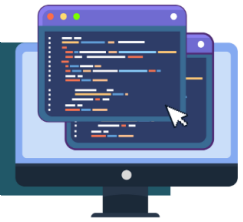
COMPUTER SCIENCE CURRICULUM MAP
2026 – 2027



Intent:

Computer Science at Wembley High Technology College balances knowledge and practical application in order to provide students with both specialised and transferable skills that are greatly valued in the marketplace. All students will experience physical computing to develop and hone their ability to code through the completion of increasingly complex programming projects.

The course emphasises three distinct strands within computing extracted from the KS3 National Curriculum: computer science (CS), information technology (IT) and digital literacy (DL). Each component is essential in preparing our students to thrive in an increasingly digital world. Computer science incorporates techniques and methods for solving problems and includes a distinct way of thinking and working that complements other disciplines.

Implementation:

Year	Half Term 1	Half Term 2	Half Term 3	Half Term 4	Half Term 5	Half Term 6
7	**Digital Foundations & PowerPoint** Students begin with safe and responsible use of the ICT suite, school accounts and strong passwords. They develop confidence with MS Teams, Edulink and Outlook, professional email etiquette, Windows desktop navigation and file management. Online safety is revisited through digital footprints, privacy, cyberbullying, phishing and online reputation. The half term concludes with core PowerPoint skills, including themes, layouts, text formatting, images, icons and shapes.	**PowerPoint Project & Computer Systems** Students refine presentation design through consistent themes and transitions before completing an independent PowerPoint project and evaluating their work. Following assessment and feedback, they are introduced to computer systems by distinguishing hardware from software, classifying input and output devices, and identifying internal and external hardware components. Career link: Digital Marketing Assistant.	**Inside a Computer & Farmbots** Students deepen their understanding of computer systems by studying CPU structure, the ALU, Control Unit, registers and the Fetch-Decode-Execute cycle, followed by RAM, ROM and primary and secondary storage. They then begin text-based programming through TuringLab Farmbots, using movement algorithms, functions and arguments, debugging syntax, logic and name errors, writing maintainable code and comments, and completing code-from-scratch challenges.	**Farmbots: Iteration & Independent Programming** Students extend their Farmbots programming by using iteration and for loops to reduce repeated code and improve efficiency. After assessment and feedback, they apply sequencing, loops, debugging and problem-solving independently in consolidation challenges, planning, writing, testing and evaluating complete programs.	**Networks, the Internet & Connected Devices** Students learn what computer networks are and why they are used, comparing wired and wireless networks and identifying networking hardware. They explore how the Internet works, distinguish the Internet from the World Wide Web, use browsers and URLs safely, and develop effective online searching and source-reliability skills. The unit then introduces the Internet of Things and common network threats, including malware and social engineering. Career link: Network Engineer.	**Network Security & MakeCode Arcade** Students examine how networked systems are protected using measures such as firewalls, encryption, access control, patching and anti-malware, and evaluate security measures in context. Following assessment and feedback, they create a game in MakeCode Arcade using sprites, events, sequencing, selection, variables, loops and conditions. They test, debug, evaluate and peer-review their finished game. Career link: Games Developer.

