



Curriculum Intent Overview

What pupils will know and be able to do by the end of KS3/KS4

By the end of KS3, pupils will have developed secure digital literacy, basic programming skills and an understanding of how computer systems are used in the real world. They will be able to use technology safely, responsibly and creatively. Pupils will understand key concepts such as algorithms, sequencing, selection, iteration, data representation, spreadsheets, databases, networks, cybersecurity, and the wider impact of technology on society.

By the end of KS4, pupils studying Computer Science will have developed a deeper understanding of computational thinking, programming, computer systems, networks, cybersecurity, data representation, ethical issues and algorithms. They will be able to design, write, test and refine programs using Python and apply their knowledge to exam-style scenarios.

The key knowledge and disciplinary skills

Key Knowledge:

- computer systems and hardware
- software and operating systems
- networks and cybersecurity
- algorithms and computational thinking
- programming concepts and techniques
- data representation

disciplinary skills:

- problem-solving
- decomposition
- abstraction
- algorithmic thinking
- logical reasoning
- programming and debugging
- testing and evaluation
- interpreting exam scenarios
- applying knowledge to unfamiliar problems
- communicating technical ideas clearly using accurate terminology

Recurring conceptual themes in Computing

Across KS3 and KS4, pupils revisit a set of recurring Computing concepts with increasing complexity. These are: computational thinking, programming, data and representation, computer systems, networks, cyber security, and the impact of technology.

In KS3, pupils meet these concepts through accessible and practical contexts, such as Scratch, spreadsheets, online safety, computer systems, Python, networks, app design, databases and AI. At GCSE, these same concepts are revisited with greater technical depth through computational thinking, data representation, computer systems, networks, cyber security, programming and issues and impact.

This ensures that pupils do not experience Computing as isolated topics, but as connected ideas that develop over time. For example, pupils move from using simple algorithms in Scratch, to writing Python programs, to tracing and refining GCSE algorithms. They move from basic online safety to cyber security threats and prevention, to network security. They move from using spreadsheets and databases to understanding how data is represented, stored and processed by computer systems.

How the curriculum builds cultural capital

The Computing curriculum builds cultural capital by helping pupils understand the digital world they live in and how technology affects society, employment, communication, privacy, security and the economy. Pupils learn about real-world issues such as cybercrime, artificial intelligence, digital footprints, online safety, automation, data protection, environmental impact and ethical use of technology.

The curriculum also exposes pupils to careers in computing and digital industries, including software development, cybersecurity, data science, artificial intelligence, networking, games development and IT support. Pupils develop transferable skills such as resilience, logical thinking, creativity and independence, which prepare them for future study and employment.

How it prepares pupils for next stages (GCSE/A-Level and beyond)

Across the curriculum, pupils revisit six recurring Computing concepts: computational thinking, programming and problem-solving, data and representation, computer systems, cyber security, and the impact of technology. These concepts are introduced in accessible KS3 contexts and revisited with increasing complexity at GCSE. For example, pupils move from Scratch and basic Python to GCSE programming; from simple online safety to cyber security and network protection; and from using spreadsheets and databases to understanding how data is represented, stored and processed by computer systems.

At KS4, pupils develop the knowledge and exam skills needed for GCSE Computer Science. They learn to apply theory to real-world contexts and develop programming skills that support further study at A Level, apprenticeships or digital qualifications

What a Grade 7 / B student looks like in your subject

A Grade 7 GCSE student can apply knowledge confidently to both familiar and unfamiliar scenarios. They use accurate technical terminology, explain concepts in detail and justify their answers. They can write, trace, debug and refine programs using sequencing, selection, iteration, lists and subroutines. They can explain algorithms, compare solutions, interpret pseudocode and apply computational thinking independently.

