

The logo for Teach Computing, featuring the words 'Teach' and 'Computing' stacked vertically in white text on a teal background. The text is contained within a white, stylized shape that resembles a folded piece of paper or a document.

Teach

Computing

Teach Computing Progression

Spiral curriculum

The units for key stages 1 and 2 are based on a spiral curriculum. This means that each of the themes is revisited

regularly (at least once in each year group), and pupils revisit each theme through a new unit that consolidates and builds on prior learning within that theme.

This style of curriculum design reduces the amount of knowledge lost through forgetting, as topics are revisited yearly. It also ensures that connections are made even if different teachers are teaching the units within a theme in consecutive years

Coherence and flexibility

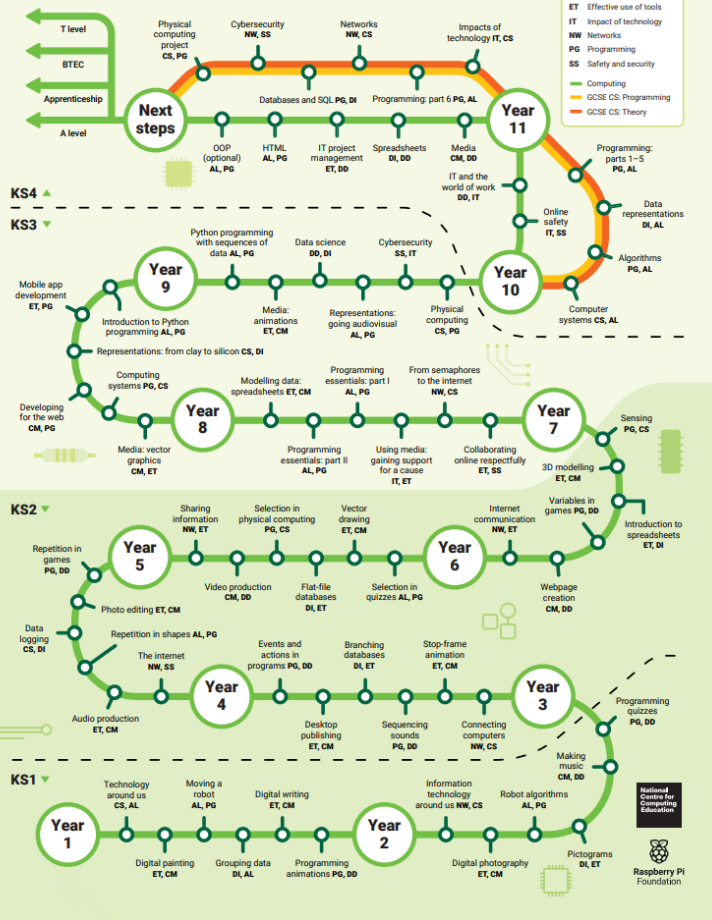
The Teach Computing Curriculum is structured in units.

For these units to be coherent, the lessons within a unit must be taught in order. However, across a year group, the units themselves do not need to be taught in order, with the exception of 'Programming' units, where concepts and skills rely on prior learning and experiences.

Teach Computing Curriculum Journey

Download the curriculum resources: nccpe.io/get_tcc

Key	
AL	Algorithms
CS	Computing systems
CM	Creating media
DI	Data and information
DD	Design and development
ED	Effective use of tools
IT	Impact of technology
NW	Networks
PG	Programming
SS	Safety and security
— Computing	
— GCSE CS: Programming	
— GCSE CS: Theory	



Core principles

Inclusive and ambitious

The Teach Computing Curriculum has been written to support all pupils. Each lesson is sequenced so that it builds on the learning from the previous lesson, and where appropriate, activities are scaffolded so that all pupils can succeed and thrive. Scaffolded activities provide pupils with extra resources, such as visual prompts, to reach the same learning goals as the rest of the class.

