

**CURRICULUM MAP**

Subject	Computer Science
Head of Department	Mr N Rahman

SCHOOL CURRICULUM INTENT**Rutlish School: Curriculum Intent****Rutlish School: Curriculum Intent**

Rutlish School provides a meaningful, broad and balanced curriculum which is accessible to all, as well as supports and challenges all students. Through all we do, we prepare students for opportunities, responsibilities and experiences later in life for them to be aware of their responsibilities and feel confident to participate and contribute to society. We aim to inspire, enable and facilitate lifelong learners that build on their individual strengths and capabilities and achieve their ambitions. We seek to support our students in becoming healthy, happy, resilient young adults, who are knowledgeable, kind, aware, confident, capable and skilful members of society.

The school aims to:

- ensure that the curriculum is designed for every student of every ability and every background to be supported in making the best possible progress and attainment from their starting point; all students can access the curriculum offer, with planning and teaching that support, stretch and challenge all learners across a full range of abilities, and making any reasonable adjustments where particular needs are identified to reduce barriers to learning and enable all pupils to achieve highly, including for disadvantaged students, students with SEND, and those facing barriers to learning and wellbeing;
- provide a curriculum that is sequenced to build skills and knowledge throughout students' time at Rutlish School, to promote a deeper understanding of the world outside the classroom and equip them for their next steps in education, careers and in life;
- ensure that our curriculum offer support different educational and career pathways, including EBACC and vocational;
- ensure our curriculum consistently promotes high moral standards, social and self-awareness and allows students to formulate informed opinions on social issues such as, equality, diversity and inclusivity as well as the practical aspects of society;
- enrich the curriculum and provide opportunities for students to build cultural capital, enhance a wide range of skills and knowledge beyond requirements of the national curriculum, and personalise and apply learning in other contexts;
- provide students with the skills and knowledge necessary to becoming independent, analytical, critical, and innovative thinkers and encourage students' curiosity, creativity, self-expression, resilience, and confidence;

DEPARTMENT CURRICULUM INTENT

The aim of Computer Science at Rutlish is to provide a high-quality computing education which equips students to use computational thinking and to creatively understand and change the world. We study Computer Science to help us think in a more logical way and become better at making decisions and solving problems in a world that is rapidly changing and demanding technological expertise. The curriculum will teach students key knowledge about how computers and computer systems work, and how they are designed and programmed.

- Students develop strong comprehension skills by interpreting worksheets and then completing practical tasks on the computers
- Students are guided to extract key information, identify target audiences, and understand constraints (such as purpose, format, and deadlines), ensuring they can respond accurately and creatively to real-world scenarios
- Numeracy is embedded through technical and design-based activities, such as calculating image dimensions, resolution, file sizes, timelines and data representation
- Literacy skills are developed through writing of planning documents, worksheet completion and the need to read, interpret and understand key terminology required at KS3. Through the writing of long answers in different units of work.
- Through evaluations and justification of choices and recommendations based on research-based tasks.
- Students use subject-specific terminology
- Teaching is adapted to remove barriers by using a range of accessible resources and differentiated strategies, such as visual exemplars, step-by-step demonstrations, scaffolded templates, and assistive technology.
- Units are broken into manageable stages enabling all learners to progress at their own pace while developing confidence with digital tools

KEY STAGE 3 RATIONALE/ INTENT

The KS3 curriculum has been designed to ensure learners have sufficient knowledge to stay safe online and use computers safely in life. The KS3 curriculum also provides a focus on developing resilient learners who are able to learn from mistakes and effectively solve problems. The topics at KS3 give a basis of knowledge, skills and understanding to allow students to progress onto either iMedia or Computer Science at KS4 and will provide exposure to those subjects so that students can make an informed decision on their GCSE choices.

KEY STAGE 4 RATIONALE/ INTENT

At KS 4 students build on the skills they have learned in KS 3. The curriculum develops the student's ability to become a more resilient learner that is focused on logical thinking, decomposition and abstraction of problems to ultimately become advanced at solving computational problems. Students also continue to develop skills in digital literacy where they use computer systems to create and evaluate digital products for specific audiences and purposes.

KEY STAGE 5 RATIONALE/ INTENT

The KS 5 program of study develops the ability of students to become independent learners and to use research to extend their knowledge beyond the scope of the specification. Computer Science at KS 5 provides students with the challenge of programming a computer game that embeds complexities, creativity and allows students to problem solve and follow a test-driven development process. The curriculum enables students to become critical thinkers that are able to apply and evaluate their knowledge to different scenarios. The curriculum enables students to become critical thinkers that are able to apply and evaluate their knowledge to different scenarios.



QUALITY FIRST TEACHING

- Scaffolding, modelling and sample answers incorporated into all lesson resources.
- All programming tasks and theory questions set in GCSE exam question format
- Metacognition strategies incorporated into lessons through providing opportunities to peer and self-assess and reflect on progress.
- Explicit teaching of programming and computational skills to enable students to become independent learners focused on problem solving.
- Continuous recap and revise promote knowing more and remembering more.
- Self-assessment to promote independent learning, self-correction and developing individual subject knowledge; algorithmic quizzes targeted at student weaknesses.