	HT1	HT2	HT3	HT4	HT5	HT6	Assessment Criteria
Year 7	Unit: Digital Literacy Foundations Knowledge: Pupils learn how to use the school network safely and independently, including logging on, saving work, using folders, accessing school platforms, and using email appropriately. They also learn about social networking, personal data, privacy, and how to keep information safe online. Skills: I can log on and access school systems independently. I can create folders and organise my files clearly. I can save, retrieve and rename work using suitable file names. I can use school email appropriately and follow safe routines when using technology. Prior/Future links: Builds on basic digital skills from primary school. Supports all future Computing units as pupils need to access, save and organise their work independently. Links to Year 7 Computer Safety and Crime and Year 8 Networks and Cyber Security.	Unit: Computer Safety and Crime Knowledge: Pupils learn about online safety, passwords, personal information, cyberbullying, digital footprint, phishing, malware, hacking and how to report concerns. They learn how to use technology safely, respectfully, responsibly and securely. Skills: I can identify common online risks and explain how they may affect users. I can recognise suspicious messages and explain what action should be taken. I can create strong passwords and describe how to protect personal information. I can explain how to use technology safely, respectfully and responsibly. Prior/Future links: Builds on online safety from primary school. Links to Year 8 Networks and Cyber Security and Year 9 Technology, AI and Society.	Unit: Scratch Knowledge: Pupils learn the basics of programming using a block-based language. They learn about sequencing, selection, iteration, variables, events, sprites, costumes and debugging. Skills: I can create simple algorithms using sequence, selection, and iteration. I can use blocks to control sprites, events, inputs and outputs. I can test my program to check if it works as intended. I can identify and fix simple errors in a Scratch program. Prior/Future links: Builds on computational thinking from primary school. Prepares pupils for Year 8 Python and Year 9 Computational Thinking.	Unit: Understanding Computers Knowledge: Pupils learn about the main hardware and software components of a computer system, including input devices, output devices, storage, CPU, RAM, operating systems and applications. They learn how components work together using input, process, output, and storage. Skills: I can identify examples of hardware, software, input devices, output devices, and storage devices. I can explain the role of key computer components, including the CPU, RAM, and storage. I can use the input-process-output model to describe how a computer system works. I can explain how data moves through a simple computer system. Prior/Future links: Builds on pupils' use of school computers in HT1. Links to Year 8 Data Representation and Networks and supports understanding of how programs run in Year 8 Python.	Unit: Spreadsheets Basics Knowledge: Pupils learn the purpose of spreadsheets and how data is organised using cells, rows, columns, cell references and worksheets. They learn basic formulas and functions such as SUM, AVERAGE, MIN, and MAX. Skills: I can enter and format data in a spreadsheet. I can use cell references, formulas and simple functions such as SUM, AVERAGE, MIN and MAX. I can create simple charts to present data clearly. I can check my spreadsheet for errors and improve its accuracy. Prior/Future links: Builds on digital organisation skills from HT1	Unit: Spreadsheets Project Knowledge: Pupils apply spreadsheet skills to a realistic scenario. They learn how spreadsheets can be used to organise, calculate, model and present data for a given purpose. Skills: I can select appropriate formulas and functions for a given scenario. I can create charts and format a spreadsheet so that it is clear and easy to understand. I can use spreadsheet data to answer questions and make decisions. I can improve my spreadsheet and explain the choices I have made. Prior/Future links: Builds directly on HT5 spreadsheet basics.	By the end of Year 7, pupils can • use school systems safely and organise files independently • explain common online safety risks and describe how to protect personal information • create a simple Scratch program using sequence, selection, iteration and events • identify key computer components and explain basic input, processing, output and storage • use spreadsheets to calculate, format and present data accurately

