practical programming taught in Unit 2.

UNIT 2

Practical Programming and Problem-solving

Python is used in this unit to introduce learners to core programming constructs and components, including flow control, data types and structures, functions and recursion.

They learn how to write program code to handle strings, numeric data, files, and SQLite databases.

They are introduced to the concepts of abstraction and modularity and find out what constitutes best practice for debugging and writing clean code.

This unit lays the foundation for further exploration of programming and problem-solving in Unit 4.

UNIT 3

Advanced Principles of Computer Science

This unit reinforces and builds upon the foundational concepts and principles of computer science covered in Unit 1. It includes several more advanced topics, including embedded systems, the internet of things (IoT), cloud computing, neural networks, and assembly language programming.

To support the practical work, they undertake in Unit 4, learners are introduced to several more complex ADTs, algorithms, such as Dijkstra' shortest path, designed to solve more challenging problems, and features of the OOP and functional programming paradigms.

UNIT 4

Advanced Practical Programming and Problem-solving

This unit builds on the programming skills and techniques developed in Unit 2, enabling learners to tackle more demanding problems, including how to represent and handle complex ADTs, write regular expressions and develop solutions to more complex problems.

Learners use the NumPy and Pandas library modules for data analysis.

They gain hands-on experience of writing object-oriented and functional code in Python.

5. Subject content

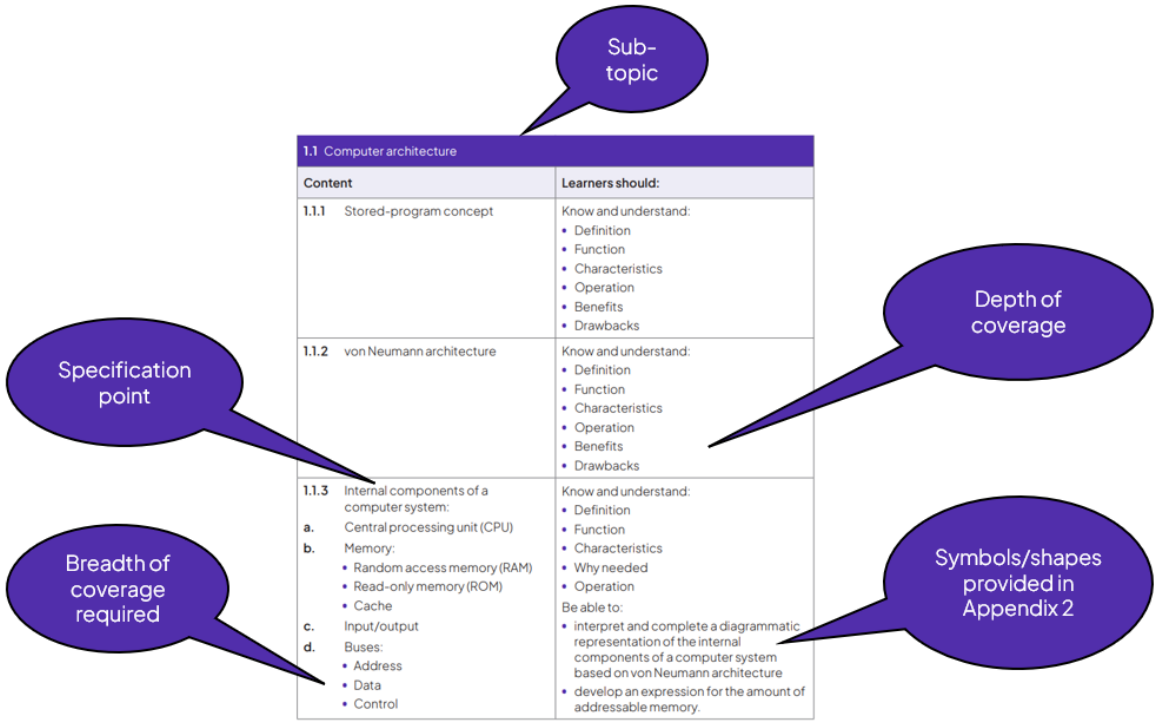
How to read a unit

The subject content of each unit is broken down into several topics. Each topic comprises several smaller, more manageable sub-topics. Each sub-topic consists of several numbered specification points (SPs), some of which are further broken down into lettered sub-bullets.

The subject content is presented in a two column format.

Lefthand column

The bulleted list in the lefthand column indicates the breadth of coverage required. The one in the righthand column specifies what learners need to know and understand about the item(s) in the lefthand column.