YEAR 10: Computer Science Theory

	Autumn 1	Autumn 2	Spring 1	Spring 2	Summer 1	Summer 2
KNOWLEDGE	UNIT OF WORK: <u>Systems Architecture</u>	UNIT OF WORK: <u>Data Representation</u>	UNIT OF WORK: <u>Networks and Protocols</u>	UNIT OF WORK: <u>Network Security and Systems Software</u>	UNIT OF WORK: <u>Impacts of Digital Technology</u>	UNIT OF WORK: <u>Computational Thinking, Algorithms and Programming</u>
	The purpose of the CPU, the common components and their functions. The different factors that affect CPU performance and how primary and secondary memory is used in computers. The need for secondary storage and the advantages and disadvantages of different types of storage. The importance of embedded systems.	How data is represented in binary and hex. To recall knowledge from KS 3 to convert binary to denary, add and subtract binary numbers. What ASCII is and to be able to recall knowledge from KS 3 on ASCII representation while appreciating character sets. To perform logic and arithmetic shifts. Understanding bitmap graphics and how each pixel is represented in binary.	Explore the different types of networks and topologies and the hardware needed to connect to different types of networks. To explore wired and wireless networks and to understand the different protocols that operate at each layer of the different types of networks. Explain IP addressing and how it links to DNS along with MAC addressing. Understand ethernet standards, encryption, client-server and peer-to-peer networks	Explore how networks come under attack and how networks can be protected from threats. To be able to identify the type of network threats and preventions of vulnerabilities. To explore the different types of OS and to understand the different purposes and functions of an OS. Types of utility software and the purpose and function of each.	Impacts of digital technology on wider society including ethical and cultural issues. Impacts of digital technology on the environment including: The impact of manufacture and disposal; The impact of upgrading or replacing; The impact of e-waste Explore the legislation relevant to Computer Science. Impacts of digital technology on wider society including: Legal issues and Privacy issues. Describe legislation relevant to Computer Science: The Data Protection Act 2018; Computer Misuse Act 1990; Copyright Designs and Patents Act 1988	To know how to use Python to code small programs from the given tasks and will be able to apply knowledge to new programming scenarios. Students will know how to use Python to code programs using the key skills listed below
KEY SKILLS						
	Analyse the performance of the CPU including the FDE cycle. Remember and understand the architecture the CPU including Von Neumann (PC, MAR, MDR and Acc). Able to know the differences between RAM, ROM and virtual memory. Remember and understand the purpose and characteristics of embedded systems. Evaluate the use of secondary storage to a given scenario.	Understand the methods of conversions. Apply the formulae to various problems. Defining terms like nibble, bits, bytes, terabytes, etc. Conversion of denary to binary and HEX and vice versa. Addition of binary and understanding of binary codes to represent characters. Able to use image meta data and explain file size and resolution. While being able to represent sound in binary v quality.	Understand the different types of network protocols and be able to recall the different network layers and their functions. To be able to apply formulae to calculate network speed. To be able to evaluate the advantages and disadvantages of different networks and topologies Understand modes of connections like WIFI and Bluetooth while understanding Hosting and the Cloud. Be able to describe and know protocols.	Understand each of the types of threats to a network including: Malware, Phishing, Social Engineering, Brute Force Attacks, DOS, Data Interception and Theft and SQL Injection. Remember and understand the types of utility software. Understand how to identify and prevent network vulnerabilities	Discuss the impacts of digital technology on the wider society including ethical issues, cultural issues and environmental issues. Discuss the impact of manufacture, disposal, upgrading and replacing digital technology Discuss the impact of e-waste Discuss the impact of digital technology regarding legal issues and privacy issues including licences for open source and propriety software. Describe legislation relevant to Computer Science including The Data Protection Act 2018 Computer Misuse Act 1990 Copyright Designs and Patents Act 1988	<u>Use Python to apply the following practical skills: (Recall embed skills from KS 3)</u> Escape sequences Code small programs Identify and correct syntax errors independently Maths operators Data type, variable and inputs String formatting Relational operators Selection



CURRICULUM SEQUENCING AND HOW WE BUILD ON SKILLS AND KNOWLEDGE	Link to KS 3: Basic understanding of how computers work and their components. Year 7: Introduction to computers. Year 8: How computers work.	Link to KS 3: Year 7: Introduction to binary. Year 8: Data representation	Link to KS 3: Year 8: Networks topic Year 9: Network topic	Link to KS 3: Year 9: Networks topic	Link to KS 3: Year 7: Basic Computer Use Year 8: Computer Use	KS 3 SOW Year 7 – Input, Output, variables, selection, data types
WIDER APPLICATION AND FUTURE PATHWAYS	The System Architecture unit of GCSE Computer Science helps students understand how computers process, store and execute data and instructions. This knowledge underpins all modern digital technologies and enables students to appreciate how computer systems support everyday life, businesses and global communication. The knowledge developed in this unit allows students to build their Cultural Capital by engaging confidently with discussions about emerging technologies and their role in shaping the future Careers: Unit provides foundational knowledge for a range of careers, including: Computer Systems Engineer, Software Developer, Cyber Security Analyst, Network Engineer, IT Support Specialist, Hardware Engineer, Cloud Computing Technician, Data Centre Technician Understanding processors, memory, storage and computer performance is essential in many digital and technology-related professions. The concepts studied provide a strong foundation for: Future Study: A Level Computer Science, Digital and	For Data Representation, the links to the wider world can be seen as follows: It underpins all digital technologies, including smartphones, computers, streaming services, social media, online banking, and artificial intelligence. Students understand how text, images, sound, and instructions are stored and processed digitally, helping them make sense of the technology they use daily. Develops awareness of how vast amounts of data are communicated globally through networks and the internet. Careers: Unit provides plenty of opportunities like: Software Developer; Cyber Security Analyst; Data Scientist; AI Engineer; Games Developer; Network Engineer; Digital Media Designer and Computer Systems Engineer Understanding binary, hexadecimal, encoding, and file sizes provides essential foundations for these professions. Future Study: Provides essential knowledge for: A Level Computer Science; BTEC Computing and IT; T Levels in Digital Production and Development and Computer Science, Software Engineering,	For Networks it links to the different connection types and their respective advantages and disadvantages. Careers: Networking knowledge is essential in many careers, including: Network Engineer; Cyber Security Analyst; Systems Administrator; Cloud Engineer; IT Support Technician; Telecommunications Engineer; Data Centre Technician; Infrastructure Engineer and Software Developer. These professionals design, maintain and secure the networks that businesses, schools, hospitals and governments rely on every day. Future Study: The networks topic provides foundations for: A-Level Computer Science; BTEC Information Technology; T Levels in Digital Production and Support Services; Digital and Technology Apprenticeships; Cyber Security qualifications; Computer Science degrees and Networking certifications such as Cisco CCNA and CompTIA Network+. Equality, Inclusion and Diversity: Networks help create a more inclusive society by: Enabling remote learning for students who cannot attend school; Supporting accessibility tools for people with disabilities;	For Network Security and System Software, here is how the topic links to the wider world: Careers: This unit develops skills needed in many technology careers, including: Cyber Security Analyst; Network Engineer; Systems Administrator; Software Developer; Cloud Computing Engineer; IT Support Technician; Ethical Hacker; Digital Forensics Investigator and Data Protection Officer. These jobs help organisations keep their systems secure, reliable and connected. Future Study: This unit provides a foundation for: GCSE Computer Science progression; A-Level Computer Science; T Levels in Digital or Cyber Security; BTEC Information Technology; Apprenticeships in IT and Networking and University courses in Computing, Cyber Security, Software Engineering and Network Engineering. Equality, Inclusion and Diversity: Secure networks allow everyone to access online services safely, regardless of background or location; Accessibility features in system software help users with disabilities use computers effectively; The technology industry encourages greater	For Impact of Digital Technology, here is how the topic links to the wider world: Careers: Understanding the impact of digital technology helps prepare students for careers such as: Software Developer; Cyber Security Analyst; Data Scientist; Web Designer; Network Engineer; Digital Marketing Specialist; Game Developer; AI Engineer; IT Support Technician; Systems Analyst These jobs require people to understand how technology affects individuals, businesses, and society. Future Study: The unit provides foundations for further study in: A-Level Computer Science; Information Technology qualifications; T Levels in Digital Production or Digital Support Services; Cyber Security qualifications; Business and Digital Media courses; Computing, Data Science and Artificial Intelligence degrees and Digital and technology apprenticeships Understanding the impact of technology supports progression into a wide range of academic and vocational pathways. Equality, Inclusion and Diversity: Digital technology can	Careers: See PPT slides. Famous programmers from the past. Companies that use Python Programming software used in the world of work.