	Assessment: digital organisation task	Assessment: End of term assessment covering HT1 and HT2	Assessment: (Practical) Scratch project with success criteria	Assessment: End of term assessment covering HT3 and HT4	Assessment: Spreadsheet skills task using formulas, formatting and charts.	Assessment: Spreadsheet project and Summer GI	
Year 8	Unit: Computer Systems, Binary and Numbers Knowledge: Pupils learn the key elements of a computer system, including hardware, software, input, processing, output and storage. They learn the role of the CPU and how storage devices are used. Pupils also begin to understand why computers use binary and how numbers can be represented using 0s and 1s, including bits, bytes, binary place value and binary to denary conversion. Skills: I can identify and classify hardware and software. I can apply the input-process-output model to explain how computer systems work. I can describe the role of the CPU and compare different storage devices. I can convert binary numbers into denary and explain why computers use binary. Prior/Future links: Builds on Year 7 digital skills and introduces the computer systems knowledge this cohort missed previously. Provides the foundation for understanding how computers process and store data. Supports Year 9 Computational Thinking and GCSE Computer Science Paper 1 content.	Unit: Binary Addition, Boolean Logic and Data Representation Knowledge: Pupils develop their understanding of binary by learning how to carry out simple binary addition and recognise overflow. They are introduced to Boolean logic, including AND, OR and NOT, and learn how computers use logic to make decisions. Pupils also learn that different types of data, including text, images and sound, can be represented using binary. Skills: I can carry out simple binary additions and check my answers for accuracy. I can identify when overflow may occur. I can complete basic truth tables using AND, OR, and NOT. I can explain how text, images, and sound can be represented using binary. Prior/Future links: Builds directly on Autumn 1 binary knowledge and strengthens pupils' understanding of how computers represent and process data. Links to Year 9 Computational Thinking, programming conditions and GCSE topics including data representation and logic.	Unit: Python Programming: Basics Knowledge: Pupils learn how to use a textual programming language to solve simple problems. They learn about sequence, variables, data types, input, output, selection and basic error checking. Skills: I can write simple Python programs using print, input and variables. I can use appropriate data types and meaningful variable names. I can use selection to make decisions in a program. I can identify and correct simple syntax errors in my code. Prior/Future links: Builds on Year 7 Scratch by moving from block-based programming to a textual language	Unit: Python Programming: Problem Solving and Algorithms Knowledge: Pupils develop their Python programming by using iteration, lists, and simple functions. They also learn about key searching and sorting algorithms, including linear search, binary search, bubble sort and merge sort, and compare when different algorithms are most useful. Skills: I can write Python programs using loops, lists and simple functions. I can trace code to predict the output of a program. I can debug programs by identifying and correcting errors. I can describe how searching and sorting algorithms work and compare when they may be useful. Prior/Future links: Builds directly on HT3 Python basics. Links to Year 9 Computational Thinking	Unit: Networks: How Computers Communicate Knowledge: Pupils learn what a computer network is and why networks are used. They learn about LANs, WANs, the internet, the World Wide Web, clients, servers, routers, switches, wireless access points, packets and how devices communicate with other systems Skills: I can identify common network devices and explain their purposes. I can explain the difference between a LAN and a WAN. I can describe how data travels across a network. I can use diagrams to represent simple networks and explain the difference between the internet and the World Wide Web. Prior/Future links: Builds on Year 7 Understanding Computers. Links to HT6 Cyber Security and Year 9 Technology, AI and Society.	