

Werneth DEPARTMENT Curriculum KS4

Year 10	Year 11
KS4 is split on a 2:1 ratio. During Year 10, 1 lesson a week will be dedicated to Python Programming and the other 2 lessons cover the content of Component 1 (Paper 1). During Year 11 this is swapped and 2 lessons will be dedicated to Component 2 and 1 lesson will be dedicated to recapping Component 1. The KS4 curriculum does not guarantee that each scheme of work will perfectly align with the end of a term. For example 1.2 Memory and Storage will finish mid way through Autumn 2. This document is generalised to understand content being delivered in a roughly estimated timeframe. For specific week by week lessons please ask for more information and I will forward the week by week plans for KS4 connor.haughton@wernethschool.com	

Autumn 1	Content (Why)	Develop understanding of Memory and Storage and how different types of data are represented and stored within computer systems. Pupils will work with binary and hexadecimal, explore character, image and sound representation, and understand how storage capacity is measured.	Develop understanding of Algorithms through computational thinking, algorithm design and algorithmic problem-solving. Pupils will explore abstraction, decomposition and algorithmic thinking before designing, interpreting and refining algorithms using pseudocode, flowcharts and trace tables. They will also investigate standard searching and sorting algorithms and compare their efficiency.
	Content (What)	This unit provides essential knowledge of how computers represent, process and store information. It develops pupils' numerical and technical understanding of data representation while preparing them for more advanced study of computer systems, memory, storage and digital media.	This unit develops pupils' ability to solve problems logically and systematically. Understanding how algorithms are designed, tested and evaluated provides a foundation for programming and enables pupils to assess different approaches based on accuracy and efficiency.
	Acquire (Knowledge)	• Computers represent numerical data using binary and hexadecimal, with pupils converting between binary, denary and hexadecimal and performing binary addition accurately. • Different types of digital information require different methods of representation, with pupils understanding character encoding using ASCII and Unicode, alongside image and sound representation through pixels, resolution, colour depth, sampling rate and bit depth. • Digital storage capacity is measured using standard units, with pupils understanding the relationships between bits, nibbles, bytes, kilobytes, megabytes, gigabytes and terabytes and how these relate to storage requirements.	• Computational thinking simplifies complex problems into manageable solutions, with pupils using abstraction to remove unnecessary detail, decomposition to break problems into smaller components and algorithmic thinking to develop logical sequences of instructions. • Algorithms can be represented and tested using different methods, with pupils designing and interpreting pseudocode and flowcharts and using trace tables to identify, investigate and correct errors. • Searching and sorting algorithms use different approaches and have different efficiencies, with pupils understanding how linear and binary searches operate alongside bubble, insertion and merge sort.
	Apply (Skills)	Convert values between binary, denary and hexadecimal and perform binary calculations. Calculate and compare the storage requirements of different types of	Design, interpret and refine algorithms using pseudocode and flowcharts, applying trace tables to test solutions and identify errors. Implement and compare

		digital data, applying knowledge of character encoding, image resolution and colour depth, and sound sampling and bit depth to explain how digital information is represented and stored.	searching and sorting algorithms, explaining how they operate and evaluating their relative efficiency when solving problems.
Autumn 2	Content (Why)	Develop understanding of System Architecture , focusing on the structure and operation of the CPU within a computer system. Pupils will explore the Von Neumann architecture, CPU components, the Fetch–Decode–Execute cycle, factors affecting processor performance and the role of embedded systems.	Develop understanding of Creating Robust Programs , focusing on designing, testing, debugging and maintaining reliable software. Pupils will explore defensive design, authentication, different types of test data, syntax and logic errors, debugging techniques and programming practices that improve readability and maintainability.
	Content (What)	This unit provides a detailed understanding of how computers process instructions and data. It develops pupils' ability to explain the interaction between CPU components and memory while evaluating how architectural features influence system performance.	This unit develops pupils' ability to produce programs that are reliable, secure and fit for purpose. It reinforces the importance of anticipating potential misuse, systematically testing solutions and writing maintainable code that can be understood, modified and improved over time.
	Acquire (Knowledge)	• The CPU processes instructions and data using interconnected components, with pupils understanding the roles of the ALU, Control Unit, cache and registers including the Program Counter, Accumulator, MAR and MDR. • The Von Neumann architecture stores instructions and data in the same memory, with pupils understanding how the Fetch–Decode–Execute cycle enables instructions to be processed. • CPU performance is influenced by several architectural factors, including clock speed, number of cores and cache size, while embedded systems are designed to perform specific functions within larger devices.	• Robust programs anticipate errors and misuse, with pupils understanding defensive design techniques such as validation, authentication and anticipating inappropriate or unexpected user input. • Systematic testing identifies errors and confirms program reliability, with pupils understanding iterative and final testing and using normal, boundary and erroneous test data appropriately. • Errors can be identified, corrected and prevented through effective programming practice, with pupils distinguishing between syntax and logic errors and using debugging techniques alongside meaningful names, comments and consistent indentation to maintain code.
	Apply (Skills)	Trace and explain the Fetch–Decode–Execute cycle, accurately describing the roles of the CPU components and registers involved. Compare how clock speed, number of cores and cache size can affect performance, and identify suitable applications for general-purpose and embedded computer systems.	Design and develop programs that anticipate invalid input and potential misuse, incorporating appropriate validation and authentication. Create and apply systematic test plans using normal, boundary and erroneous data, identify and debug syntax and logic errors, and maintain code using clear naming, comments and consistent formatting to support future development.
Spring 1	Content (Why)	Develop understanding of Computer Networks alongside continued Python Programming . Pupils will explore LANs, WANs and the structure of the internet, including network hardware, transmission media, topologies, protocols, addressing, DNS and cloud computing, while continuing to develop practical Python programming skills.	Consolidate knowledge of Networking and Network Security while developing understanding of Computational Thinking, Algorithms and Programming . Pupils will recap network structures, hardware, protocols and addressing before exploring common cyber security threats and preventative measures. They will also apply computational thinking to algorithm design and programming problems.