Computing T Levels, BTEC Information Technology qualifications, Apprenticeships in Digital Technology, University courses in Computer Science, Cyber Security, Software Engineering and Computer Engineering Equality, Inclusion and Diversity: Learning about system architecture encourages students to consider how computing devices can be designed to meet the needs of diverse users. Advances in hardware and processing power have enabled the development of assistive technologies such as voice-controlled systems, screen readers and communication devices, helping to create a more inclusive digital society. The technology sector benefits from diverse perspectives in designing hardware and systems that meet the needs of people from different backgrounds and abilities. Employment: Employers increasingly value individuals who understand how computer systems operate. Knowledge of system architecture develops: Logical thinking, Analytical skills, Technical problem-solving, Understanding of performance and efficiency These skills are transferable across many sectors, including technology, engineering, healthcare, finance and manufacturing.	Data Science, AI and related university courses Data representation forms a core concept that supports many advanced computing topics. Equality, Inclusion and Diversity: Learning about Digital data representation supports accessibility technologies such as screen readers, speech-to-text software, subtitles, and translation tools. Students explore how technology can make information available to people with different needs, abilities, languages, and backgrounds. Encourages understanding of the importance of designing technologies that work for diverse users worldwide. Cultural Capital. Builds digital literacy and understanding of the technological world. Enables students to engage confidently with modern technologies and discussions about digital innovation. Develops awareness of how technology has transformed communication, entertainment, education, and business globally. Employment: Problem-solving skills; Logical thinking; Analytical skills; Accuracy and attention to detail; Mathematical application in real-world contexts. These transferable skills are valued across many industries, not just computing.	Allowing people from different cultures and countries to communicate instantly; Providing opportunities for under-represented groups to access education and employment online and encouraging diverse teams to collaborate globally. Students should understand the importance of ensuring everyone has fair access to technology and the internet. Employment: Almost every organisation requires computer networks, including: Schools and universities; Hospitals and healthcare services; Banks and financial institutions; Government departments; Technology companies; Retail businesses and Manufacturing industries Understanding networking develops valuable technical and problem-solving skills sought by employers.	diversity, with opportunities for people of all genders, cultures and backgrounds; Cybersecurity helps protect vulnerable groups from online fraud, identity theft and discrimination. Employment: Students develop transferable skills valued by employers: Problem-solving; Logical thinking; Troubleshooting; Risk assessment; Attention to detail; Communication and teamwork; Understanding digital systems These skills are useful in both technical and non-technical careers. Students learn to be responsible digital citizens by understanding: Online safety and cybersecurity threats; Data protection and privacy; Ethical use of technology; Laws relating to hacking, computer misuse and intellectual property and how to protect themselves and others online	increase access to education, employment and services for people from different backgrounds; Accessibility tools such as screen readers, voice recognition and subtitles support people with disabilities; Online platforms enable greater participation from diverse communities around the world; Students explore both the positive and negative impacts of technology, including the digital divide and unequal access to technology; The technology sector actively promotes diversity and inclusion to improve innovation and representation. Employment: Students develop valuable workplace skills, including: Critical thinking; Ethical decision-making; Problem-solving; Research skills; Evaluation and analysis; Communication; Digital literacy; Understanding emerging technologies These skills are increasingly important in almost every employment sector. This helps students make informed decisions when using technology both now and in the future.	
--	--	---	--	---	--



ASSESSMENTS Summative and Formative as applicable	Formative: Self and peer assessment Class discussions Classwork and homework Starter (Do Now) Exit tickets Think pair share I do, We do and You do Quizzes Mnemonics	Formative: Self and peer assessment Class discussions Classwork and homework Starter questions (Do Now) Exit tickets Think pair share I do, We do and You do Quizzes Mnemonics	Formative: Self and peer assessment Class discussions Classwork and homework Starter questions (Do Now) Exit tickets Think pair share I do, We do and You do Quizzes Mnemonics	Formative: Self and peer assessment Class discussions Classwork and homework Exit tickets Think pair share I do, We do and You do Quizzes Mnemonics Retrieval power Ticket	Formative: Self and peer assessment Class discussions Classwork and homework Starter questions Exit tickets Think pair share I do, We do and You do Quizzes Mnemonics Retrieval power Ticket	Formative: Self-evaluation of programs Class discussions Think pair share Peer feedback Summative: End of unit test
	Summative: End of unit tests Practise tests End of year	Summative: End of unit test Practise tests End of year	Summative: End of unit test Practise tests End of year	Summative: End of unit test	Summative: End of unit test	
FEEDBACK SUPPORTS LEARNING	Opportunity for students to reflect on learning, respond to feedback, improve work, etc. 	Opportunity for students to reflect on learning, respond to feedback, improve work, etc. 	Opportunity for students to reflect on learning, respond to feedback, improve work, etc. 	Opportunity for students to reflect on learning, respond to feedback, improve work, etc. 	Opportunity for students to reflect on learning, respond to feedback, improve work, etc. 	Opportunity for students to reflect on learning, respond to feedback, improve work, etc. 
INCLUSION AND ACCESSIBILITY	The System Architecture unit is designed to be accessible to all pupils through high-quality adaptive teaching, appropriate support and high expectations. All students are given opportunities to develop their understanding of how computer systems work while maintaining ambition and challenge for every learner. To support pupils SEND: Key vocabulary (e.g., CPU, RAM, cache, fetch-execute cycle) is explicitly taught and regularly revisited; Visual aids, diagrams, animations and physical models are used to support understanding of abstract concepts; Information is broken down into manageable steps with clear, structured explanations; Retrieval practice and regular knowledge checks help reinforce learning; Writing frames, glossaries and guided notes support literacy	The Data Representation unit is designed around clear, structured concepts that can be built progressively from prior mathematical and logical understanding. To support SEND: Pre-teach subject vocabulary; Use visual aids, diagrams and dual coding; Provide sentence starters and writing frames; Chunk learning into small steps; Provide calculation guides and reference sheets; Use regular recap and retrieval activities; Maintain predictable lesson structures; Use clear, unambiguous instructions; Link concepts to real-world systems and interests; Allow additional thinking and response time and presenting information in multiple formats; Reduce unnecessary cognitive load. To ensure disadvantaged pupils can fully participate: Avoiding assumptions about prior	For Networks Unit, the following can be included in a curriculum: Use a variety of teaching approaches including teacher modelling, practical demonstrations, videos and hands-on activities; Provide key vocabulary lists with definitions and visual representations; Use network diagrams, animations and real-life examples to support understanding of abstract concepts; Chunk learning into small, manageable steps; Regular retrieval practice to reinforce prior knowledge; Opportunities for collaborative learning and discussion; Use practical network simulations to make concepts accessible; Provide scaffolded worksheets and structured note-taking templates. To ensure SEND can access the unit: Pre-teach technical vocabulary such as LAN, WAN, router, switch and protocol; Provide dual coding through	For Network Security and System Software unit the following applies. To support SEND: Use clear and consistent subject-specific vocabulary with definitions and glossaries; Provide network diagrams, flowcharts and visual representations to support understanding; Chunk information into smaller sections to reduce cognitive load; Model processes such as data transmission, authentication and system software functions step-by-step; Use worked examples before independent application; Provide scaffolded note templates, writing frames and keyword banks where appropriate; Allow additional processing time and opportunities for overlearning of key concepts; Use assistive technology where needed, such as text-to-speech, speech-to-text and screen readers; Check understanding frequently through low-stakes	For Impact of Digital Technology unit, the following applies: To support SEND: Pre-teach key terminology such as digital divide, privacy, cybersecurity, ethics, and environmental impact; Use knowledge organisers and visual aids to support understanding; Chunk reading materials into smaller sections; Provide writing frames and model answers. Communication and Interaction; Provide clear, concise instructions; Allow think-pair-share activities before whole-class discussion; Use visual prompts and structured questioning; Ensure digital resources are compatible with assistive technologies; Provide enlarged text, coloured overlays, or screen-reading software where appropriate; Offer alternative methods of recording responses; Create predictable lesson structures; Use positive reinforcement and scaffolded	