Unit: Cyber Security and Online Protection Knowledge: Pupils learn about common cyber security threats, including phishing, malware, social engineering, brute-force attacks, weak passwords and data breaches. They learn how individuals and organisations can protect systems using strong passwords, two-factor authentication, anti-malware, updates, access levels, and safe online behaviour. Skills: I can identify cyber security threats from different scenarios. I can recognise suspicious messages and explain how users should respond. I can recommend suitable methods to protect users and systems. I can explain how strong passwords, two-factor authentication, anti-malware, updates and access levels improve security Prior/Future links: Builds on HT5 Networks, links to Year 7 Computer Safety and Crime and Year 9 Technology, AI and Society.	By the end of Year 8, pupils can: • explain how computer systems use hardware, software, CPU, memory and storage • represent numbers using binary and carry out simple binary addition • describe how text, images and sound can be represented using binary • write simple Python programs using input, output, variables, selection and iteration • describe how networks allow devices to communicate • identify cyber security threats and recommend ways to stay secure online
	Assessment: Computer Systems, CPU, storage and binary conversion.	Assessment: End of term assessment covering HT1 and HT2	Assessment: Short practical Python tasks.	Assessment: End of term assessment covering HT3 and HT4	Assessment: Network scenario task	Assessment: Summer GI	
Year 9	Unit: Spreadsheet Modelling Knowledge: Pupils learn how spreadsheets can be used to model real-world situations and support decision-making. They build on basic spreadsheet knowledge by using formulas, functions, percentages, charts and “what if” changes to calculate and analyse data. Skills: I can create spreadsheet models for real-world scenarios. I can use	Unit: Computational Thinking Knowledge: Pupils learn how computational thinking is used to solve problems. They study logical thinking, logic gates, algorithmic thinking, abstraction and decomposition. They learn how problems can be broken down, simplified and represented using algorithms and diagrams. Skills:	Unit: App Design: Planning and User Needs Knowledge: Pupils learn how digital products are designed for a specific audience and purpose. They learn about user needs, success criteria, accessibility, layout, navigation, and how app ideas are planned before they are created. Pupils also learn flowchart symbols and how flowcharts can model user journeys.	Unit: App Design: Prototype, Testing and Evaluation Knowledge: Pupils learn how to create, test and improve a digital prototype. They understand how navigation, layout, consistency, accessibility and feedback affect the usability of a digital product. Skills: I can create a clickable app prototype using suitable software. I can add screens, navigation, and	Unit: Databases Knowledge: Pupils learn how databases are used to store, organise, search and retrieve data. They learn about tables, records, fields, data types, primary keys, queries, forms and reports. Pupils understand why databases are more suitable than simple documents for storing structured information.	Unit: Technology, AI and Society Knowledge: Pupils learn how modern technologies, including artificial intelligence, recommendation algorithms, chatbots and data-driven systems, affect individuals and society. They consider misinformation, deepfakes, privacy, bias, automation, digital wellbeing and responsible use of technology. Skills:	By the end of Year 9, pupils can • use spreadsheet models to analyse data, test “what if” scenarios and make recommendations • apply computational thinking, including logic, abstraction, decomposition and algorithms, to solve problems • design, prototype, test and evaluate a digital