Werneth DEPARTMENT Curriculum KS4

	Content (What)	This unit develops pupils' understanding of how computers communicate and how modern networks enable the internet and cloud-based services. Alongside this theoretical knowledge, continued Python programming provides opportunities to apply computational thinking and develop practical coding fluency.	This unit consolidates key knowledge required to understand how modern computer networks operate and how they can be protected from security threats. It also strengthens pupils' ability to use computational thinking to analyse problems, design algorithms and develop effective programming solutions.
	Acquire (Knowledge)	• Networks connect devices to enable communication and resource sharing, with pupils understanding LANs, WANs, network hardware, transmission media and topologies such as star and mesh. • Network communication relies on protocols and addressing systems, with pupils understanding the purpose of TCP/IP, HTTP and FTP alongside IP and MAC addresses, network layers and DNS. • The internet is a network of interconnected systems and services, with pupils understanding how the internet operates at a global scale and how technologies such as cloud computing provide remote access to data and applications.	• Computer networks use hardware, protocols and addressing systems to communicate, with pupils consolidating knowledge of LANs, WANs, network topologies, Ethernet, Wi-Fi, TCP/IP, HTTP, FTP, IP and MAC addresses, DNS and cloud computing. • Networks face a range of security threats that require appropriate preventative measures, including malware, social engineering, brute-force attacks, denial-of-service attacks and data interception, with pupils understanding measures such as firewalls, encryption, anti-malware, penetration testing, access levels and strong passwords. • Computational thinking supports effective algorithm and program design, with pupils applying decomposition, abstraction, pattern recognition and algorithmic thinking to break down problems and develop structured solutions.
	Apply (Skills)	Identify and explain the purpose of network hardware, transmission media, topologies and protocols, and analyse how data is communicated across networks and the internet. Explain the roles of IP and MAC addresses, DNS and cloud computing, while applying computational thinking and Python programming skills to develop, test and refine solutions to programming problems.	Explain and evaluate how networks operate and how security threats can be identified, prevented and mitigated. Apply computational thinking to analyse unfamiliar problems, design and refine algorithms, and translate solutions into effective programs, testing and debugging them to ensure they meet the required outcomes.
Spring 2	Content (Why)	Develop understanding of Network Security alongside continued Python Programming . Pupils will investigate common network security threats, including malware, social engineering, brute-force attacks, denial-of-service attacks and data interception, before exploring methods used to prevent and mitigate these threats.	Consolidate knowledge of Computer Systems, Ethical, Environmental and Legal Issues and Python Programming . Pupils will revisit CPU architecture, the Von Neumann model and the Fetch–Decode–Execute cycle, alongside the wider ethical, legal, cultural and environmental impacts of digital technology. Python programming will continue to develop pupils' ability to apply computational thinking and programming skills.
	Content (What)	This unit develops pupils' understanding of the risks faced by computer networks and the measures used to protect systems, data and users. Continued Python programming strengthens computational thinking and provides regular opportunities to develop, test and refine practical programming solutions.	This unit consolidates core knowledge needed to understand how computer systems operate and how digital technology affects individuals, society and the environment. It also develops pupils' ability to consider the legal and ethical responsibilities associated with technology while continuing to strengthen their practical programming skills.