Exploratory tasks foster a deeper understanding of a concept, encouraging pupils to apply their learning in different contexts and make connections with other learning experiences. As well as scaffolded activities, embedded within the lessons are a range of pedagogical strategies (defined in the 'Pedagogy' section of this document), which support making computing topics more accessible

Pedagogy

Computing is a broad discipline, and computing teachers require a range of strategies to deliver effective lessons to their pupils. The National Centre for Computing Education's pedagogical approach consists of 12 key principles underpinned by research: each principle has been shown to contribute to effective teaching and learning in computing.

It is recommended that computing teachers use their professional judgement to review, select, and apply relevant strategies for their pupils.

These 12 principles are embodied by the Teach Computing Curriculum, and examples of their application can be found throughout the units of work at every key stage. Beyond delivering these units, you can learn more about these principles and related strategies in the National Centre for Computing Education pedagogy toolkit (nccce.io/pedagogy).



Lead with concepts

Support pupils in the acquisition of knowledge, through the use of key concepts, terms, and vocabulary, providing opportunities to build a shared and consistent understanding. Glossaries, concept maps, and displays, along with regular recall and revision, can support this approach.



Work together

Encourage collaboration, specifically using pair programming and peer instruction, and also structured group tasks. Working together stimulates classroom dialogue, articulation of concepts, and development of shared understanding.



Get hands-on

Use physical computing and making activities that offer tactile and sensory experiences to enhance learning. Combining electronics and programming with arts and crafts (especially through exploratory projects) provides pupils with a creative, engaging context to explore and apply computing concepts.



Unplug, unpack, repack

Teach new concepts by first unpacking complex terms and ideas, exploring these ideas in unplugged and familiar contexts, then repacking this new understanding into the original concept. This approach, called 'semantic waves', can help pupils develop a secure understanding of complex concepts.



Model everything

Model processes or practices – everything from debugging code to binary number conversions – using techniques such as worked examples and live coding. Modelling is particularly beneficial to novices, providing scaffolding that can be gradually taken away.



Foster program comprehension

Use a variety of activities to consolidate knowledge and understanding of the function and structure of programs, including debugging, tracing, and Parson's Problems. Regular comprehension activities will help secure understanding and build connections with new knowledge.



Create projects

Use project-based learning activities to provide pupils with the opportunity to apply and consolidate their knowledge and understanding. Design is an important, often overlooked aspect of computing. Pupils can consider how to develop an artefact for a particular user or function, and evaluate it against a set of criteria.



Add variety

Provide activities with different levels of direction, scaffolding, and support that promote learning, ranging from highly structured to more exploratory tasks. Adapting your instruction to suit different objectives will help keep all pupils engaged and encourage greater independence.



Challenge misconceptions

Use formative questioning to uncover misconceptions and adapt teaching to address them as they occur. Awareness of common misconceptions alongside discussion, concept mapping, peer instruction, or simple quizzes can help identify areas of confusion.



Make concrete

Bring abstract concepts to life with real-world, contextual examples, and a focus on interdependencies with other curriculum subjects. This can be achieved through the use of unplugged activities, proposing analogies, storytelling around concepts, and finding examples of the concepts in pupils' lives.



Structure lessons

Use supportive frameworks when planning lessons, such as PRIMM (Predict, Run, Investigate, Modify, Make) and (Use-Modify-Create). These frameworks are based on research and ensure that differentiation can be built in at various stages of the lesson.



Read and explore code first

When teaching programming, focus first on code 'reading' activities, before code writing. With both block-based and text-based programming, encourage pupils to review and interpret blocks of code. Research has shown that being able to read, trace, and explain code augments pupils' ability to write code.



How we teach computing

12 pedagogy principles

Lead with concepts



Support pupils in the acquisition of knowledge, through the use of key concepts, terms, and vocabulary, providing opportunities to build a shared and consistent understanding. Glossaries, concept maps, and displays, along with regular recall and revision, can support this approach.

Unplug, unpack, repack

Teach new concepts by first unpacking complex terms and ideas, exploring these ideas in unplugged and familiar contexts, then repacking this new understanding into the original concept. This approach (semantic waves) can help pupils develop a secure understanding of complex concepts.

Create projects



Use project-based learning activities to provide pupils with the opportunity to apply and consolidate their knowledge and understanding. Design is an important, often overlooked aspect of computing. Pupils can consider how to develop an artefact for a particular user or function, and evaluate it against a set of criteria.