	formulas, functions, percentages, charts and formatting to calculate and present data. I can test “what if” scenarios by changing values and comparing outcomes. I can use spreadsheet results to make justified recommendations. Prior/Future links: Builds on Year 7 spreadsheet basics and project work. Links to Year 9 Databases, where pupils continue to organise and analyse structured data. Supports future vocational IT and business-related digital skills.	I can use logical thinking to solve problems and explain decisions clearly. I can complete truth tables and interpret simple logic gates. I can design algorithms using sequence, instructions, loops, and nested loops. I can use abstractions to remove unnecessary details and represent problems using simplified diagrams or networks. Prior/Future links: Builds on Year 7 Scratch and Year 8 Python programming. Supports the App Design unit, where pupils use decomposition, flowcharts and logical thinking to plan a digital product.	Skills: I can analyse the audience and purpose of a digital product. I can identify user needs and create success criteria. I can use flowcharts to model user journeys. I can design wireframes and plan screens for an app prototype. Prior/Future links: Builds on Year 9 Computational Thinking by applying decomposition, sequencing and logic to a real design problem. Links to Year 8 programming and Year 7 Scratch using algorithms and user interaction.	consistent design features. I can test buttons and user journeys to check that the prototype works. I can collect feedback, make improvements and evaluate how well the prototype meets user needs. Prior/Future links: Builds directly on HT3 planning and design work. Links to Technology, AI and Society through consideration of responsible design, accessibility, user data and the impact of digital products on users.	Skills: I can create and use tables to store structured data. I can choose suitable field names and data types. I can sort, filter and search data using simple queries. I can use forms and reports and explain how databases are used in real-life situations. Prior/Future links: Builds on spreadsheet work from Year 7 and Year 9 by developing pupils’ understanding of structured data.	I can identify benefits and risks of modern technologies, including AI. I can explain how AI, recommendation algorithms and data-driven systems are used in everyday life. I can recognise misinformation and evaluate online information. I can discuss privacy, bias, automation and digital wellbeing using examples to justify my views. Prior/Future links: Builds on Year 7 online safety, Year 8 cyber security	app for a specific audience and purpose • create and query databases to store, organise and retrieve structured data • explain how technology and AI can affect individuals and society, using examples to justify their views
	Assessment: Spreadsheet modelling task	Assessment: End of term assessment covering HT1 and HT2	Assessment: App planning assessment	Assessment: Final App Design project.	Assessment: Practical database task	Assessment: Summer GI	
Year 10	Unit: Topic 1 Computational thinking + Topic 6 programming foundations Knowledge: Decomposition, abstraction, algorithms, pseudocode, flowcharts, sequence, input/output, variables and data types. Skills: I can use decomposition and abstraction to break problems down and focus on key information. I can follow and write simple algorithms using pseudocode and flowcharts. I can write basic Python programs using sequence, input, output, variables and data types. I can use meaningful identifiers and clear formatting to make programs easier to understand. Prior/Future links: Builds on KS3 Scratch/Python and prepares pupils for Paper 1 algorithm questions and Paper 2 coding tasks.	Unit: Topic 1 Algorithms and Logic + Topic 6 Programming Knowledge: Searching and sorting algorithms, trace tables, logic, Boolean expressions, selection and iteration. Skills: I can complete trace tables to follow how algorithms work. I can identify syntax, logic and runtime errors. I can use selection, iteration and logical operators in algorithms and Python programs. I can compare searching and sorting algorithms and explain how they work. Prior/Future links: Secures core computational thinking before applying the same logic to data and practical programming problems.	Unit: Topic 2 Data representation + Topic 6 data handling Knowledge: Binary, denary, hexadecimal, binary arithmetic, characters, images, sound and units of storage. Skills: I can convert between binary, denary and hexadecimal. I can perform binary arithmetic and explain overflow. I can explain how characters, images and sound are represented by computers. I can use strings, integers, real numbers and casting in Python programs. Prior/Future links: Links mathematical representation to how programs store, process and output data.	Unit: Topic 2 Data storage/compression + Topic 6 lists and validation Knowledge: Compression, storage calculations, data accuracy, validation and verification. Skills: I can calculate file sizes and data capacity requirements. I can explain the difference between lossy and lossless compression. I can use lists or arrays to store and process data in Python. I can use validation and verification to improve the accuracy and reliability of programs. Prior/Future links: Supports Paper 1 data questions and strengthens Paper 2 problem solving with stored data.	Unit: Topic 3 Computers: hardware and software + Topic 6 modular programming Knowledge: CPU, FDE cycle, memory, secondary storage, embedded systems, operating systems and utility software. Skills: I can explain how the CPU, memory and storage work together in a computer system. I can describe the fetch-decode-execute cycle using accurate terminology. I can compare storage devices and explain the role of operating systems and utility software. I can write and use subprograms or functions to structure programs more effectively. Prior/Future links: Connects system architecture to how programs are executed and managed.	Unit: Topic 3 Programming languages + Topic 6 Knowledge: High-level and low-level languages, translators, IDEs, errors, and maintainable code. Skills: I can compare high-level and low-level programming languages. I can explain the difference between compilers and interpreters. I can interpret, refine and debug Python code. I can use comments, testing and clear structure to make programs easier to read, maintain and improve. Prior/Future links: Prepares pupils for Year 11 networks, cyber security and full Paper 2 programming practice.	By the end of the year pupils can... • explain and apply core Paper 1 knowledge from computational thinking, data representation and computer systems • trace, write and refine algorithms using accurate computational thinking vocabulary • write Python programs using sequence, selection, iteration, input/output, variables, data types, lists and subprograms • use testing and debugging to identify errors and improve programming solutions • apply knowledge to GCSE-style scenarios using clear technical terminology
	Assessment: Baseline programming task, Topic 1 retrieval quiz and short Paper 1-style questions.	Assessment: Topic 1 end-of-topic assessment, algorithm tracing tasks and practical Python construct tasks.	Assessment: Topic 2 number systems/data representation assessment and Python data handling task.	Assessment: Data storage/compression assessment and programming task using lists and validation.	Assessment: Topic 3 hardware/software assessment and modular programming checkpoint.	Assessment: assessment covering Topics 1-3 and a Paper 2-style programming task.	Assessment: End of Year 10 grade indicator