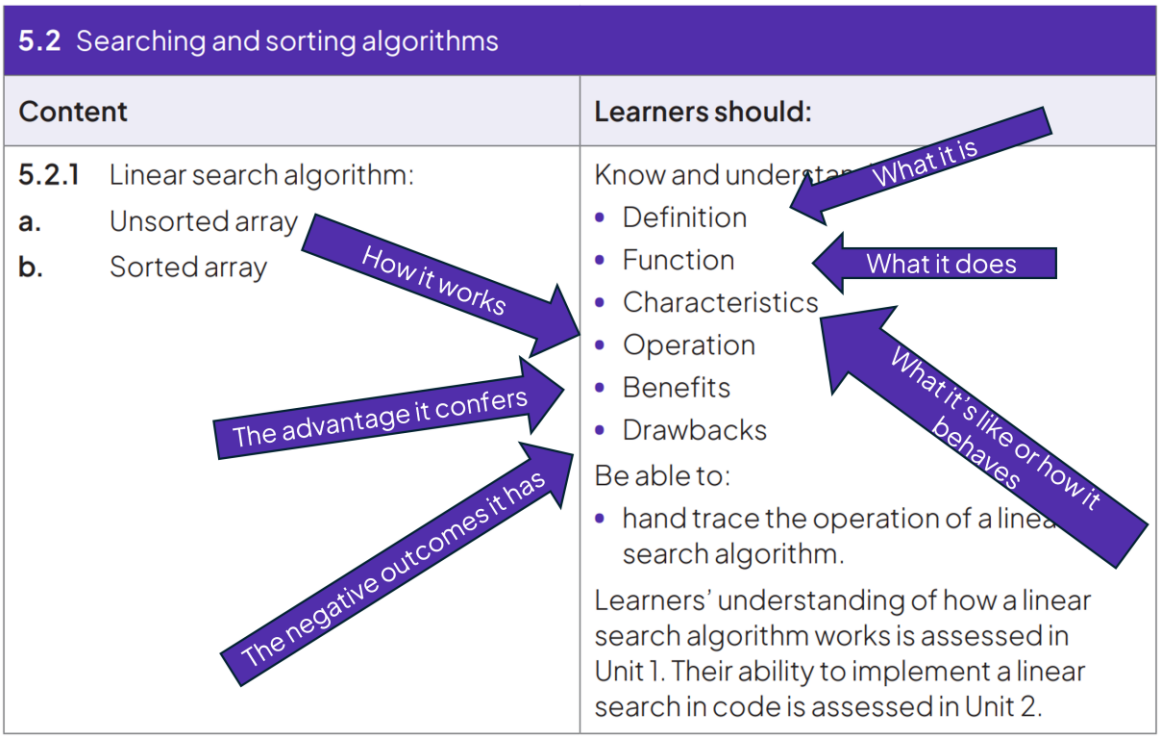


In this example, SP 1.1.3 lists the four components of a computer system that learners need to know. Sub-bullets b. and d. are further broken down, so as to clarify the breadth of coverage required.

As a minimum, all the numbered SPs (including any sub-bullets) in the content must be taught.

Righthand column

The righthand column clarifies the depth of coverage required.



In this example, SP 5.2.1 specifies that the linear search algorithm is required content. The righthand column spells out exactly what learners need to know and understand about the algorithm and what they need to be able to do.

SP 5.2.1: Linear search algorithm	
Definition	A sequential algorithm used to find the position of an item in an array
Function	To find the position of a specified target value within an array of elements
Characteristics	– Uses a loop to iterate through the array, and a conditional statement to compare the item in the current position with the target value – An in-place algorithm – no additional memory needed – A brute-force method – Performance directly proportional to the size of the array – Best case = target value is the first element in the array, and worst case = target value is the last element in the array or isn't present in the array

SP 5.2.1: Linear search algorithm	
Operation	<u>On an unsorted array</u> Starting with the first element, it iterates over all the elements in the array, comparing each one in turn with the target value, until either a match is found or the end of the array is reached. If a match is found, the search stops, and the index of the matching element is returned. If no match is found after examining all the elements, a -1 is returned. <u>Sorted list</u> As above, but the search stops once the value of the element in the current index position is greater than the target value.
Benefits	– Simple to implement – Works on sorted and unsorted arrays – Works with both numeric and non-numeric data – Works well if the dataset is small
Drawbacks	Not suitable for large arrays

Unit 1: Principles of Computer Science

Level: IAS	**Strand:** Theory	**Specification:** pages 13 – 36

Overview

The aim of this unit is to give learners a thorough grounding in the foundational principles of computer science, and to provide the underpinning theory for the practical programming undertaken in Unit 2.

Topic 1 deals with how the hardware components of a computer system are organised and work together, and how a multitasking operating system utilises process scheduling and memory management techniques to share limited system resources between competing processes.

Topic 2 looks at how numbers and text are represented in binary, and how binary arithmetic is performed.

Topic 3 focuses on the components of computer network, network metrics, and methods of encrypting data.

Topic 4 examines different ways of structuring and handing data.

Topic 5 covers various aids to problem-solving, including algorithms, trace tables, and programming paradigms.

Contemporary technologies, including Big Data, machine learning, and large language models are the focus of sub-topics 6.1 and 6.2, whilst sub-topic 6.3 covers a number of software development tools, some of which learners will gain firsthand experience of using in Unit 2.

Unit 1: Principles of Computer Science	
Topics	
Topic 1: Computer systems 1.1 Computer architecture 1.2 The operating system (OS) **Topic 2: Data representation** 2.1 Numbers 2.2 Binary arithmetic 2.3 Text **Topic 3: Networks and encryption** 3.1 Network fundamentals 3.2 Encryption	**Topic 4: Structuring data** 4.1 Relational databases 4.2 Data structures 4.3 Abstract data types (ADTs) **Topic 5: Problem solving** 5.1 Tools and techniques 5.2 Searching & sorting algorithms 5.3 Boolean logic 5.4 Programming paradigms **Topic 6: Enabling technologies** 6.1 Data science 6.2 Artificial intelligence (AI) 6.3 Tools

Unit 1: Principles of Computer Science

Computer-related mathematics

- Working with and converting between number bases and units of measurement
- Interpreting and writing expressions using units of measurement
- Representing signed and fractional numbers in binary
- Performing binary arithmetic and bitwise manipulation
- Constructing expressions involving file size, transfer rate and time
- Performing operations on sets
- Interpreting and writing Boolean expressions
- Interpreting and constructing truth tables

Specialist symbols & notation

- Components of the CPU
- Register transfer operations
- ERDs
- Boolean algebra (engineering notation)
- Flowcharts

(see Appendix 2)

- Program code for expressing algorithms (covered in Unit 2)

Assessment

- Externally set and marked
- Choice of written or onscreen examination
- 1 hour and 30 minutes
- Four sections, each worth 20 marks
- 80 marks in total
- Candidates must attempt all questions
- 12 – 14 marks targeting computer-related mathematics
- Use of a calculator is not permitted
- Available in January and June, first assessment June 2027
- Contributes 50% to the overall qualification grade for IAS, and 25% for IAL

Unit 2: Practical Programming and Problem-solving		
Level: IAS	**Strand:** Programming	**Specification:** pages 37 – 47

Overview

This unit introduces learners to fundamental programming concepts, and equips them with the skills they need to design, write, test and refine code.