Challenge misconceptions



Use formative questioning to uncover misconceptions and adapt teaching to address them as they occur. Awareness of common misconceptions alongside discussion, concept mapping, peer instruction, or simple quizzes can help identify areas of confusion.

Structure lessons

Use supportive frameworks when planning lessons, such as PRIMM (Predict, Run, Investigate, Modify, Make) and Use-Modify-Create. These frameworks are based on research and ensure that differentiation can be built in at various stages of the lesson.

Work together



Encourage collaboration, specifically using pair programming and peer instruction, and also structured group tasks. Working together stimulates classroom dialogue, articulation of concepts, and development of shared understanding.

Model everything

Model processes or practices – everything from debugging code to binary number conversions – using techniques such as worked examples and live coding. Modelling is particularly beneficial to novices, providing scaffolding that can be gradually taken away.

Add variety

Provide activities with different levels of direction, scaffolding, and support that promote active learning, ranging from highly structured to more exploratory tasks. Adapting your instruction to suit different objectives will help keep all pupils engaged and encourage greater independence.

Make concrete

Bring abstract concepts to life with real-world, contextual examples and a focus on interdependencies with other curriculum subjects. This can be achieved through the use of unplugged activities, proposing analogies, storytelling around concepts, and finding examples of the concepts in pupils' lives.

Read and explore code first

When teaching programming, focus first on code 'reading' activities, before code writing. With both block-based and text-based programming, encourage pupils to review and interpret blocks of code. Research has shown that being able to read, trace, and explain code augments pupils' ability to write code.

Get hands-on



Use physical computing and making activities that offer tactile and sensory experiences to enhance learning. Combining electronics and programming with arts and crafts (especially through exploratory projects) provides pupils with a creative, engaging context to explore and apply computing concepts.

Foster program comprehension



Use a variety of activities to consolidate knowledge and understanding of the function and structure of programs, including debugging, tracing, and Parson's Problems. Regular comprehension activities will help secure understanding and build connections with new knowledge.

Find out more about
our principles and
add some or all
to your personal
pedagogy toolkit.

nccpe.io/pedagogy

Progression across key stages

All learning objectives have been mapped to the National Centre for Computing Education's taxonomy of ten strands, which ensures that units build on each other from one key stage to the next.

Progression across year groups

Within the Teach Computing Curriculum, every year group learns through units within the same four themes, which combine the ten strands of the National Centre for Computing Education's taxonomy (see table, right).