Year 11	Unit: Topic 4 Networks + Topic 6 applied programming Knowledge: Network types, LAN/WAN, topologies, hardware, wired/wireless connections, protocols and layers. Skills: I can explain different network types, hardware and connection methods. I can compare network topologies and justify suitable choices for different scenarios. I can explain the purpose of protocols and layers in network communication. I can apply structured programming skills to solve longer programming problems. Prior/Future links: Builds on Topic 3 hardware/software and prepares for network security and cyber security.	Unit: Topic 4 Network security + Topic 6 testing and refinement Knowledge: Threats, malware, social engineering, encryption, authentication, backups and prevention methods. Skills: I can identify network security threats and explain their impact. I can recommend suitable prevention methods, including encryption, authentication, backups and anti-malware. I can justify security choices using the context of a scenario. I can test and refine Python programs using clear test data. Prior/Future links: Links directly to Topic 5 cyber security and extended Paper 1 scenario questions.	Unit: Topic 5 Issues and impact + Topic 6 exam programming practice Knowledge: Environmental, ethical, legal, cultural, privacy, ownership and cyber security issues. Skills: I can discuss ethical, legal, cultural, environmental, privacy and ownership issues linked to technology. I can write balanced extended responses using examples and accurate terminology. I can complete Paper 2 programming tasks involving code reading, debugging and program design. I can explain the impact of technology from different viewpoints. Prior/Future links: Completes first teaching of Topics 1-5 and prepares pupils for full GCSE revision.	Unit: Revision of Topics 1-3 + Topic 6 programming consolidation Knowledge: Computational thinking, data, hardware, software and programming language concepts. Skills: I can retrieve and apply knowledge from computational thinking, data representation and computer systems. I can answer GCSE-style questions using command words accurately. I can complete trace tables, correct code and write independent Python programs. I can identify weaker areas and improve my responses through targeted revision. Prior/Future links: Strengthens weaker areas identified from mocks and prepares for final GCSE examinations.	Unit: Revision of Topics 4-6 and final exam preparation Knowledge: Networks, network security, issues and impact, and the full Programming Language Subset for Python. Skills: I can retrieve and apply knowledge from networks, network security, issues and impact, and programming. I can complete timed Paper 1 and Paper 2 questions with appropriate exam technique. I can debug, test, refine and evaluate programming solutions. I can use precise technical vocabulary in written and programming responses. Prior/Future links: Prepares pupils for Paper 1 and Paper 2	Course completed	By the end of the year pupils can... • apply knowledge from GCSE Topics 1–6 accurately across familiar and unfamiliar scenarios • design, write, test, debug and refine Python programs using the GCSE programming language subset • explain algorithms, data, systems, networks, cyber security and impacts of technology using precise terminology • complete Paper 1 and Paper 2 questions with appropriate exam technique • write clear, justified extended responses about ethical, legal, cultural, environmental and security issues
	Assessment: Topic 4 networks assessment and applied programming checkpoint.	Assessment: GI1	Assessment: GI2	Assessment: GCSE exam paper 1: 10 May 2027 Paper 2: 19 May 2027	Assessment: GCSE exam paper 1: 10 May 2027 Paper 2: 19 May 2027		
Year 13	Unit: Programming paradigms: OOP and functional programming Knowledge Students develop their understanding of different programming paradigms, focusing on object-oriented programming and functional programming. They learn about classes, objects, attributes, methods, encapsulation, inheritance, recursion, first-class functions, higher-order functions, immutability and list processing. Skills: I can write, trace, and explain programs that use object-oriented programming concepts. I can use classes, objects, attributes, and methods accurately in Python. I can explain encapsulation and inheritance using examples. I can	Unit: Data structures Knowledge Students learn how data can be organised, stored and accessed using a range of data structures. This includes arrays, records, stacks, queues, linked lists, trees, binary trees, graphs and hash tables. They learn how each structure works, when it is suitable, and how operations such as adding, removing, searching and traversal are carried out. Skills: I can explain how different data structures store and organise data. I can compare the suitability of arrays, records, stacks, queues, linked lists, trees, graphs, and hash tables for different problems. I can trace operations on stacks,	Unit: Algorithms Knowledge Students develop their understanding of key algorithms and algorithmic efficiency. They study searching, sorting, recursion, tree traversal, graph traversal, optimisation and algorithmic complexity. They learn how algorithms can be represented, traced, compared and evaluated using Big O notation and computational thinking. Skills: I can trace searching, sorting, tree traversal, and graph traversal algorithms. I can write and interpret pseudocode for common algorithms. I can use recursion to solve appropriate problems. I can compare the efficiency of	Unit: Regular languages and theory of computation Knowledge: Students learn how languages and computational processes can be represented formally. They study finite state machines, regular expressions, regular languages, BNF/syntax diagrams, and language classification. They understand how valid strings can be described, recognised and generated using formal language tools. Skills: I can construct and interpret finite state machines. I can write and apply regular expressions to identify valid strings. I can use BNF and syntax diagrams to describe the structure of a language. I can explain the	Unit: Final revision, skeleton code and exam preparation Knowledge Students consolidate knowledge from across the A Level course, including programming paradigms, data structures, algorithms, regular languages and key Paper 1 and Paper 2 theory content. They also focus on the preliminary material/skeleton code, exam technique and weaker areas identified from mocks. Skills: I can analyse skeleton code and explain how key sections of the program work. I can debug, test and refine code accurately. I can complete timed Paper 1 and Paper 2	Course completed	• By end of year pupils can... • Write, trace, debug and refine complex programs using procedural, object-oriented and functional programming concepts. • Explain and compare key data structures, including stacks, queues, linked lists, trees, graphs and hash tables, and justify their use in different scenarios. • Trace, write and evaluate algorithms, including searching, sorting, recursion,