8	**Digital Productivity & Smart Cities Python** Students revisit safe ICT-suite use before developing spreadsheet skills in Excel: formatting, cell references and ranges, arithmetic formulae, AutoFill and functions including SUM, AVERAGE, MIN and MAX. They then begin Python programming through TuringLab Smart Cities, using variables, data types, print(), selection, comparison operators and increasingly independent conditional algorithms.	**Complex Selection & Computational Thinking** Students extend Smart Cities programming with if-else, if-elif-else, nested conditions, logical operators and independent algorithm challenges. Following assessment and feedback, they develop computational thinking through abstraction and decomposition, then apply algorithmic thinking with Lightbot using sequences, procedures and loops.	**Introduction to Python Programming** Students transition from block-based programming to text-based Python using Turtle. They learn Python syntax and the IDE, coordinates and colour, variables, for loops and reusable functions, applying decomposition and debugging as programs become more complex. The unit builds confidence in writing, tracing, testing and improving Python programs.	**Python Turtle Project & Smart Cities Loops** Students apply their Python Turtle knowledge independently to a mini-project using functions, loops, colour, comments, testing and evaluation. Following assessment and feedback, they return to Smart Cities to use loops with index values, write DRY code, combine iteration with conditions and proximity data, and debug algorithms written from scratch.	**Searching, Sorting & Binary Representation** Students study core searching and sorting algorithms, including linear search, binary search, bubble sort, merge sort and insertion sort. They trace algorithms manually, compare efficiency and select suitable methods for different scenarios. The half term also introduces digital character representation through ASCII and Morse code and develops understanding of binary representation. Career link: Software Engineer.	**Data Storage & MakeCode Arcade** Students extend data representation by comparing ASCII and Unicode, calculating simple text file sizes and working with data-storage units from bits and bytes through KB, MB, GB and TB. Following assessment and feedback, they design and develop a MakeCode Arcade game using sprites, events, variables, selection, loops and conditions, then test, debug, evaluate and showcase the finished product.
9	**Web Design & HTML** Students revisit safe ICT-suite use before exploring effective web-design principles, usability, accessibility, navigation and site maps. They then build web pages using HTML structure, headings, paragraphs, ordered and unordered lists, images and alt text, hyperlinks and basic inline styling. Practical work develops understanding of tags, elements, attributes, nesting and how pages render in a browser.	**CSS & Computer Systems** Students extend their websites using external CSS, element, class and ID selectors, common properties, the box model, margin, padding, borders and flexbox. Following assessment and feedback, they study the Von Neumann stored-program concept, CPU components and registers, the Fetch-Decode-Execute cycle, and factors affecting CPU performance including clock speed, cores and cache.	**Memory, Storage & Data Representation** Students compare RAM and ROM and evaluate secondary storage technologies including HDD, SSD and optical storage. They then develop data-representation knowledge through binary and denary conversion, hexadecimal, ASCII and Unicode character encoding, and bitmap images, including calculations involving file size, colour depth and resolution.	**Sound, Compression & Boolean Logic** Students complete their data-representation work by explaining how sound is represented digitally, calculating sound file sizes and comparing lossy and lossless compression. Following assessment and feedback, they evaluate Boolean expressions and truth tables using logic gates including NOT, AND, OR, NAND, NOR and XOR, then perform binary addition and explain overflow.	**Python Programming** Students develop Python programming through variables and data types, input and formatted output, selection with if/elif/else, for and while loops, functions with parameters and return values, lists, string manipulation and text-file handling. Practical tasks require students to write increasingly independent programs and apply decomposition, iteration, data structures and persistent storage.	**Python Project & HTML/CSS Website Project** Students consolidate Python by completing an independent mini-project that applies concepts learned across the programming unit. Following assessment and feedback, they return to web development to build a multi-page HTML and CSS website with consistent navigation, images and alt text, reusable styling and structured content. They test links, images and styles, complete peer review and evaluate the finished website.