demands; Dual coding techniques combine text and visuals to aid memory and understanding; Additional processing time is provided where appropriate; Assistive technologies, including screen readers, speech-to-text software and adapted resources, are used when required. To ensure disadvantaged pupils can fully participate and succeed: Knowledge gaps are identified early through assessment and targeted intervention; Key concepts and technical vocabulary are pre-taught and regularly reviewed; Access to digital resources, revision materials and independent learning opportunities is supported both in school and, where possible, at home; Real-world examples make learning relevant and help build background knowledge and cultural capital; Structured scaffolding helps pupils access complex content while gradually increasing independence; Positive role models from diverse backgrounds within the computing industry are highlighted to raise aspirations. Teaching is adapted to meet varying learning needs while maintaining the same ambitious curriculum: Complex concepts such as processor performance and machine code are broken into smaller learning steps; Teachers use modelling and worked examples to demonstrate processes before independent practice; Questioning is carefully planned to check understanding and address	technological experience; Explicitly teaching key terminology and background knowledge; Providing access to digital resources and revision materials; Frequent opportunities for guided practice and feedback; Making links to future careers and pathways that pupils may not otherwise be aware of. Adaptive teaching ensures that all pupils work towards the same learning goals but receive support appropriate to their needs. Examples include: Scaffolding binary and hexadecimal conversion activities; Tiered questioning that gradually increases in complexity; Worked examples followed by partially completed examples; Flexible grouping for collaborative problem solving; Use of manipulatives, diagrams and visual models; Additional challenge through extension tasks such as exploring Unicode, data compression algorithms or file size calculations.	diagrams and written explanations; Use colour-coded network diagrams to highlight components and connections; Offer instructions in clear, concise steps; Allow additional processing time for complex concepts ;Use assistive technologies where appropriate (text-to-speech, speech-to-text, screen readers); Provide worked examples before independent practice; Check understanding regularly through low-stakes questioning; Reduce cognitive load by limiting unnecessary information on slides and worksheets. Disadvantaged Pupils are catered for: Avoid assuming prior access to home technology or broadband; Explicitly teach technological experiences that some pupils may not have encountered; Provide access to devices and software during lessons where possible; Use school-based practical activities to ensure equal opportunities to experience networking concepts; Build cultural capital through exposure to real-world technology careers and industry examples; Provide structured revision materials and knowledge organisers; Ensure key resources are available both digitally and in print; Use targeted questioning to identify and address knowledge gaps. Adaptive Teaching is catered for by: Scaffold tasks through graduated levels of support; Provide sentence starters and writing frames for extended responses; Use challenge questions to deepen understanding for higher-attaining pupils; Vary questioning techniques to engage all learners; Adapt practical activities	questioning and retrieval activities; To ensure Disadvantaged Pupils are catered for: Avoid assumptions about prior access to computers, broadband or technology at home; Ensure all essential learning can be completed within school where possible; Explicitly teach terminology and technical language that may not have been encountered outside school; Make links to everyday technologies that all pupils are likely to experience, such as smartphones, gaming consoles and social media; Provide access to revision materials and digital resources through school systems; Use structured support to develop confidence with technical concepts and exam-style questions; Promote awareness of careers and pathways into computing that are accessible to all students. Adaptive Teaching is achieved by: Adjust the level of scaffolding according to pupil need whilst maintaining curriculum expectations; Use a range of teaching approaches including direct instruction, discussion, practical tasks and independent learning; Provide differentiated support through by Visual aids and prompts, Guided practice, Vocabulary support, Extension tasks and Targeted questioning. Also: Offer multiple ways for pupils to demonstrate understanding, for example verbally, practically or through written responses; Use formative assessment regularly to identify misconceptions and adapt teaching accordingly; Develop	tasks; Consider sensitive discussion around social media, online behaviour, and cyberbullying; Lessons use a range of teaching approaches, including discussion, videos, case studies, diagrams, real-world examples, and written tasks; Key vocabulary is explicitly taught and revisited regularly; Learning is broken into manageable steps with clear success criteria; Opportunities for collaborative learning enable pupils to learn from peers; Resources are provided in accessible formats where needed; Technology-related concepts are linked to pupils' everyday experiences, such as social media, gaming, streaming services, and smartphones; Retrieval practice and regular review activities support long-term memory. Disadvantaged Pupils are supported by: Avoid assuming prior knowledge of technology access at home; Provide access to devices and resources during lessons where possible; Explicitly teach digital literacy and online safety; Use examples that reflect a range of socioeconomic backgrounds; Provide additional vocabulary support and opportunities for guided practice; Ensure homework can be completed without requiring specialist equipment or paid software; Build cultural capital through exposure to current technological developments and career pathways. Adaptive Teaching includes: Support; Simplified texts while maintaining core learning objectives; Guided worksheets and structured note-taking templates; Vocabulary banks and glossaries; Additional modelling	
---	--	--	--	---	--