The theory covered in Topics 4 and 5 of Unit 1 provides a solid foundation for the programming undertaken in this unit.

Topic 7 covers standard programming constructs and components, i.e. flow control (sequence, selection, iteration and exception handling), variables and constants, operators, and subprograms.

Topic 8 deals with data types and structures, along with ways of handling data, including methods to represent and handle stacks and queues implemented as fixed-length lists. Sub-topic 8.3 focuses on use of SQLite in Python to create, query and manage databases.

Best practice techniques for designing effective solutions, writing clean code, and ensuring solutions are fit for purpose and produce accurate results are covered in Topic 9.

Sub-topic 10.1 identifies the computational thinking techniques learners need to use when tackling problems.

The search and sort algorithms covered in sub-topic 5.2 in Unit 1, map to those listed in sub-topic 10.3.

Unit 2: Practical Programming and Problem-solving	
Topics	
Topic 7: Programming 7.1 Constructs **Topic 8: Organising and handling data** 8.1 Data types and structures 8.2 Data handling methods 8.3 Relational databases	**Topic 9: Best Practice** 9.1 Program design 9.2 Clean code 9.3 Functionality **Topic 10: Computational thinking and algorithms** 10.1 Computational thinking 10.2 Implementing algorithms 10.3 Search and sort algorithms 10.4 Recursion

Unit 2: Practical Programming and Problem-solving

Computer-related mathematics

- Writing expressions that use arithmetic, relational, Boolean and bitwise operators
- Handling numeric data, e.g. randomisation, rounding, truncating, type conversion
- Performing set operations
- Using SQL aggregation functions, e.g. AVG, COUNT, MAX, MIN, SUM
- Converting mathematical formulas into code
- Using methods from the Math and Random libraries
- Writing recursive functions

Specialist symbols & notation

- ERDs
- Flowcharts

} (see Appendix 2)

Assessment

- Externally set and marked
- Onscreen examination
- 3 hours
- Five questions, each requiring candidates to write code in Python using an integrated development environment (IDE) of their choice
- 80 marks in total
- Candidates must attempt all questions
- 4 – 6 marks target computer-related mathematics
- Digital code files and a digital copy of the PLS are provided in each candidate's user area
- Available in January and June, first assessment June 2027
- Contributes 50% to the overall qualification grade for IAS, and 25% for IAL

Unit 3: Advanced Principles of Computer Science

Level: IA2	**Strand:** Theory	**Specification:** pages 48– 67

Overview

This unit picks up where Unit 1 leaves off, broadening and deepening learners' knowledge and technical understanding and underpinning the practical work undertaken in Unit 4.

Topic 11 focuses on enhancements to the von Neumann architecture, and introduces learners to the concept of an embedded system. Sub-topic 11.2 looks at how the OS manages devices, virtualisation, and virtual machines.

Topic 12 hones in on data transmission, the RSA encryption, and error detection mechanisms. It covers internet-dependent technologies, including the IoT and edge computing.

As well as studying OOP and functional programming paradigms in Topic 13, learners write simple programs in assembly language.

More complex ADTs, including linked lists, hash tables and trees, are explored in Topic 14.

Topic 15 delves further into problem-solving techniques and algorithms, including use of Big O notation to describe the efficiency of algorithms.

Topic 16 looks at a range of emerging technologies, such as blockchain, quantum computing and deep learning neural networks, and considers what constitutes responsible practice for computer scientists.

Unit 3: Advanced Principles of Computer Science	
Topics	
Topic 11: Computer systems and data representation 11.1 Computer architecture 11.2 The operating system (OS) 11.3 Data representation **Topic 12: Networks and cybersecurity** 12.1 Networking 12.2 The Internet of Things (IoT) 12.3 Cybersecurity 12.4 Computing paradigms **Topic 13: Programming languages** 13.1 Types of programming languages	**Topic 14: Structuring data** 14.1 Abstract data types (ADTs) **Topic 15: Problem solving** 15.1 Algorithm design 15.2 Algorithms 15.3 Algorithmic efficiency 15.4 Boolean logic **Topic 16: Emerging technologies and professional practice** 16.1 Emerging technologies 16.2 Professionalism

Unit 3: Advanced Principles of Computer Science

Computer-related mathematics

- Using floating point representation
- Converting fractional numbers into binary and vice versa
- Calculating parity bits and checksums
- Using higher-order functions in calculations, e.g. evaluate fold, map, filter
- Calculating hash values
- Determining Big O complexity
- Writing assembly instructions for shifts / bit mask operations
- Using Boolean algebra and Karnaugh maps

Specialist symbols & notation

- CPU components
- Register transfer operations
- Boolean algebra (engineering notation)
- Flowcharts
- UML class

} (see Appendix 2)

- Assembly language instruction set (see Appendix 3)

- Program code for expressing algorithms (see Unit 2)

Assessment

- Externally set and marked
- Choice of written or onscreen examination
- 1 hour and 30 minutes
- Four sections, each worth 20 marks
- 80 marks in total
- Candidates must attempt all questions
- 8 – 10 marks target computer-related mathematics
- Assembly language instruction set available to candidates during the exam
- Use of a calculator is not permitted
- Available January and June, first assessment June 2028
- Contributes 25% to the overall qualification grade for IAL

Unit 4: Advanced Practical Programming and Problem-solving

Level: IA2	**Strand:** Programming	**Specification:** pages 68 – 78

Overview

This unit picks up where Unit 2 leaves off, further developing learners' programming and problem-solving skills.