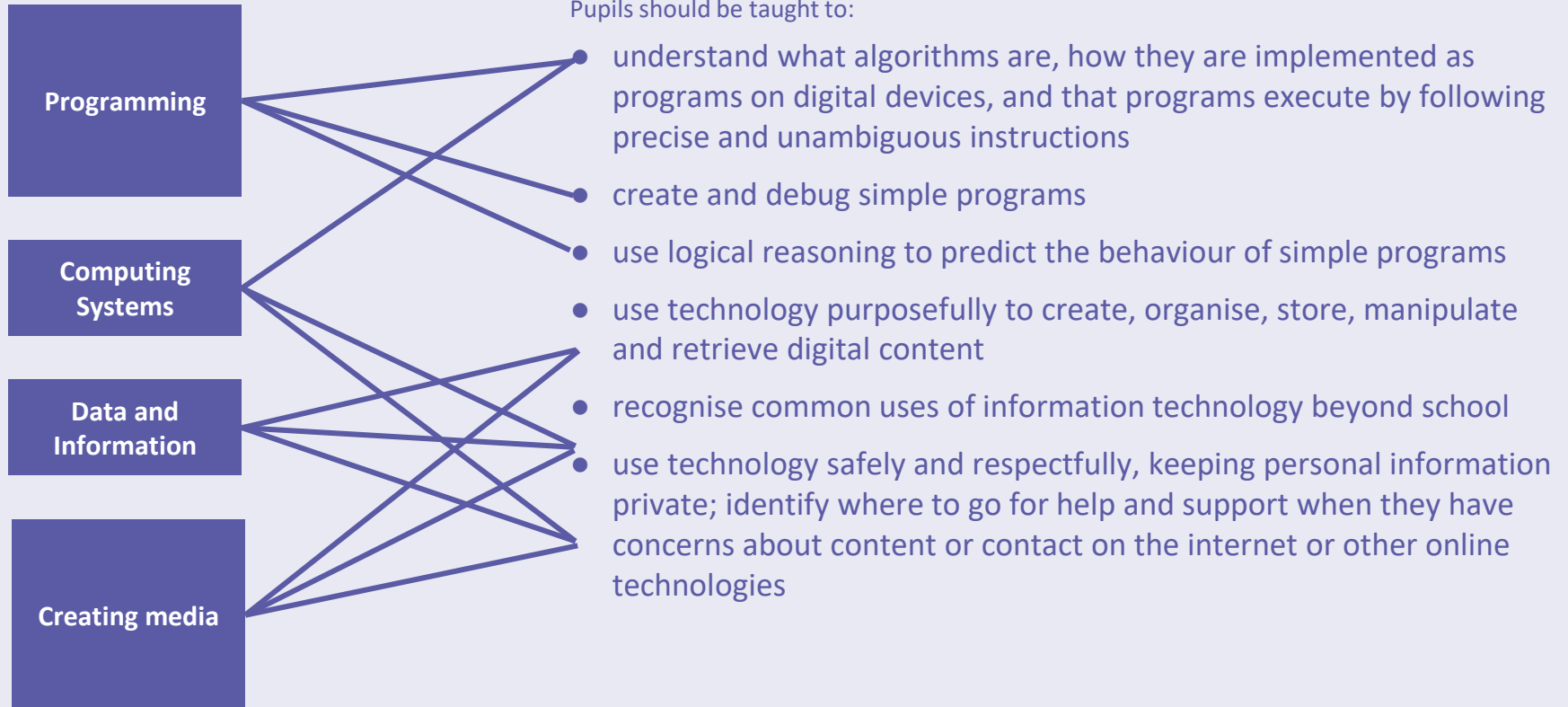
This approach allows us to use the spiral curriculum approach (see the 'Spiral curriculum' section for more information) to progress skills and concepts from one year group to the next.

Primary themes	Computing systems and networks	Programming	Data and information	Creating media
Taxonomy strands	Computer systems	Programming	Data and information	Creating media
	Computer networks	Algorithms		Design and development
		Design and development		
			Effective use of tools	
			Impact of technology	
				Safety and security

Teach Computing Taxonomy

Abbreviation	Strand	Description
NW	Networks	Understand how networks can be used to retrieve and share information and come with associated risks
CM	Creating Media	Select and create a range of media including text, images, sounds and video.
DI	Data & Information	How is data stored, organised and used to represent real world artefacts and scenarios
DD	Design & Deveopment	The activities involved in planning, creating and evaluating computing artefacts
CS	Computing Systems	What is a computer, how do it's constituent parts function together as a whole
IT	Impact of Technology	How individuals, systems and society as a whole interact with computer systems
AL	Algorithms	Being able to comprehend, design, create and evaluate algorithms
PG	Programming	Creating software to allow computers to solve problems
ET	Effective Use of tools	Use software tools to support computing work
SS	Safety & Security	Understanding risks when using technology and how to protect individuals and systems

National Curriculum KS1



National Curriculum KS2



Teach Computing Curriculum overview

Brief overview

	Computing systems and networks ¹	Creating media	Programming A	Data and information	Creating media	Programming B
Year 1	Technology around us (1.1)*	Digital painting (1.2)	Moving a robot (1.3)	Grouping data (1.4)	Digital writing (1.5)	Programming animations (1.6)
Year 2	Information technology around us (2.1)	Digital photography (2.2)	Robot algorithms (2.3)	Pictograms (2.4)	Digital music (2.5)	Programming quizzes (2.6)