10	**Systems Architecture & Primary Memory** Students begin OCR J277 Component 01 with CPU purpose and components, registers and the Fetch-Decode-Execute cycle, CPU performance and embedded systems. They then study RAM, ROM and virtual memory, with topic assessment and OCR-style exam practice embedded throughout.	**Memory, Storage & Number Representation** Students study the need for secondary storage and compare magnetic, optical and solid-state technologies using capacity, speed, portability, durability, reliability and cost. They then develop data-representation skills through units of data, binary numbers, binary addition and overflow, and hexadecimal.	**Data Representation & Networks** Students complete data representation through character sets, ASCII and Unicode, bitmap images, sound and compression, followed by assessment. They then begin networks by comparing LANs and WANs, factors affecting network performance, and client-server and peer-to-peer models.	**Networks & Protocols** Students extend their understanding of networks through routers, switches, wireless access points and NICs; wired and wireless connections; the Internet, DNS and hosting; common protocols including TCP/IP, HTTP/S, FTP, POP, IMAP and SMTP; and the purpose and advantages of layered protocols.	**Network Security & Operating Systems** Students consolidate networks through assessment before studying security threats including malware, social engineering, brute-force attacks, DoS, interception and SQL injection. They examine vulnerability identification and prevention methods, complete a security assessment, and begin operating-system functions.	**Systems Software, Ethical, Legal & Environmental Impacts** Students study utility software and complete the systems-software unit before examining ethical, cultural and environmental impacts of digital technology. They apply the Data Protection Act 2018, Computer Misuse Act 1990 and Copyright, Designs and Patents Act 1988 to scenarios and compare software licensing models.
11	**Computer Systems – Revision and exam techniques** Students will be supported at the start of the year to recall prior learning and revisit content from Year 10 on key topics to strengthen their knowledge and understanding using technical vocabulary on the following topics: • Systems Architecture • Memory & Storage • Networks • Network Threats • Systems Software • Ethical, legal, cultural and environmental impacts *Students will develop discussions and debates on key topics that link to implications of digital technology.* Students will also have an opportunity to practice exam questions using past papers.		**Computational thinking, Algorithms & Programming -Revision and exam techniques** Students will be supported at the start of the year to recall prior learning and revisit content from Year 10 on key topics to strengthen their knowledge and understanding using technical vocabulary on the following topics: • Algorithms • Programming languages • Programming fundamentals • Producing robust programs • Boolean logic *Students will revisit practical skills and recall key definitions.* Students will also have an opportunity to practice exam questions using past papers.			

Enrichment Opportunities:

Year 7 Coding Club –Passionate coders opt in to attend weekly coding club sessions taking place in an ICT suite at lunchtimes. The programme is designed to aspire students to develop their text-based programming as well as online simulators. Students work towards completing mini projects which are celebrated in the school during assemblies.

Impact:

KS3 units are adapted and taught purely by topic or with a consideration of underpinning knowledge and skills from KS2, which forms part of the unit delivery from the Year 7 baseline assessment. From this, the programme of study is built on from the previous topic and links to the next topic. Key concepts such as programming, Computational thinking, algorithms and systems architecture are weaved into the KS3 as a starting point to help students familiarise with the technical vocabulary before they start KS4 and build onto complex content.

Formative assessment is an integral part of our approach to Teaching and Learning. Over the course of their study, all lessons use and apply formative assessment using questions, which is displayed, on the board or on paper, class teacher observation and recall retrieval activity/consolidation of learning. Examples of formative assessment, which is used, is self and peer assessment, different forms of questioning, quizzes to support learning and self-explanations or rubrics for assessing projects. Carefully chosen questions (diagnostic questions) are used to reveal student's misconceptions about the key principles and helps the Computer Science teacher understand the students' conception of the key topic. During practical programming lessons, multiple choice questions are used to assess programming skills by using snippets of block/text based code, which students will need to read and work out what the outcome of the running the snippets will be. Questions like these highlight that it is important for students to engage in code comprehension and code reading activities when learning to program.

Students will also sit a summative assessment at the end of each topic or unit at the end of the term. This assessment will be cumulative and will assess not only what the students have learned over the previous term, but also their understanding of all relevant material previously taught. Summative assessments consist of vocabulary to ensure students can speak the 'language' of Computer Science, students can fill in a missing component in a block of code or determine the output of a block of code and finally students can find and correct errors. Staff are supported to mark these accurately and post assessment moderation takes place to ensure the validity of the data. All data is analysed centrally (not by teachers) and the CS Subject Leader is given a report outlining the areas of strength and weakness. This is then used to inform future planning, pedagogical action/support with additional interventions and set changes.

Homework will be set for students using a combination of online as well as paper-based activities on the current and previous learning content with the addition of written extended exam questions at KS4.