Some of the theory content covered in Unit 3 supports the programming undertaken in this unit.

Topics 17 and 19 recap on the programming concepts, computational thinking skills, and best practice techniques introduced in Unit 2. In this unit, learners must utilise them when tackling more complex tasks and using different programming paradigms.

Topic 18 focuses on methods of handling different types of data, including floating-point numbers, regular expressions, numerical arrays (using methods from the NumPy library module), data analysis (using methods from the Pandas library module), and complex ADTs.

Topic 20 covers methods for OOP and functional programming, giving learners hands-on experience of these paradigms.

Algorithms requiring a significantly higher degree of technical understanding are covered in Topic 21.

Unit 4: Advanced Practical Programming and Problem-solving

Topics

Topic 17: Programming

17.1 Fundamental concept

17.2 Creating solutions

Topic 18: Representing and handling data

18.1 Data handling methods

18.2 Methods to represent and handle ADTs

Topic 19: Best practice

19.1 Program design

19.2 Clean code

19.3 Functionality

Topic 20: Additional programming paradigms

20.1 Object-oriented programming (OOP) and functional programming

Topic 21: Algorithms

21.1: Sorting algorithms

21.2: Shortest path and compression

Unit 4: Advanced Practical Programming and Problem-solving

Computer-related mathematics

- Using arithmetic, relational, Boolean and bitwise operators (in combination)
- Coding set operations, as part of data structures
- Manipulating decimal numbers
- Generating regular expressions
- Using methods from the NumPy and Pandas libraries
- Using hash maps and calculating hash values
- Functional programming, including higher-order functions, lambda expressions, recursion
- Converting mathematical formulas into code
- Using methods from the Math and Random libraries
- Writing recursive functions

Diagrams, symbols & notation

- Flowcharts

(see Appendix 2)

Assessment

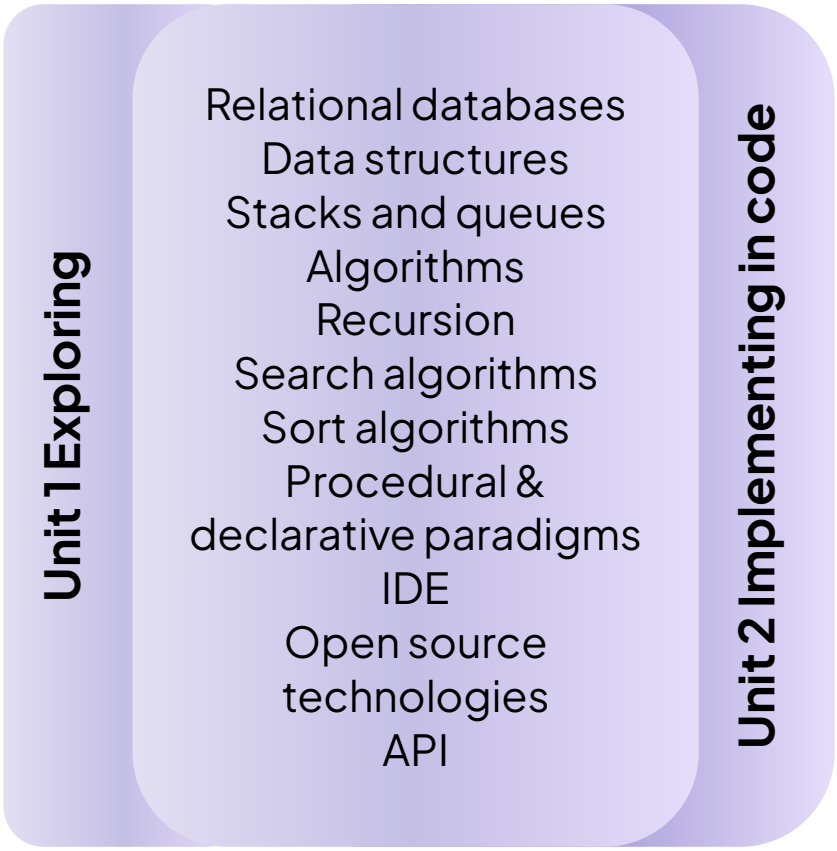
- Externally set and marked
- Onscreen examination
- 3 hours
- Five questions, each requiring candidates to write code in Python using an integrated development environment (IDE) of their choice
- 80 marks in total
- Candidates must attempt all questions
- 8 – 10 marks target computer-related mathematics
- Digital code files and a digital copy of the PLS are provided in each candidate's user area
- Available in January and June, first assessment June 2028
- Contributes 25% to the overall qualification grade for IAL

6. Connections between units

Whilst each unit has its own defined area of content, there are a number of cross-unit connections, both across strands and within strands. These are designed to give learners a deeper understanding and a better grasp of concepts, and help them see how what they are learning fits together.