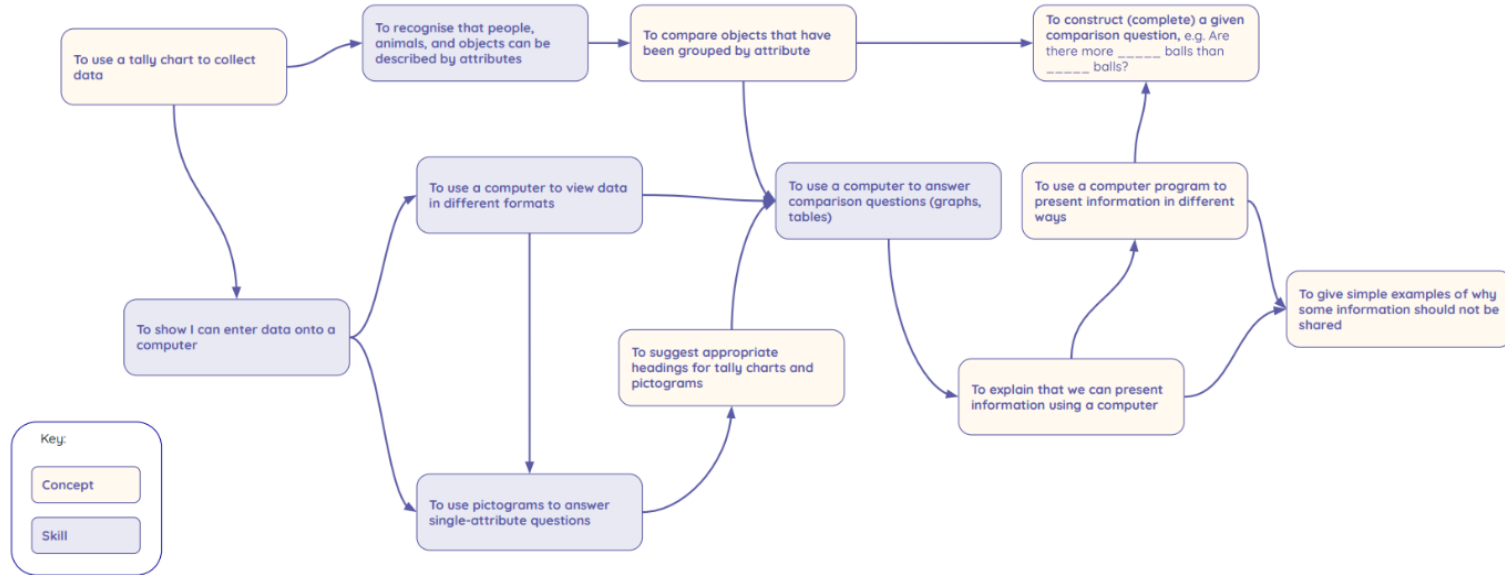
Unit summaries

	Computing systems and networks	Creating media	Programming A	Data and information	Creating media	Programming B
Year 1	Technology around us Recognising technology in school and using it responsibly.	Digital painting Choosing appropriate tools in a program to create art, and making comparisons with working non-digitally.	Moving a robot Writing short algorithms and programs for floor robots, and predicting program outcomes.	Grouping data Exploring object labels, then using them to sort and group objects by properties.	Digital writing Using a computer to create and format text, before comparing to writing non-digitally.	Programming animations Designing and programming the movement of a character on screen to tell stories.
Year 2	Information technology around us Identifying IT and how its responsible use improves our world in school and beyond.	Digital photography Capturing and changing digital photographs for different purposes.	Robot algorithms Creating and debugging programs, and using logical reasoning to make predictions.	Pictograms Collecting data in tally charts and using attributes to organise and present data on a computer.	Digital music Using a computer as a tool to explore rhythms and melodies, before creating a musical composition.	Programming quizzes Designing algorithms and programs that use events to trigger sequences of code to make an interactive quiz.