	misconceptions; Tasks may vary in the level of support provided, while learning objectives remain consistent for all pupils; Alternative methods for recording understanding, such as verbal explanations, labelled diagrams or practical activities, may be used where appropriate; Challenges and extension activities are available for pupils who are ready to deepen their understanding.		according to pupil needs and confidence levels; Use retrieval quizzes at different difficulty levels; Offer alternative ways to demonstrate learning, including diagrams, verbal explanations and practical demonstrations; Regularly assess understanding and adapt instruction accordingly.	understanding through concrete examples before moving to more abstract networking and security concepts.	and teacher-led examples; Strategic seating and targeted questioning; Challenge; Independent research into emerging technologies such as AI, quantum computing, or the Internet of Things; Evaluation of ethical and legal issues from multiple viewpoints; Extended discussion and debate activities; Analysis of real-world case studies involving digital technology; Adaptations focus on supporting access to learning rather than lowering expectations.	
SPECIALIST VOCABULARY	CPU, Fetch-Execute, RAM, ROM, Virtual Memory, Non-Volatile, Optical, Magnetic Solid state, Disk, Hard Disk, Blue-Ray, DVD, Portability, Speed, Durability, Reliability, Cost, Media, Primary and secondary storage, ALU, CU, Cache, Registers, Von Neumann, MAR, MDR. Cores, Clock speed, Embedded Systems, Memory Address,	Bit, Nibble, Byte, Kilobyte, Megabyte, Gigabyte, Terabyte, Petabyte, Binary, Sample Rate, Colour Depth, Pixel, Bit per Character, Binary Shift, Shift Left, Shift Right, Most Significant Bit, Least Significant Bit, Character Set, ASCII, Unicode, Metadata, Hertz, Compression, Lossy, Lossless, Overflow, Two's complement, Sign and Magnitude,	LAN, Local Area Network, WAN, Wide Area Network, bandwidth, latency, Wireless access points, routers, switches, NIC, Network Interface Controller/Card, Transmission media, DNS, Domain Name Server, Hosting, The Cloud, Web servers and clients, star network, mesh network, topology, IP address, web server, file server, wired network, wireless network, Ethernet, Wi-Fi, Bluetooth, encryption, IP addressing, MAC addressing, TCP/IP, Transmission Control Protocol/Internet Protocol, FTP, File Transfer Protocol, HTTP, HTTPS, FTP,POP, MAP, Post Office Protocol, IMAP, Internet Message Access Protocol, SMTP, Simple Mail Transfer Protocol, layers, IPv4, IPv6, MAC address.	Malware, Virus, Trojan Horse, Worms, Social Engineering, Phishing, Brute-Force Attack, Denial of Service Attack, Data Interception and Theft, SQL Injection, Penetration Testing, Anti-Malware Software, Anti-Virus Software, Firewalls, User Access Levels, Passwords, Encryption, Physical Security, Operating System, User Interface, Graphical User Interface (GUI), Command Line Interface (CLI), Memory Management, Multitasking, Peripheral Management, Drivers, User Management, File Management, Utility Software, Encryption Software, Defragmentation and Data Compression.	Ethical, cultural, environmental, legislation, manufacture, disposal, upgrade, replace, e-waste, privacy, legal, data protection, computer misuse, copyright, copyright designs and patents act, open source, proprietary, software licence	Escape sequences, Data types, Variables, String formatting, Relational operators, Input/Output, IF ELSE
QUALITY FIRST TEACHING	✓ Strategies to learn more, remember more (metacognition) used in lessons e.g. retrieval, elaboration, interleaving, dual coding, etc. ✓ Differentiation and reasonable adjustments for students with SEND, EAL, etc. such as scaffolding, visual aids, audio, physical resources, planned questioning, etc. ✓ Opportunities for Literacy, Numeracy and Oracy, including a focus on reading ✓ Opportunities to apply key concepts and address misconceptions					



YEAR 10 & 11: Computer Science Coding

	Autumn 1	Autumn 2	Spring 1	Spring 2	Summer 1	Summer 2
KNOWLEDGE	UNIT OF WORK: Computational Thinking, Algorithms and Programming	UNIT OF WORK: Algorithms & Pseudo Code	UNIT OF WORK: Subprograms	UNIT OF WORK: Subprograms	UNIT OF WORK: Lists and loops	UNIT OF WORK: Nested Lists & File Reading
	To know how to use Python to code small programs from the given tasks and will be able to apply knowledge to new programming scenarios. Students will know how to use Python to code programs using the key skills listed below.	Students will be able to solve computational problems using flowcharts and pseudo code. They will be able to use trace tables to trace through a program to identify logic errors and to trace data at different stages of a program. They will understand Boolean logic and will be able to recall the rules for AND, NOT and OR gates	Through practical programming students will be able to use and recall their knowledge of selection and iteration and apply it to code programs using functions and procedures. Understand the difference between a function and a procedure and learn to organise their subprograms into larger programs that solve problems.	Through practical programming students will be able to use and recall their knowledge of selection and iteration and apply it to code programs using functions and procedures. Understand the difference between a function and a procedure and learn to organise their subprograms into larger programs that solve problems	Through practical programming students will be able to use and recall their knowledge of index positions, selection, and iteration and apply it to code programs that makes use of lists. Students will understand the difference between 1 dimensional and 2 dimensional arrays. They will develop solutions to larger programs using lists.	Through practical programming students will be able to use and recall their knowledge of lists to create programs using nested lists. Students will build their existing knowledge to code a program that can read and write data to a file. They will apply all their programming knowledge to effectively solve a complex problem.
KEY SKILLS	Use Python to apply the following practical skills: (Recall embed skills from KS 3) Escape sequences Code small programs Identify and correct syntax errors independently Maths operators Data type, variable and inputs String formatting Relational operators Selection	Use Python to apply the following practical skills: To be able to: Use the correct flowchart symbols to accurately sequence the steps to solve a problem. Use.draw.io software to create flowcharts. Use normal data, boundary data and erroneous data to test a program. Use a trace table effectively to trace the content of data in variables. Use the OCR Exam ref language to write and read pseudo code. Draw Boolean gates and tables.	Use Python to apply the following practical skills: Decomposition Breaking larger programs down into subprograms. Difference between functions and procedures. Calling function/procedure Return a value to a function	Use Python to apply the following practical skills: Decomposition Breaking larger programs down into subprograms. Difference between functions and procedures. Calling function/procedure Return a value to a function	Use Python to apply the following practical skills: For loops While loops Built in functions Validation in programs Sub programs	Use Python to apply the following practical skills: Reading and writing to a file Nested lists Flowcharts Pseudo code
CURRICULUM SEQUENCING AND HOW WE BUILD ON	KS 3 SOW Year 7 – Input, Output, variables, selection, data types	KS 3 SOW Practical programming using OR, AND to code programs making use of Boolean logic e.g. grade calculator, password program.	KS 3 SOW Year 9 SOW – selection and iteration.	KS 3 SOW Year 9 SOW – selection and iteration	KS 3 SOW Year 9 SOW – sorting and searching algorithms	KS 3 SOW Iteration Selection Decomposition
	Careers: Algorithms are used in many careers, including: Software Developer; Data Scientist; Cyber Security Analyst; Artificial Intelligence Engineer; Game Developer; Robotics Engineer; Web and App Developer Professionals use algorithms to solve problems efficiently and create technology that people use every day. Future Study: Knowledge of algorithms provides a foundation for: A-Level Computer Science; BTEC Computing qualifications; University degrees in Computer Science, Software Engineering, Data Science, AI, and Cyber Security; Apprenticeships in digital and technology sectors Algorithms are a core concept in all advanced computing courses.					
WIDER APPLICATION AND FUTURE PATHWAYS						