Across strands

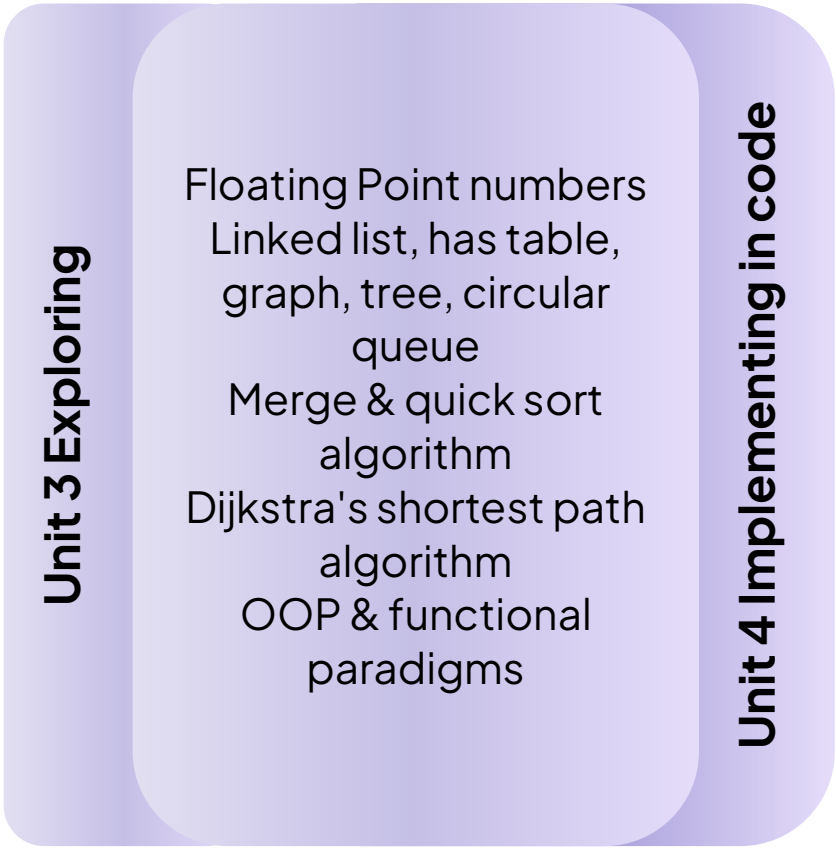
Learners apply the theory they learn in Unit 1 to the practical work they undertake in Unit 2.



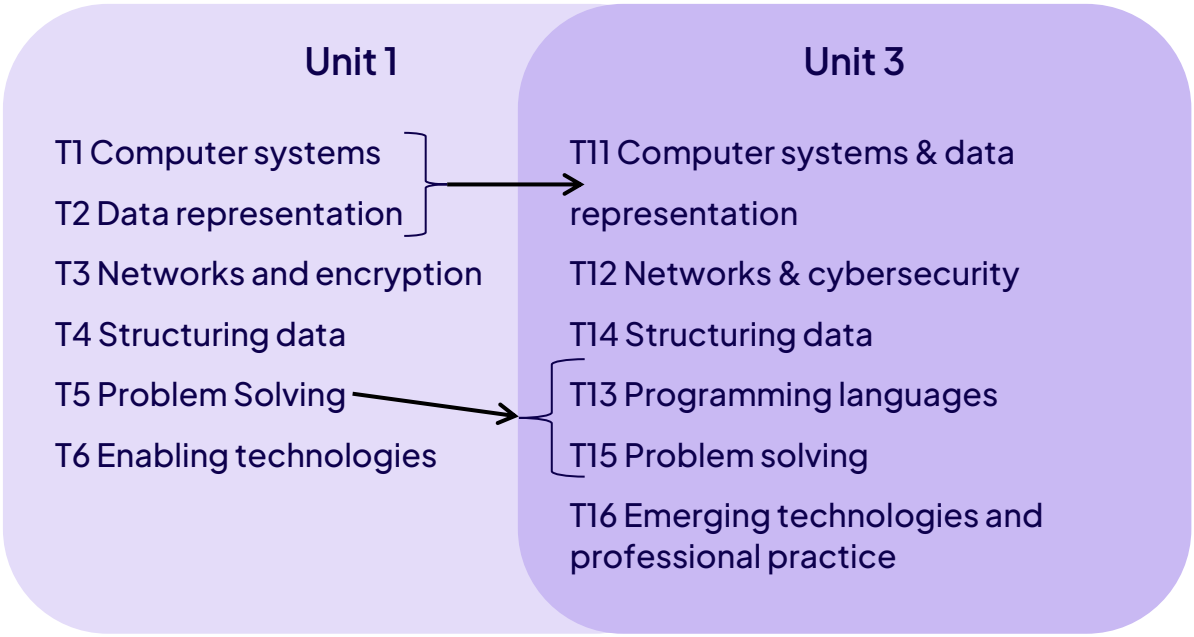
For example, in Topic 4, learners study relational databases, data structures and abstract data types, and in Topic 5 they learn about algorithms. In Unit 2, they learn methods to implement these data structures in code.

The Python language learnt in Unit 2 is used to express algorithms in Unit 1.

Units 3 and 4 work together in a similar way. For example, in Topic 13 learners study the OOP and functional programming paradigms, and in Unit 4 they use Python to write object-oriented and functional code.