National Curriculum Coverage – Years 1 and 2

	1.1 Technology around us	1.2 Digital painting	1.3 Moving a robot	1.4 Grouping data	1.5 Digital writing	1.6 Programming animations	2.1 Information technology around us	2.2 Digital photography	2.3 Robot algorithms	2.4 Pictograms	2.5 Digital music	2.6 Programming quizzes
Understand what algorithms are, how they are implemented as programs on digital devices, and that programs execute by following precise and unambiguous instructions			✓			✓			✓			✓
Create and debug simple programs			✓			✓			✓			✓
Use logical reasoning to predict the behaviour of simple programs			✓			✓			✓			✓
Use technology purposefully to create, organise, store, manipulate, and retrieve digital content	✓	✓		✓	✓		✓	✓		✓	✓	✓
Recognise common uses of information technology beyond school	✓		✓				✓	✓				
Use technology safely and respectfully, keeping personal information private; identify where to go for help and support when they have concerns about content or contact on the internet or other online technologies	✓			✓	✓		✓	✓	✓	✓		

KS1 Example learning graph Year 2 - Data and Information - Pictograms



Software and hardware overview

Requirements for pupils – below

✓ Used for the unit – reflected in screenshots ● Could be used as an alternative

	Desktop or laptop	Chromebook	Tablet	Software or hardware
1.1 Technology around us	✓	✓	●	paintz.app
1.2 Digital painting	✓	✓	●	Microsoft Paint or similar
1.3 Moving a robot				Bee-Bot, Blue-Bot, or other fixed-movement floor robot
1.4 Grouping data	✓	✓		Google Slides or Microsoft PowerPoint
1.5 Digital writing	✓	✓	●	Google Docs or Microsoft Word
1.6 Programming animations	●	●	✓	ScratchJr
2.1 Information technology around us	✓	✓		Google Slides or Microsoft PowerPoint
2.2 Digital photography	✓		●	Digital camera
2.3 Robot algorithms				Bee-Bot, Blue-Bot, or other fixed-movement floor robot
2.4 Pictograms	✓	✓	●	j2data Pictogram
2.5 Digital music	✓	✓	●	Chrome Music Lab
2.6 Programming quizzes	●	●	✓	ScratchJr

Teach Computing Curriculum overview

	Computing systems and networks	Creating media	Programming A	Data and information	Creating media	Programming B
Year 3	Connecting computers (3.1)	Stop-frame animation (3.2)	Sequencing sounds (3.3)	Branching databases (3.4)	Desktop publishing (3.5)	Events and actions in programs (3.6)
Year 4	The internet (4.1)	Audio production (4.2)	Repetition in shapes (4.3)	Data logging (4.4)	Photo editing (4.5)	Repetition in games (4.6)
Year 5	Systems and searching (5.1)	Video production (5.2)	Selection in physical computing (5.3)	Flat-file databases (5.4)	Introduction to vector graphics (5.5)	Selection in quizzes (5.6)
Year 6	Communication and collaboration (6.1)	Webpage creation (6.2)	Variables in games (6.3)	Introduction to spreadsheets (6.4)	3D modelling (6.5)	Sensing movement (6.6)

Unit summaries

	Computing systems and networks	Creating media	Programming A	Data and information	Creating media	Programming B
Year 3	<p>Connecting computers</p> <p>Identifying that digital devices have inputs, processes, and outputs, and how devices can be connected to make networks.</p>	<p>Stop-frame animation</p> <p>Capturing and editing digital still images to produce a stop-frame animation that tells a story.</p>	<p>Sequencing sounds</p> <p>Creating sequences in a block-based programming language to make music.</p>	<p>Branching databases</p> <p>Building and using branching databases to group objects using yes/no questions.</p>	<p>Desktop publishing</p> <p>Creating documents by modifying text, images, and page layouts for a specified purpose.</p>	<p>Events and actions in programs</p> <p>Writing algorithms and programs that use a range of events to trigger sequences of actions.</p>
Year 4	<p>The internet</p> <p>Recognising the internet as a network of networks including the WWW, and why we should evaluate online content.</p>	<p>Audio production</p> <p>Capturing and editing audio to produce a podcast, ensuring that copyright is considered.</p>	<p>Repetition in shapes</p> <p>Using a text-based programming language to explore count-controlled loops when drawing shapes.</p>	<p>Data logging</p> <p>Recognising how and why data is collected over time, before using data loggers to carry out an investigation.</p>	<p>Photo editing</p> <p>Manipulating digital images, and reflecting on the impact of changes and whether the required purpose is fulfilled.</p>	<p>Repetition in games</p> <p>Using a block-based programming language to explore count-controlled and infinite loops when creating a game.</p>

