Year 8 Curriculum – 2023-2024						
	Autumn Term		**Spring Term**		**Summer Term**	
	1	**2**	**1**	**2**	**1**	**2**
Key Concepts	**Computer Science**	**Computer Science**	**Computer Science / Algorithms & Programming**	**Algorithms & Programming**	**Information Technology**	**Computer Science**
National Curriculum Knowledge & Understanding	**Creating webpages:** Pupils will use a text-based programming language (HTML) to design and create a website. They will create, reuse, revise and repurpose digital artefacts for a given audience.	**Binary bits and bobs:** Pupils will understand simple Boolean logic [for example, AND, OR and NOT] and some of its uses in circuits and programming. They will understand how numbers can be represented in binary, and be able to carry out simple operations on binary numbers such as binary addition, and conversion between binary, Hex and decimal.	**Computational Thinking:** Pupils will understand several key algorithms that reflect computational thinking and use logical reasoning to compare the utility of alternative algorithms for the same problem.	**Introduction to Python:** Pupils will use a text-based programming language to solve a variety of computational problems. They will design and develop modular programs that use procedures or functions.	**Mobile app development:** Pupils will design, use and evaluate computational abstractions that model the state and behaviour of real-world problems and physical systems.	**Networks:** Pupils will understand the hardware that makes up computer systems, either as a standalone device or a series of networked devices, and how they communicate with one another and with other external systems. They will understand how data is transferred and processed

						within a computer system.
Assessment	Assessment portfolio	Written assessment	Written assessment	Assessment portfolio	Create an online gaming app – assess coding skills.	Written assessment
Why this?	Pupils have used simple text-based statements when coding in Small Basic previously. HTML allows pupils to understand and experience the building blocks of a webpage and CSS enables pupils to apply unique design skills to their website.	This is the language of computers and it is essential that pupils understand how basic circuits can transfer and save data. Pupils need to know how instructions and data, stored as binary values, are used in the fetch-execute cycle. They need to understand that different types of data can be represented digitally and know that data is transferred between logic gates within the CPU.	Pupils must be able to think computationally through abstraction and decomposition in order to plan a solution to a problem. Thye should be able to read, write and follow algorithms to support their practical programming skills.	Pupils will understand how block-based code is a visual way of programming text-based code, and be able to implement their prior knowledge of Scratch into the newly introduced Python. They will use this language to code solutions to given problems with increasing complexity.	In a world where there's an app for every possible need, this unit will take pupils from designer to project manager to developer in order to create their own mobile app. Pupils will familiarise themselves with the coding environment before undertaking a project to design, develop and evaluate an app. Learners will work in pairs to consider the needs of the user; decompose the project into smaller, more manageable parts; use the pair	Pupils should understand how devices are connected on a network, including the Internet. They will recognise protocols used and network topologies, being able to describe benefits of and drawbacks.

					programming approach to develop their app together; and finish off by evaluating the success of the project against the needs of the user.	
Why now?	Pupils must understand the importance of syntax when programming. Creating a website using HTML rather than WYSIWYG software provides more opportunities for pupils to debug their errors. Pupils will be coding using a text-based language – Python - in term 3 so this is a suitable project to practice coding and debugging whilst introducing creativity.	Pupils have an understanding of data being input, processed and output by a 'computer'. They are now delving into the mathematical and physical connections within the circuitry.	In order to write text based programs at KS4, students need to be able to decompose a real-world problem and plan an effective solution. This will require an understanding of standard algorithms e.g. sorting and searching algorithms and the ability to write algorithms to solve a given problem using programming techniques.	In order to write text-based programs at KS4, students need to be able to decompose a real-world problem and plan an effective solution. This basic introduction to Python introduces the concepts of computational thinking; sequence, selection and iteration.	In order to write text based programs at KS4, students need to be able to decompose a real-world problem and plan an effective solution. App development software is block based like Scratch which has been used at KS2 but pupils will practice using the main programming constructs.	Pupils who study Computing at KS4 will need to know how internal components of a computer work together. They should be aware of the importance of interconnected devices and the impact on cybersecurity.