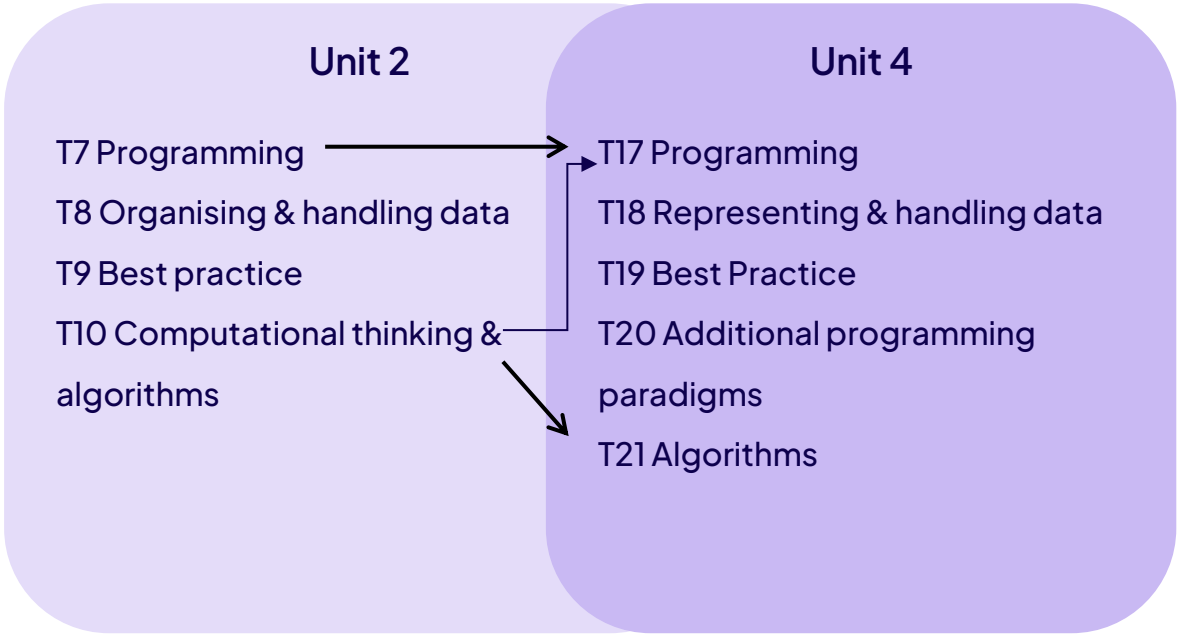


Within strands



Unit 3 broadens and deepens the knowledge and technical understanding learners acquire from studying Unit 1.

Similarly, Unit 4 reinforces and enhances the problem-solving and programming skills learners acquire in Unit 2.



7. The Programming Language Subset (PLS)

The Programming Language Subset (PLS) specifies a core set of Python 3 programming constructs that learners must be familiar with and able to use. These constructs are found in most high-level programming languages and form a good foundation for progression.

While there is nothing to prevent learners utilising more complex constructs or approaches not included in the PLS, there is no need for them to do so.

There is a PLS for Unit 2 (PLS2) and a separate one for Unit 4 (PLS4). With a couple of exceptions, notably SQLite and turtle graphics, PLS4 includes all of the PLS2 content, as well as additional content applicable to Unit 4.

The PLS is intended to provide clarity and focus for both teaching and assessment, supporting the consistent development of learners' programming skills. Both teachers and learners should familiarise themselves with its content.

Candidates are provided with an electronic copy of the PLS to refer to when sitting the practical examinations.

All problems set in the in the Unit 2 and Unit 4 examinations can be solved using only the functionalities presented in the relevant PLS. Any candidate successfully using more esoteric or complex constructs, or approaches not included in the PLS will still be awarded marks providing the solution is valid.

The PLS will remain valid for the lifetime of the qualification and available to download from the qualification page of the Pearson Edexcel website.

See page 81 of the specification for further details.

The Good Programming Practice Guide (GPPG)

The GPPG complements and supplements the content of the PLS, providing more in-depth information, and a wide range of examples.

A single GPPG is provided covering both Units 2 and 4.

The GPPG is available to download from the qualification page of the Pearson Edexcel website.

While candidates will have a copy of the relevant PLS to refer during the external assessment, the GPPG must not be used during the examination.

8. Assessment objectives

Assessment objectives (AOs) define the knowledge, understanding and skills that learners must demonstrate in the examinations.

There are three AOs for the IAS and IAL qualifications.

AO1:	Demonstrate knowledge and understanding of the key principles and concepts of computer science, including to solve programming problems
AO2:	Apply knowledge and understanding of key principles and concepts of computer science
AO3:	Design, program, test and refine programmed solutions using different types of programming paradigms

Each AO is further split into parts, measuring different facets of a learner's performance.

Breakdown of assessment objectives across units

	Descriptor	Weighting			
		Unit 1	**Unit 2**	**Unit 3**	**Unit 4**
AO1	(a) Demonstrate knowledge of the key principles and concepts of computer science	25%		20%	
	(b) Demonstrate understanding of the key principles and concepts of computer science	30%		35%	
AO2	(a) Apply knowledge and understanding of key principles and concepts of computer science	45%		45%	
	(b) Apply knowledge and understanding to solve specific programming problems		40%		30%
A03	(a) Design solutions		20%		27.5%
	(b) Program solutions		25%		27.5%
	(c) Test and refine solutions		15%		15%

Key features

AO1(a)
• Assesses recall of facts, definitions and terminology – the 'what' rather than the 'why or 'how' • Typical command words: 'Define', 'Give', 'State', 'Which' • Only tested in the Unit 1 and Unit 3 examinations
AO1(b)
• Assesses understanding of the principles of computer science • Requires a deeper grasp of a topic • Typical command words: 'Describe', 'Explain', 'Outline' • Only tested in the Unit 1 and Unit 3 examinations
AO2(a)
• Assesses application of knowledge and understanding of the principles of computer science to a particular scenario or problem • Requires a sustained and logical chain of reasoning • Typical command words: 'Calculate', 'Complete', 'Construct', 'Convert', 'Describe', 'Explain', 'Outline' • Associated with practical activities, such as working with algorithms, performing calculations, writing expressions • Use of computer-related mathematics maps to AO2(a) • Only tested in the Unit 1 and Unit 3 examinations
AO2(b)
• Assesses ability to complete straight-forward, directed programming tasks • Command word: 'Amend' • Only tested in the Unit 2 and Unit 4 examinations
AO3(a)
• Assesses ability to make own design decisions, concerning which constructs and components to use in code, how to minimise side effects etc. • Command words: 'Amend', 'Write' • Only tested in the Unit 2 and Unit 4 examinations