	Equality, Inclusion and Diversity: Algorithms affect many people: Inclusive algorithms consider the needs of different users; Developers work to reduce bias in algorithms that could unfairly disadvantage certain groups; Diverse teams often create better algorithms because they bring different perspectives and experiences. This highlights the importance of representation in the technology industry. Employment: Many modern workplaces require problem-solving skills that relate to algorithmic thinking. Opportunities exists in: Banking and finance; Healthcare; Manufacturing; Transport; Retail; Scientific research; Weather forecasting and Online shopping and delivery tracking.					
ASSESSMENTS Summative and Formative as applicable	Formative: Self-evaluation of programs Class discussions Think pair share Peer feedback Summative: End of unit test	Formative: Self and peer assessment of trace tables, pseudo code and flowcharts. Think pair share Summative: End of unit test	Formative: Self-evaluation of programs Class discussions Think pair share Peer feedback Summative: End of unit test	Formative: Self-evaluation of programs Class discussions Think pair share Peer feedback Summative: End of unit test	Formative: Self-evaluation of programs Class discussions Think pair share Peer feedback Summative: End of unit test	Formative: Self-evaluation of programs Class discussions Think pair share Peer feedback Summative: End of unit test
FEEDBACK LESSONS PLANNED	Opportunity for students to reflect on learning, respond to feedback, improve work, etc. ✓	Opportunity for students to reflect on learning, respond to feedback, improve work, etc. ✓	Opportunity for students to reflect on learning, respond to feedback, improve work, etc. ✓	Opportunity for students to reflect on learning, respond to feedback, improve work, etc. ✓	Opportunity for students to reflect on learning, respond to feedback, improve work, etc. ✓	Opportunity for students to reflect on learning, respond to feedback, improve work, etc. ✓
INCLUSION AND ACCESSIBILITY	All pupils should be able to engage with algorithms through carefully sequenced teaching and multiple representations of concepts. SEND considerations are as follows :Pre-teach technical vocabulary; Use visual aids and graphic organisers; Provide scaffolded worksheets and partially completed algorithms; Chunk learning into smaller sections; Use retrieval practice to reinforce learning; Reduce unnecessary reading demands; Use symbol-supported resources where appropriate; Model thinking processes aloud; Provide sentence starters and writing frames; Give clear, step-by-step instructions; Use checklists for algorithm development; Break tasks into shorter timed activities; Minimise cognitive overload by introducing one new concept at a time; Provide predictable lesson structures; Make expectations explicit; Use logical, rule-based contexts that often align well with algorithmic thinking; Ensure accessibility of devices and software; Use assistive technologies where required; Provide alternative methods for recording work; Start with real-life examples such as recipes, traffic lights, or morning routines before introducing formal algorithms; Use flowcharts, pseudocode, diagrams, videos, and practical programming activities; Break complex algorithms into smaller manageable steps; Provide opportunities for collaborative learning and discussion; Explicitly teach key vocabulary such as sequence, selection, iteration, variable, and trace; Use worked examples before independent practice; Allow pupils to demonstrate understanding through different formats, including verbal explanations, flowcharts, and code. To support Disadvantaged Pupils: Avoid assuming prior exposure to technology or programming; Explicitly teach background knowledge and technical language; Provide access to digital resources both in school and, where possible, at home; Use examples from diverse contexts and experiences; Build confidence through regular success opportunities; Ensure pupils can access intervention and additional support when needed; The focus should be on removing barriers rather than lowering expectations. Adaptive teaching ensures pupils can access learning while working towards the same ambitious curriculum goals: Scaffolding tasks rather than simplifying content; Providing worked examples before independent application; Offering varying levels of support during coding tasks; Using questioning tailored to pupil understanding; Providing visual representations alongside text-based explanations; Using live modelling and guided practice; Gradually removing support as confidence develops.					
SPECIALIST VOCABULARY	Escape sequences Data types Variables String formatting Relational operators Input/Output IF ELSE	Algorithms Boolean Pseudo code Abstraction Trace table Flowchart Input, Process, Output Decision	Function Procedure Parameter Argument Return Call Decomposition Len Min	Function Procedure Parameter Argument Return Call Decomposition Len Min	For loop While loop Lists Array Functions Subprograms	Reading and writing to a file Decomposition Readline Writeline Try except else



			Max Scope Global vs Local Types of errors – Syntax, Logic, Runtime	Max Scope Global vs Local Types of errors – Syntax, Logic, Runtime		
QUALITY FIRST TEACHING	✓ Strategies to learn more, remember more (metacognition) used in lessons e.g. retrieval, elaboration, interleaving, dual coding, etc. ✓ Differentiation and reasonable adjustments for students with SEND, EAL, etc. such as scaffolding, visual aids, audio, physical resources, planned questioning, etc. ✓ Opportunities for Literacy, Numeracy and Oracy, including a focus on reading ✓ Opportunities to apply key concepts and address misconceptions					



YEAR 11: Computer Science Theory

	Autumn 1	Autumn 2	Spring 1	Spring 2	Summer 1	
	UNIT OF WORK: <u>Algorithms</u>	UNIT OF WORK: <u>Programming</u>	UNIT OF WORK: <u>Logic and Languages</u>	UNIT OF WORK: <u>Revision</u>	UNIT OF WORK: <u>Revision</u>	UNIT OF WORK: <u>End of course</u>
KNOWLEDGE	Be able to produce structure diagrams to show: The structure of a problem Subsections and their links to other subsections Understand and use different types of searches and sorts including Linear and Binary searches and Bubble, Insertion and Merge Sort. Understand arithmetic operators and variables Define the data types integer, real, Boolean, character, string	Students will know and understand the outline of the program they have to code and will be able to brainstorm to decompose the problem. Understand and use data types: integer, real, Boolean, character and string. Declare and use constants and variables. Use input, output and assignment statements. Use random number generation Write algorithms in pseudocode involving sequences	Construct truth tables for the following logic gates: NOT, AND, OR Understand how to make maintainable programs including: Naming conventions and Indentations.	Recap subject knowledge for each topic using know more remember more strategies:	Recap subject knowledge for each topic using know more remember more strategies:	
KEY SKILLS	Understand the principles of computational thinking including: Abstraction, Decomposition and Algorithmic thinking. Be able to produce structure diagrams to show: The structure of a problem; Subsections and their links to other subsections; Understand and use different types of searches. Be able to apply each algorithm to a data set. Understand flowchart symbols and able to refine and correct algorithms and flowcharts. Be able to use Boolean operators. Write algorithms in pseudocode involving sequence, selection and iteration. Create and use of trace tables to follow an algorithm	Use arithmetic operators including MOD and DIV. Use string handling and conversion functions. Use selection and nested selection statements. Use NOT, AND and OR when creating Boolean expressions. Understand and use iteration in an algorithm Write algorithms in pseudocode involving sequence, selection and iteration. Use one- and two-dimensional arrays in the design of solutions to simple problems Understand the concept of subroutines. Understand and use basic file handling operations: open, read, write, close Use SQL (Structured Query Language) statements to search for data: Formulate criteria involving AND, OR and LIKE; Use SELECT, FROM, WHERE, ORDER BY statements (Use the wildcard *)	Construct truth tables for simple logic circuits. Create, modify and interpret simple logic circuit diagrams. Describe defensive design considerations: Input validation, Anticipating misuse; Authentication Understand how to make maintainable programs including: Commenting; Understand the purpose of testing including: Iterative testing; Final/terminal testing; Identify syntax and logic errors; Select and use suitable test data including: Normal, Boundary, Invalid and Erroneous. Understand the purpose of translators and describe the characteristics of a compiler and interpreter.	Study techniques enhanced Creating knowledge organisers Using exam reference language to write and read program code.	Study techniques enhanced Creating knowledge organisers Using exam reference language to write and read program code.	