Skills & Characteristics	• Use and understand the need for tools specific to user needs and applications • Create digital documents to meet a specification • Skills Builder: ○ *Creativity* when creating a website which is formatted appropriately to meet the specific requirement ○ *Speaking* through class discussion and 'Think,	• Draws and follows logic gates including AND, OR and NOT. • Can complete truth table for logic gates and circuits with up to three inputs. • Carry out binary conversion and arithmetic. • Skills builder ○ *Aiming high* skills when carrying out binary calculations.	• Be able to refine algorithms to reduce the number of instructions required. • Create algorithms for compressing data. • Be able to create abstractions for different purposes • Be able to break down a large Computing problem into its parts • Skills builder ○ *Problem solving* skills when creating algorithms	• Finds where information can be filtered out in generalising problem solutions (abstraction). • Uses logical reasoning to predict outputs, showing an awareness of inputs. Selects similarities and differences in situations and uses these to solve problems (pattern recognition). • Constructs solutions (algorithms) that use repetition and two-way selection. Solves problems through decomposition. • Demonstrates simple algorithms using loops, and selection. Detects and corrects errors i.e. debugging, in algorithms. • Designs, writes and debugs modular	• Finds where information can be filtered out in generalising problem solutions (abstraction). • Uses logical reasoning to predict outputs, showing an awareness of inputs. Selects similarities and differences in situations and uses these to solve problems (pattern recognition). • Constructs solutions (algorithms) that use repetition and two-way selection. Solves problems	• Understands that computer systems consist of hardware and software. • Recognises the different types of network hardware used in a networked computer system • Understands that protocols are required when transferring data in the TCP/IP stack • Understands the need for hardware and software security • Describes how data is transferred

	pair, share' activities. ○ *Listening* during class discussion and 'Think, pair, share' activities. ○ *Aiming High* when producing documents using applications that they have never used before.		to solve an existing problem. ○ *Teamwork* when working together to form suitable solutions to problems.	programs using functions. Selects appropriate variables and relational operators within a loop to govern termination. Establishes the difference between a while loop and a for loop. • Identifies the differences between, and appropriately uses if and if, then and else statements, gaining practical experience of a high-level textual language. • Uses logical reasoning to predict the behaviour of programs. Builds programs that implement algorithms to achieve given goals • Demonstrates how arithmetic operators, if statements, and loops, are used within programs.	through decomposition. • Demonstrates simple algorithms using loops, and selection. Detects and corrects errors i.e. debugging, in algorithms. • Designs, writes and debugs modular programs using functions. Selects appropriate variables and relational operators within a loop to govern termination. Establishes the difference between a while loop and a for loop. • Identifies the differences between, and appropriately	across the Internet • Skills Builder: ○ *Speaking* through class discussion and 'Think, pair, share' activities. ○ *Listening* during class discussion and 'Think, pair, share' activities

				Declares and assigns variables. - Skills builder ○ *Problem solving* skills when forming a programmable solution to a problem. ○ *Aiming high* when programming in a text-based language which pupils have never used before.	uses if and if, then and else statements, gaining practical experience of a high-level textual language. - Uses logical reasoning to predict the behaviour of programs. Builds programs that implement algorithms to achieve given goals - Demonstrates how arithmetic operators, if statements, and loops, are used within programs. Declares and assigns variables. - Skills builder ○ *Problem solving* skills	

					when forming a programmable solution to a problem. ○ *Aiming high* when programming in a text-based language which pupils have never used before.	
Aspirations & Careers	• Pupils will gain an insight into the requirements of ICT/Computing based careers such as coding, network management and data management. • Pupils will be offered the opportunity to enhance their computing skills at 'Computing Club'. • Online coding competitions open to all pupils. • Opportunity to attend 'ICT Live' visit to Disneyland Paris					

Computer Science End Points			
Year Group	**Basic** **(Lower Ability End Points)**	**Clear** **(Middle Ability End Points)**	**Detailed** **(Higher Ability End Points)**
8	• Pupils understand basic HTML statements. • Pupils understand simple Boolean logic and some of its uses in circuits. • Pupils understand how numbers can be represented in binary, and	• Pupils can produce a webpage structure using HTML. • Pupils construct circuits using Boolean expressions. • Pupils understand how numbers can be represented in binary, converting between binary, denary and hexadecimal numbers.	• Pupils use CSS to develop effective websites. • Pupils use logic gates to construct complex circuits and form Boolean expressions. • Pupils understand how numbers can be represented in binary and can carry out

	carry out conversions between binary and denary numbers. • Pupils understand how instructions are stored and executed within a computer system in the form of binary digits • Pupils understand how data of various types (including text, sounds and pictures) are represented digitally. • Pupils recognise a range of sorting and searching algorithms. • Pupils create sequences of instructions and understand the need to be precise when framing and sequencing instructions.	• Pupils understand how data of various types (including text, sounds and pictures) can be represented and manipulated digitally. • Pupils use a range of sorting and searching algorithms, selecting the most appropriate for given scenarios. • Pupils develop, try out and refine sequences of instructions and show efficiency in framing these instructions, using sub-routines where appropriate.	binary addition and conversion operations. • Pupils understand how data of various types (including text, sounds and pictures) can be represented and manipulated digitally and the impact of colour depth on the quality of images. • Pupils use a range of sorting and searching algorithms, justifying the most appropriate for given scenarios. • Pupils develop and refine efficient sequences of instructions, using sub-routines where appropriate.