Unit summaries

	Computing systems and networks	Creating media	Programming A	Data and information	Creating media	Programming B
Year 5	Systems and searching Recognising IT systems in the world and how some can enable searching on the internet.	Video production Planning, capturing, and editing video to produce a short film.	Selection in physical computing Exploring conditions and selection using a programmable microcontroller.	Flat-file databases Using a database to order data and create charts to answer questions.	Introduction to vector graphics Creating images in a drawing program by using layers and groups of objects.	Selection in quizzes Exploring selection in programming to design and code an interactive quiz.
Year 6	Communication and collaboration Exploring how data is transferred by working collaboratively online.	Webpage creation Designing and creating webpages, giving consideration to copyright, aesthetics, and navigation.	Variables in games Exploring variables when designing and coding a game.	Introduction to spreadsheets Answering questions by using spreadsheets to organise and calculate data.	3D modelling Planning, developing, and evaluating 3D computer models of physical objects.	Sensing movement Designing and coding a project that captures inputs from a physical device.

National curriculum coverage - Years 3 and 4

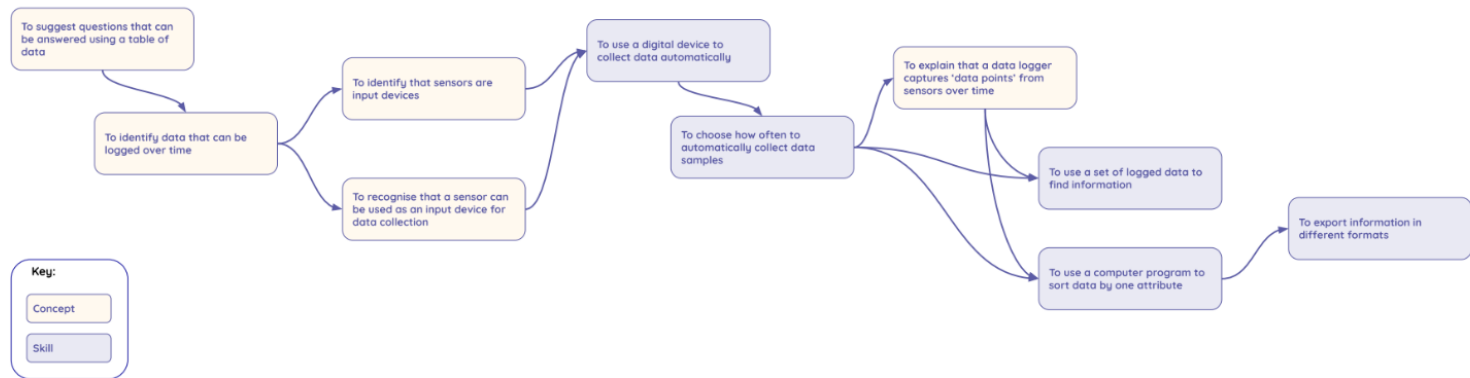
	3.1 Connecting computers	3.2 Stop-frame animation	3.3 Sequencing sounds	3.4 Branching databases	3.5 Desktop publishing	3.6 Events and actions in programs	4.1 The internet	4.2 Audio production	4.3 Repetition in shapes	4.4 Data logging	4.5 Photo editing	4.6 Repetition in games
Design, write and debug programs that accomplish specific goals, including controlling or simulating physical systems; solve problems by decomposing them into smaller parts			✓			✓			✓			✓
Use sequence, selection, and repetition in programs; work with variables and various forms of input and output	✓		✓			✓			✓	✓		✓
Use logical reasoning to explain how some simple algorithms work and to detect and correct errors in algorithms and programs			✓			✓			✓			✓
Understand computer networks, including the internet; how they can provide multiple services, such as the World Wide Web, and the opportunities they offer for communication and collaboration	✓						✓					
Use search technologies effectively, appreciate how results are selected and ranked, and be discerning in evaluating digital content					✓		✓	✓			✓	
Select, use and combine a variety of software (including internet services) on a range of digital devices to design and create a range of programs, systems and content that accomplish given goals, including collecting, analysing, evaluating and presenting data and information	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Use technology safely, respectfully and responsibly; recognise acceptable/unacceptable behaviour; identify a range of ways to report concerns about content and contact		✓		✓			✓	✓			✓	