CURRICULUM SEQUENCING AND HOW WE BUILD ON SKILLS AND KNOWLEDGE	Link to KS 3: Year 7, 8 and 9 Algorithms and Coding, Breaking down tasks.	Link to KS 3 & year 10: Python programming skills learned in years 7 – 10.	Link to KS 3: Year 7-10 Logic Gates	Revisit subject knowledge: Section 1 Section 2 Section 3 Section 4 Section 5 Section 6 Programming revision theory	Revisit subject knowledge: Section 1 Section 2 Section 3 Section 4 Section 5 Section 6 Programming revision theory	
WIDER APPLICATION AND FUTURE PATHWAYS	For Algorithms unit, you could explain the links to the wider world as follows: Careers: Algorithms are used in many careers, including: Software Developer; Data Scientist; Cyber Security Analyst; Artificial Intelligence Engineer; Game Developer; Robotics Engineer; Web and App Developer Professionals use algorithms to solve problems efficiently and create technology that people use every day. Future Study: Knowledge of algorithms provides a foundation for: A-Level Computer Science; BTEC Computing qualifications; University degrees in Computer Science, Software Engineering, Data Science, AI, and Cyber Security; Apprenticeships in digital and technology sectors Algorithms are a core concept in all advanced computing courses. Equality, Inclusion and Diversity: Algorithms affect many people: Inclusive algorithms consider the needs of different users; Developers work to reduce bias in algorithms that could unfairly disadvantage certain groups; Diverse teams often create better algorithms because they	For Programming in Python unit, here is how the topic links to the wider world: Careers: Learning Python develops skills used in many careers, including: Software Developer; Games Programmer; Data Analyst; Data Scientist; Cyber Security Analyst; Web Developer; Artificial Intelligence (AI) Engineer; Robotics Engineer; Network Engineer; IT Support and Systems Administrator. Python is one of the most widely used programming languages in industry, making it valuable for future employment. Future Study: Python provides a foundation for further qualifications, including: A Level Computer Science; T Levels in Digital Production, Design and Development; BTEC Computing courses; University degrees in Computing, Software Engineering, Data Science and Cyber Security; Apprenticeships in Digital and Technology sectors Many advanced computing courses use Python as a teaching language. Equality, Inclusion and Diversity: Programming is for everyone, regardless of gender, ethnicity, disability, religion or	The study of logic and languages underpins almost all modern digital technology. Every computer, smartphone, gaming console and smart device relies on logical operations and programming languages to function. Careers: Knowledge of logic and programming languages is relevant to many careers, including: Software Developer; Computer Programmer; Cyber Security Analyst; Systems Analyst; Web Developer; Games Developer; Artificial Intelligence Engineer; Robotics Engineer; Data Scientist; Embedded Systems Engineer; Network Engineer; Digital Forensics Investigator. Many technology careers require an understanding of how logical operations control computer systems and how programming languages are translated into machine-executable code. Future Study: This unit provides essential foundations for: GCSE Computer Science programming content; A Level Computer Science; T Levels in Digital Production, Design and Development; BTEC qualifications in Computing and IT; Computer Science degrees; Software Engineering degrees; Cyber Security qualifications;			



	bring different perspectives and experiences. This highlights the importance of representation in the technology industry. Employment: Many modern workplaces require problem-solving skills that relate to algorithmic thinking. Opportunities exist in: Banking and finance; Healthcare; Manufacturing; Transport; Retail; Scientific research; Weather forecasting and Online shopping and delivery tracking.	background. By: Python's simple syntax makes coding more accessible to beginners; Technology companies actively encourage participation from underrepresented groups in computing; Diverse teams create better software because they bring different perspectives and experiences; Programming can be adapted using accessibility tools such as screen readers, voice input and magnification software. Students learn that digital technology should be designed to be inclusive and accessible to all users. Employment: Employers value programming because it demonstrates: Problem-solving ability; Logical thinking; Persistence and resilience; Creativity; Attention to detail and Ability to work on complex projects. Python skills can support careers in technology, science, finance, engineering, healthcare and many other sectors.	Apprenticeships within digital and technology sectors. Equality, Inclusion and Diversity: The technology sector benefits from diverse perspectives and inclusive practices. Students learn that: Computing is accessible to individuals from all backgrounds; Diverse development teams create better and more inclusive technological solutions; Software should be designed to meet the needs of a wide range of users; Accessibility features within software rely on logical programming structures; Programming languages are used globally by people from different cultures, backgrounds and experiences; The unit helps challenge stereotypes surrounding who can work in computing and technology. Employment: Employers value the skills developed through studying logic and languages, including: Problem solving; Analytical thinking; Precision and accuracy; Attention to detail; Systematic reasoning; Troubleshooting and debugging skills. These transferable skills are useful not only in technology careers but also in engineering, finance, science, healthcare and many other professions.			
--	--	---	---	--	--	--



ASSESSMENTS Summative and Formative as applicable	Formative: Self and peer assessment Class discussions Classwork and homework Exit tickets Think pair share Quizzes Mnemonics Summative: End of unit test	Formative: Self and peer assessment Class discussions Classwork and homework Exit tickets Think pair share Quizzes Mnemonics Summative: End of unit test Independent programming challenges.	Formative: Self and peer assessment Class discussions Classwork and homework Exit tickets Think pair share Quizzes Mnemonics Summative: End of unit test Independent programming challenges.	Retrieval practice Interleaving Spaced practice Dual coding Quizzes Summative Revision tests	Retrieval practice Interleaving Spaced practice Dual coding Quizzes Summative Revision tests	
FEEDBACK SUPPORTS LEARNING	Opportunity for students to reflect on learning, respond to feedback, improve work, etc. 	Opportunity for students to reflect on learning, respond to feedback, improve work, etc. 	Opportunity for students to reflect on learning, respond to feedback, improve work, etc. 	Opportunity for students to reflect on learning, respond to feedback, improve work, etc. 	Opportunity for students to reflect on learning, respond to feedback, improve work, etc. 	Opportunity for students to reflect on learning, respond to feedback, improve work, etc. 
INCLUSION AND ACCESSIBILITY	All pupils should be able to engage with algorithms through carefully sequenced teaching and multiple representations of concepts. SEND considerations are as follows :Pre-teach technical vocabulary; Use visual aids and graphic organisers; Provide scaffolded worksheets and partially completed algorithms; Chunk learning into smaller sections; Use retrieval practice to reinforce learning; Reduce unnecessary reading demands; Use symbol-supported resources where appropriate; Model thinking processes aloud; Provide sentence starters and writing frames; Give clear, step-by-step instructions; Use checklists for algorithm development; Break tasks into shorter timed activities; Minimise cognitive overload by introducing one new concept at a time; Provide predictable lesson structures; Make expectations explicit; Use	Programming in Python should be accessible to all learners through carefully planned teaching, scaffolding and support. SEND considerations are: Provide step-by-step instructions and checklists; Reduce cognitive load by introducing one concept at a time; Use partially completed programs and code templates; Recap key vocabulary regularly; Allow additional processing time; Pre-teach technical vocabulary such as variable, iteration and algorithm; Use visual supports and coding diagrams; Provide opportunities for discussion before independent coding; Use consistent programming terminology; Offer structured routines and predictable lesson sequences; Celebrate small successes to build confidence; Use collaborative programming activities with clear roles; Chunk tasks into achievable milestones; Ensure compatibility with assistive technologies including screen readers and speech-to-text software; Use adjustable	The Logic and Languages unit can be made accessible through clear explanations, visual representations and structured learning opportunities. SEND considerations are: Teach one concept at a time to minimise cognitive overload; Provide scaffolded truth tables and partially completed examples; Use step-by-step guidance when introducing logic gates and Boolean expressions; Provide knowledge organisers and vocabulary lists; Use repetition and retrieval activities to reinforce learning; Explicitly teach subject-specific vocabulary such as compiler, interpreter, Boolean, operand and logic gate; Support explanations with visuals and symbols; Use sentence starters and discussion prompts; Check understanding regularly through questioning and verbal explanations; Establish predictable lesson structures and routines; Break tasks into manageable sections with clear success criteria; Celebrate			