	Acquire (Knowledge)	Computer networks are vulnerable to a range of security threats, including viruses, ransomware, phishing, brute-force attacks, denial-of-service attacks and data interception. Different security measures protect networks from different threats, including anti-malware software, firewalls, encryption, penetration testing, strong passwords and appropriate user access levels. Physical and user security are important parts of network protection, with pupils understanding how physical security measures and responsible user practices can reduce the risk of unauthorised access and data loss.	Computer systems process instructions through interconnected hardware, with pupils consolidating knowledge of the CPU, ALU, Control Unit, cache, registers, Von Neumann architecture, the Fetch–Decode–Execute cycle and factors affecting CPU performance. Digital technology creates ethical, legal, cultural and environmental impacts, with pupils understanding issues including data privacy, e-waste, energy consumption and emerging technologies, alongside UK legislation such as the Data Protection Act, Computer Misuse Act and Copyright, Designs and Patents Act. Software and digital technology operate within legal and ethical frameworks, with pupils understanding privacy, software licensing and the distinction between open-source and proprietary software, while continuing to develop effective Python programming solutions.
	Apply (Skills)	Identify and evaluate network security threats and select appropriate preventative measures based on the nature of the threat. Explain how technical, physical and user-based security controls work together to protect networks and data, while applying Python programming skills to develop, test and debug solutions to defined problems.	Explain and evaluate how computer systems operate and how digital technology affects individuals, society and the environment. Apply knowledge of UK legislation and software licensing to real-world scenarios, making reasoned judgements about ethical and legal issues. Continue to design, write, test and debug Python programs, applying appropriate programming constructs to solve problems effectively.
Summer 1	Content (Why)	Develop understanding of System Software through the study of operating systems and utility software. Pupils will explore how system software manages hardware and supports applications, including user interfaces, memory, multitasking, peripherals, users and files. They will also investigate common utility software such as encryption, defragmentation and data compression, alongside continued Python programming.	Targeted Tricky Topics lessons based on question-level analysis (QLA), identifying and addressing individual gaps in pupils' knowledge and understanding.
	Content (What)	This unit develops pupils' understanding of the software that enables computer systems to operate effectively. It helps pupils understand the relationship between operating systems, hardware and applications, while developing their ability to evaluate how different system and utility software supports users and protects or manages data.	This approach ensures revision is responsive to pupils' specific needs rather than relying on generic revision. It allows teachers to address misconceptions, strengthen weaker areas and maximise progress by focusing lesson time on the knowledge and skills pupils need most.
	Acquire (Knowledge)	Operating systems manage computer hardware and provide services for applications and users, including memory management, multitasking, peripheral management, file management and user management. Different user interfaces provide different ways to interact with a computer system, including graphical, command-line, touch and voice interfaces, each offering advantages and disadvantages for different users and purposes.	QLA identifies specific areas of weakness, enabling teachers and pupils to recognise gaps in knowledge and misconceptions. Targeted intervention can address individual learning needs, with pupils revisiting difficult concepts and strengthening their understanding through focused teaching and practice. Effective revision requires retrieval, practice and feedback, allowing

Werneth DEPARTMENT Curriculum KS4

		• Utility software performs specific maintenance and security functions, with pupils understanding how encryption, defragmentation and data compression can protect, organise and optimise computer systems. •	pupils to identify errors, correct misconceptions and build confidence in previously challenging areas.
	Apply (Skills)	Explain how operating systems manage hardware, software and users, and compare different types of user interface for specific applications. Evaluate the purpose of utility software and select appropriate tools for maintaining, securing and managing data, while continuing to apply Python programming skills to develop, test and refine solutions.	Use QLA information to identify individual and common areas of weakness and deliver targeted lessons accordingly. Pupils complete focused retrieval, practice and exam-style activities before using feedback to correct errors and demonstrate improved understanding of previously identified tricky topics.
Summer 2	Content (Why)	Develop understanding of Ethical, Environmental and Legal Issues in Computing alongside continued Python Programming . Pupils will explore the impact of digital technology on individuals, society, culture and the environment, including data privacy, emerging technologies, UK legislation, e-waste, energy consumption and software licensing.	Examination Revision
	Content (What)	This unit develops pupils' ability to understand and evaluate the wider consequences of digital technology. It enables pupils to make informed judgements about ethical and legal issues while recognising their responsibilities as users and creators of technology. Continued Python programming strengthens computational thinking and practical problem-solving skills.	Examination Revision
	Acquire (Knowledge)	• Digital technology creates ethical and societal issues, with pupils exploring data privacy, emerging technologies, cultural change and the moral implications of how technology is developed and used. • Technology operates within legal frameworks, with pupils understanding key UK legislation including the Data Protection Act, Computer Misuse Act and Copyright, Designs and Patents Act. • Digital technology has environmental and licensing implications, with pupils understanding e-waste, energy consumption, privacy and the differences between open-source and proprietary software licences.	Examination Revision
	Apply (Skills)	Analyse real-world scenarios involving the ethical, legal, cultural and environmental impacts of digital technology, making justified judgements about appropriate actions and responsibilities. Apply knowledge of UK legislation and software licensing to evaluate situations involving technology, while continuing to design, write, test and debug Python programs to solve defined problems.	Examination Revision