National curriculum coverage - Years 5 and 6

	5.1 Systems and searching	5.2 Video production	5.3 Selection in physical computing	5.4 Flat-file databases	5.5 Introduction to vector graphics	5.6 Selection in quizzes	6.1 Communication and collaboration	6.2 Webpage creation	6.3 Variables in games	6.4 Introduction to spreadsheets	6.5 3D modelling	6.6 Sensing movement
Design, write and debug programs that accomplish specific goals, including controlling or simulating physical systems; solve problems by decomposing them into smaller parts			✓			✓	✓		✓			✓
Use sequence, selection, and repetition in programs; work with variables and various forms of input and output			✓			✓			✓			✓
Use logical reasoning to explain how some simple algorithms work and to detect and correct errors in algorithms and programs			✓			✓			✓			✓
Understand computer networks, including the internet; how they can provide multiple services, such as the World Wide Web, and the opportunities they offer for communication and collaboration	✓						✓					
Use search technologies effectively, appreciate how results are selected and ranked, and be discerning in evaluating digital content		✓		✓				✓				
Select, use and combine a variety of software (including internet services) on a range of digital devices to design and create a range of programs, systems and content that accomplish given goals, including collecting, analysing, evaluating and presenting data and information	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Use technology safely, respectfully and responsibly; recognise acceptable/unacceptable behaviour; identify a range of ways to report concerns about content and contact	✓	✓						✓	✓		✓	

KS2 Example learning graph

Year 4 - Data and Information - Data logging



Software and hardware overview

Requirements for pupils – below

	Desktop or laptop	Chromebook	Tablet	Software or hardware
3.1 Connecting computers	✓	●	●	Painting program (any)
3.2 Stop-frame animation	●	●	✓	iMotion (app for iOS)
3.3 Sequencing sounds	✓	✓	●	Scratch
3.4 Branching databases	✓	✓	●	j2data Branch and Pictogram
3.5 Desktop publishing	✓	●	●	Canva.com
3.6 Events and actions in programs	✓	✓	●	Scratch
4.1 The internet	✓	✓	✓	Various websites
4.2 Audio production	✓			Audacity
4.3 Repetition in shapes	✓	●	●	FMSLogo
4.4 Data logging	✓	+	+	Data logger and associated software
4.5 Photo editing	✓	●		Paint.NET (for Microsoft Windows)
4.6 Repetition in games	✓	✓	●	Scratch

✓ Used for the unit – reflected in screenshots ● Could be used as an alternative + Data loggers that work with Chromebooks or tablets are available. Check with suppliers.

Software and hardware overview, cont.

Requirements for pupils – below

	Desktop or laptop	Chromebook	Tablet	Software or hardware
5.1 Systems and Searching	✓	✓		Google Slides
5.2 Video production	✓	●	●	Microsoft Photos (for Microsoft Windows 10)
5.3 Selection in physical computing	✓	✓		Crumble controller + starter kit + motor
5.4 Flat-file databases	✓	✓	●	j2data Database
5.5 Introduction to vector graphics	✓	●		Google Drawings
5.6 Selection in quizzes	✓	✓		Scratch
6.1 Communication and collaboration	✓	✓		Google Slides
6.2 Webpage creation	✓	✓		Google Sites
6.3 Variables in games	✓	✓		Scratch
6.4 Introduction to spreadsheets	✓	✓	●	Google Sheets or Microsoft Excel
6.5 3D modelling	✓	✓	●	Tinkercad
6.6 Sensing movement	✓	✓	●	micro:bit and Microsoft MakeCode

✓ Used for the unit – reflected in screenshots ● Could be used as an alternative