logical, rule-based contexts that often align well with algorithmic thinking; Ensure accessibility of devices and software; Use assistive technologies where required; Provide alternative methods for recording work; Start with real-life examples such as recipes, traffic lights, or morning routines before introducing formal algorithms; Use flowcharts, pseudocode, diagrams, videos, and practical programming activities; Break complex algorithms into smaller manageable steps; Provide opportunities for collaborative learning and discussion; Explicitly teach key vocabulary such as sequence, selection, iteration, variable, and trace; Use worked examples before independent practice; Allow pupils to demonstrate understanding through different formats, including verbal explanations, flowcharts, and code. To support Disadvantaged Pupils: Avoid assuming prior exposure to technology or programming; Explicitly teach background knowledge and technical language; Provide access to digital resources both in school and, where possible, at home; Use examples from diverse contexts and experiences; Build confidence through regular success opportunities; Ensure pupils can access intervention and additional support when needed; The focus should be on removing barriers rather than lowering expectations. Adaptive teaching ensures pupils can access learning while working towards the	screen settings and accessible programming environments; Allow alternative input devices where required; Provide printed resources in accessible formats if needed. Disadvantaged Pupils are catered for by: To ensure equity of opportunity; Avoid assuming prior experience of coding; Explicitly teach key knowledge and vocabulary; Provide access to software and devices during school time; Offer coding clubs, intervention sessions or homework support where possible; Use culturally diverse examples and contexts; Provide revision materials and learning resources that can be accessed outside lessons; Regularly check understanding and address misconceptions early and Build pupils' confidence and sense of belonging within computing. The focus should be on removing barriers rather than lowering expectations. Adaptive teaching supports pupils to achieve the same learning goals through different routes. Examples include: Simplify instructions while maintaining challenge; Introduce concepts through concrete examples before abstraction; Provide vocabulary mats and knowledge organisers; Use guided practice before independent programming; Offer varying levels of scaffolding; Allow pupils to solve problems using different approaches; Use pair programming and peer support; Code templates and skeleton programs; Colour-coded examples to highlight key programming structures;	progress and effort; Use paired and group activities with clearly defined roles; Provide opportunities for success before increasing challenge; Ensure electronic and printed resources are accessible; Use high-contrast diagrams and enlarged visuals where appropriate; Allow access to assistive technology such as screen readers or speech-to-text software; Provide alternative methods of recording responses where needed. Disadvantaged Pupils: Do not assume prior knowledge of computing terminology; Explicitly teach and revisit key vocabulary; Provide access to high-quality revision materials and learning resources; Use examples that reflect a wide range of cultures, experiences and industries; Monitor participation and understanding regularly; Support literacy demands by modelling technical language; Offer structured opportunities for practice and revision; Build confidence by recognising success and progress. The focus should remain on enabling access to the full curriculum rather than reducing expectations. Adaptive teaching: Introduce concepts through everyday examples before moving to abstract theory; Simplify language without simplifying the knowledge; Use visual representations to support understanding of Boolean logic and truth tables; Model worked examples before independent tasks; Use guided practice and collaborative activities; Allow pupils to explain concepts			
--	--	---	--	--	--



	same ambitious curriculum goals: Scaffolding tasks rather than simplifying content; Providing worked examples before independent application; Offering varying levels of support during coding tasks; Using questioning tailored to pupil understanding; Providing visual representations alongside text-based explanations; Using live modelling and guided practice; Gradually removing support as confidence develops.	Debugged starter code for pupils requiring additional support; Video demonstrations that can be paused and replayed; Allow pupils to demonstrate understanding through explanations, annotations, debugging or completed code; Provide extension challenges for pupils who master concepts quickly.	verbally before recording written answers; Provide additional processing time where required; Use scaffolded truth tables and logic gate diagrams; Provide vocabulary mats and knowledge organisers; Offer colour-coded resources to highlight relationships between concepts; Use interactive simulations to demonstrate logic circuits and language translation processes; Allow pupils to demonstrate understanding through: Written answers, Verbal explanations, Diagram completion, Practical activities, Matching and sorting tasks.			
SPECIALIST VOCABULARY	Computational thinking, abstraction, decomposition, algorithmic thinking, inputs, processes, outputs, structure diagrams, pseudocode, flowcharts, reference language, trace tables, syntax error, logical error, algorithm, decision, terminal, sub program, process, binary search, linear search, bubble sort, merge sort, insertion sort, variables, constants, operators, assignments, sequence, selection, iteration, Boolean operators, arithmetic operators, modulus, quotient, exponentiation	Functions, Procedures, Iteration, Flowcharts, Pseudo code, Test tables, Variables, constants, operators, inputs, outputs, assignment, sequence, selection, iteration, arithmetic operators, Boolean operators, AND, OR, NOT, ==, !=, <, <=, >, >=, +, -, *, /, MOD, DIV, ^, exponentiation, data types, integer, real, Boolean, character, string, casting, string manipulation, file handling, open, read, write, close, records, SQL, arrays, one-dimensional array, two-dimensional array, sub program/subroutine, functions, random numbers, concatenation, slicing, SQL, SELECT, FROM, WHERE.	Defensive design, anticipating misuse, authentication, validation, maintainability, sub programs, naming conventions, indentation, commenting, testing, iterative testing, final/terminal testing, syntax, syntax error, logic error, test data, normal, boundary, invalid, erroneous, test plan, AND, OR, NOT, truth table, logical operators, logic gates, logic diagrams, conjunction, disjunction, negation, high-level language, low-level language, translators, compiler, interpreter, compiler, interpreter, Integrated Development Environment (IDE), editors, error diagnostics, run-time environment.	See year 10 Curriculum map for keywords related to each topic.	See year 10 Curriculum map for keywords related to each topic.	
QUALITY FIRST TEACHING	✓ Strategies to learn more, remember more (metacognition) used in lessons e.g. retrieval, elaboration, interleaving, dual coding, etc. ✓ Differentiation and reasonable adjustments for students with SEND, EAL, etc. such as scaffolding, visual aids, audio, physical resources, planned questioning, etc. ✓ Opportunities for Literacy, Numeracy and Oracy, including a focus on reading ✓ Opportunities to apply key concepts and address misconceptions					