AO3(b)
• Assesses ability to write code to solve more complex problems with little or no direction • Command words: 'Amend', 'Write' • Only tested in the Unit 2 and Unit 4 examinations
AO3(c)
• Assesses ability to test, debug and refine code in order to ensure it is fit for purpose • Command words: 'Amend', 'Write' • Only tested in the Unit 2 and Unit 4 examinations

9. Papers and mark schemes

The examination papers are designed to be as accessible as possible to allow learners to demonstrate what they know, understand and are able to do.

The papers have a consistent format and structure, so that teachers and learners always know what to expect.

Command words

The command word taxonomy in Appendix 5 of the specification (page 100) lists the permitted command words and their meaning.

Command words always appear at the start of a question. Candidates should tailor their response to meet the requirements of the command word used.

The sample assessment materials (SAMs)

The SAMs are available on the qualification section of the Pearson website. They illustrate the format and style the exams and the types of questions learners are likely to encounter, and are a valuable source of reference.

Key features of the examination papers for Units 1 and 3

- Learners can sit the exams on screen or on paper.
- The same questions are answered by learners regardless of which mode of assessment they choose.
- Both exams are 1.5 hours long.
- Each paper is divided into four sections, each worth 20 marks.
- Questions within a section are ramped, meaning that, as a candidate progresses through a section, the questions become more challenging.
- A number of question types are employed.
  - **Multiple choice**, using the command word 'Which'. For example, SAM1 Q1, which requires candidates to select two correct statements from five, and SAM3 Q9, which requires them to select the statement that best defines deep learning.
  - **Short-open**, typically use the command words 'Define', 'Give', or 'State', and require only a short response. For example, SAM1 Q3, which asks for a definition of the term, 'asymmetric encryption', and SAM3 Q5(a), which asks what RISC stands for.
  - **Medium-open**, typically use the command words 'Describe', 'Explain' or 'Outline, and require a longer written response. For example, SAM1

Q9, which asks candidates to describe the process of searching for a key-value pair in a dictionary, and SAM3 Q20, which asks for an explanation of how the source IP address in a packet header can be used.

  - **Calculations, tabular and diagrammatic items**, typically use the command words 'Calculate', 'Complete', 'Construct' or 'Convert'. For example, SAM1 Q26, which requires candidates to complete a trace table for a supplied binary search algorithm, and SAM4 Q14, which asks them to complete a class diagram.

- There are **two six-mark questions** requiring a more complex response. For example, SAM1 Q15, which asks candidates to perform a calculation, and SAM4 Q31, which asks them to use Dijkstra's shortest path algorithm to complete a table.
- Some questions may require candidates to read Python code. For example, SAM1 Q25, requires them to explain the type of error that could occur in the supplied algorithm.

Key features of the examination papers for Units 2 and 4

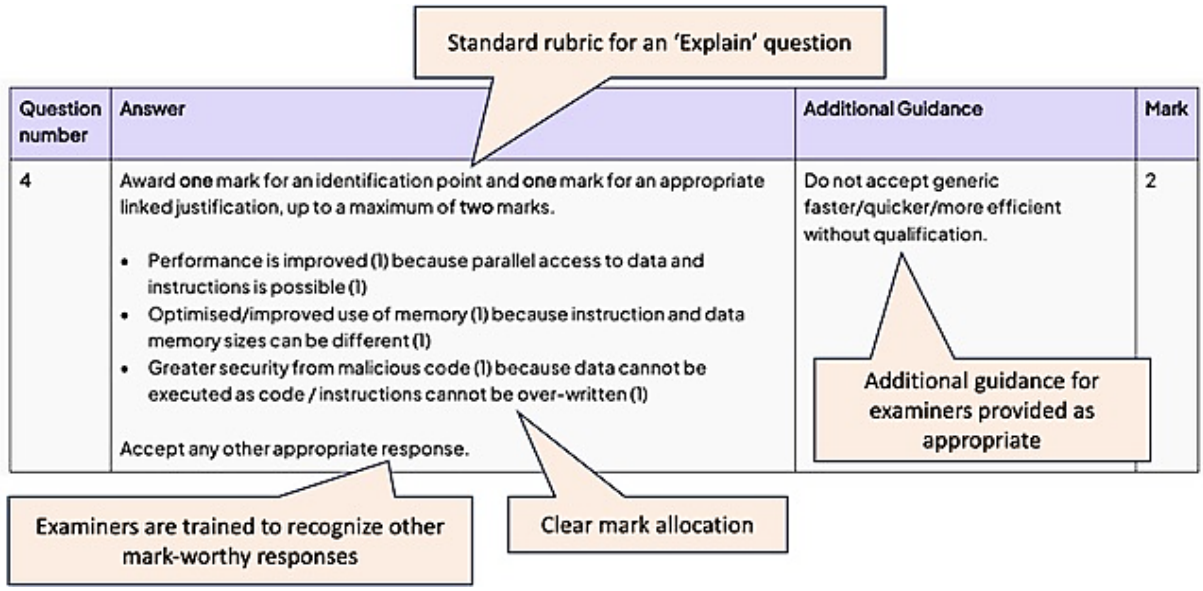
- Candidates sit the paper on screen. They carry out a series of programming tasks using Python 2 and an IDE of their choice. They are provided with a paper copy of the paper, a set of program and data files, and a digital copy of the PLS.
- Both exams are three hours long.
- Internet access is not permitted.
- There are five questions in each paper.
- Papers are 'ramped', meaning that, as a candidate progresses through a paper, the questions become more challenging.
- A number of question types are employed.
  - **Fix the errors** items require candidates to fix the errors in supplied code. For example, SAM2 Q01 and SAM4 Q01.
  - **Flowchart** items require candidates to translate a supplied flowchart into code.
  - **Parsons problem** items require candidates to reorder the lines in supplied code to create a functional program. For example, SAM2 Q02.
  - **Complete the code** items require candidates to complete lines and add new lines to supplied code, with the logic for the problem solution provided in the comments. For example, SAM2 Q03 and SAM4 Q02.
  - **Comment-first coding** items require candidates to write code to implement a high-level design. For example, SAM2 Q04 and SAM4 Q03.
  - **Open-ended** items are always the final question on the paper, and require candidates to design, write and refine a solution, with little or no scaffolding provided. For example, SAM2 Q05 and SAM4 Q05.
- Questions may provide an 'aide memoir', but candidates are expected to be familiar with the algorithms taught in theory unit.
- A standard convention is used to identify instructions in supplied code, i.e. `#=====>`