	write and trace recursive functions. I can explain key functional programming concepts, including first-class functions, higher-order functions, and immutability. I can compare programming paradigms and explain when each approach may be suitable. Prior/Future links: Builds on Year 12 programming foundations and supports Paper 1 programming questions, NEA completion, data structures, algorithms and skeleton code preparation. This unit strengthens pupils' ability to think about how programs are structured and how different approaches can be used to solve complex problems.	queues, linked lists, and trees. I can explain tree and graph terminology accurately. I can implement selected data structures in Python. I can justify why a particular data structure is appropriate for a given scenario. Prior/Future links: Builds directly on HT1 programming paradigms and Year 12 programming knowledge. Prepares pupils for algorithms, tree traversal, graph traversal, hash tables, Paper 1 questions and more complex programming scenarios.	algorithms using Big O notation. I can explain how time and space complexity affect the suitability of an algorithm. I can implement selected algorithms in Python and evaluate their effectiveness. Prior/Future links: Builds on HT2 data structures because many algorithms depend on how data is stored and accessed. Strengthens Paper 1 problem-solving and prepares pupils for skeleton code analysis, algorithm tracing and extended exam questions.	difference between regular and non-regular languages at an appropriate level. I can apply formal language concepts to exam-style questions. I can link regular languages to programming language translations and compilers. Prior/Future links: Builds on logical thinking, abstraction, algorithms and programming language concepts from Year 12 and Year 13. Supports Paper 2 theory and links to translators, compilers, syntax analysis and how programming languages are structured.	exam questions. I can write clear extended responses using precise technical terminology. I can identify my weaker areas and use targeted revision to improve them. I can apply knowledge from across the course to unfamiliar programming and theory questions. Prior/Future links: Consolidates Year 12 and Year 13 content in preparation for final A Level examinations. Supports independent revision, exam confidence and the ability to apply knowledge across programming, theory and problem-solving contexts.		tree traversal and graph traversal. - Analyse the efficiency of algorithms using Big O notation and explain the impact of time and space complexity. - Construct and interpret finite state machines, regular expressions, BNF and syntax diagrams. - Analyse skeleton code confidently, explaining how key sections work and how the program could be adapted or improved. - Apply knowledge from across the A Level course to unfamiliar programming and theory questions. - Write clear, accurate and well-structured exam responses using precise Computer Science terminology.
	Assessment: Practical programming tasks and exam-style coding questions.	Assessment: GI1	Assessment: GI2	Assessment: Theory assessment on FSMs, regular expressions and language questions.	Assessment: Full Paper 1 and Paper 2 mock/practice papers.	Assessment: A level exam	