Pearson Instructions for Conducting Examinations (ICOE)

Details concerning the administration of the practical programming examinations can be found in the Pearson Instructions for Conducting Onscreen Examination (ICOE) document, available to download on the Pearson Edexcel International Advanced Level in Computer Science qualifiations page on our website.

Mark schemes

Mark schemes are clear and easy to apply.



Examiner reports

An examiner report is produced following each examination series, providing detailed feedback on performance, identifying common mistakes and misconceptions and clarifying expectations.

10. Planning

The structure of these qualifications allows teachers to construct a course of study that can be taught and assessed as either:

- distinct modules of teaching and learning with related examinations taken at appropriate stages during the course
- or as a linear course assessed in its entirety at the end.

We envisage Units 1 and 2 being taught in parallel in Year 1, with Units 3 and 4 being taught in parallel in the second year of the course.

It is important to bear in mind that all four units are equally weighted, so the same amount of curriculum time should be allocated to each.

Resource requirements

In order to engage with the subject content of the specification, learners need access to:

- A computer with Internet access
- Productivity software (word processor, spreadsheet, PDF reader, etc.)
- Text file editor (Notepad++, etc.)
- One or more IDEs
- Language translator for Python 3
- A relational database with a programming interface (API), accessible from Python 3
- SQLite (*https://docs.python.org/3/library/sqlite3.html*)

Assembly language

In the interest of giving learners real-life experiences, we have opted to reference ARMLite, which is a cut-down version of the assembly language instruction set used for programming the ARM chips that are found in a multitude of different devices, including Raspberry Pi computers.

Teachers may choose to teach assembly language in a purely theoretical way, paper-based fashion, but can instead opt to use the ARMLite simulator to give learners a more realistic experience (see Appendix 3 of the specification, pages 96-97)

Python Environment

It is important that learners get lots of practice interacting with a programming environment to build confidence and skill. The programming environment they use, both at school and at home, must support the Python 3 language.

Any of the free, no cost, options are suitable.

If learners have access to computers at home, they can duplicate these environments for homework or practice.

Environments to consider include:

- Thonny: *https://thonny.org/*
- PyCharm: *https://www.jetbrains.com/pycharm/*
- PyScripter: *https://sourceforge.net/projects/pyscripter/*
- NetBeans: *https://netbeans.org/*
- Eclipse: *https://www.pydev.org/*
- Visual Studio: *https://visualstudio.microsoft.com/vs/features/python/*

For those learners not able to duplicate the in-school environment, there are suitable online Python environments that they can use. See, for example:

- *https://repl.it/*
- *https://create.withcode.uk/*

11. Support

Pearson is committed to supporting great computer science teaching. We have put together a comprehensive package of support to help you plan and implement the International Advanced level Computer Science. It includes:

- ✓ Two **dedicated student books** one for IAS and one for IA2, available as printed books and eBooks.
- ✓ An online **Teacher Resource Pack** containing material such as lesson plans, assessments and mark schemes, and further resources to support your planning and teaching and save you valuable time.
- ✓ An editable **course planner** and **schemes of work** that you can tailor to meet the needs of your department.
- ✓ **Sample assessment material** and additional materials, e.g., exemplars of marked students' work with examiner commentaries, to help you understand the format and level of demand of the exams and use for formative and mock assessments.
- ✓ **examWizard** which is an online data bank of past exam questions designed to support learners and teachers with exam preparation and assessment.
- ✓ Our **ResultsPlus** service provides detailed analysis of students' exam performance, enabling you to adjust your scheme of work to allocate more time to topics and skills that students struggle with. By analysing your results year on year in conjunction with students' scripts, you can improve how your department delivers the course.
- ✓ Our **Access to Scripts** service enables you to download individual student scripts instantly on results day allowing you to evaluate how your candidates performed on particular questions and identify skills gaps so you can tailor future teaching plans.
- ✓ Our **Subject advisory service** provides a direct and personal source of help and support. Sign up to receive regular subject adviser updates, which will keep you up to date with qualification and service news. Or contact us directly via our support portal.
- ✓ **Training events –** We host a series of training events on demand and live to help teachers deepen their understanding of our qualifications.
    - Getting Ready to Teach
    - Additional courses on assessment, delivery, and exam insights