Contents

Introduction

The Teach Computing Curriculum (ncce.io/tcc) is a comprehensive collection of materials produced to support 500 hours of teaching, facilitating the delivery of the entire English computing curriculum from key stage 1 to 4 (5- to 16-year-olds). The Teach Computing Curriculum was created by the Raspberry Pi Foundation on behalf of the National Centre for Computing Education (NCCE). All content is free, and editable under the Open Government Licence (OGL — ncce.io/ogl), ensuring that the resources can be tailored to each individual teacher and school setting. The materials are suitable for all pupils irrespective of their skills, background, and additional needs.

The aims of the Teach Computing Curriculum are as follows:

- Reduce teacher workload
- Show the breadth and depth of the computing curriculum, particularly beyond programming!
- Demonstrate how computing can be taught well, based on research
- Highlight areas for subject knowledge and pedagogy enhancement through training

The Teach Computing Curriculum resources are regularly updated in response to feedback. Feedback can be submitted at ncce.io/rrfeedback or by email to resourcesfeedback@raspberrypi.org.

Curriculum design

The approach

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.

Knowledge organisation

The Teach Computing Curriculum uses the National Centre for Computing Education's computing taxonomy to ensure comprehensive coverage of the subject. This has been developed through a thorough review of the KS1–4 computing programme of study, and the GCSE and A level computer science specifications across all awarding bodies. All learning outcomes can be described through a high-level taxonomy of ten strands, ordered alphabetically as follows:

- **Algorithms** — Be able to comprehend, design, create, and evaluate algorithms
- **Computer networks** — Understand how networks can be used to retrieve and share information, and how they come with associated risks
- **Computer systems** — Understand what a computer is, and how its constituent parts function together as a whole
- **Creating media** — Select and create a range of media including text, images, sounds, and video
- **Data and information** — Understand how data is stored, organised, and used to represent real-world artefacts and scenarios
- **Design and development** — Understand the activities involved in planning, creating, and evaluating computing artefacts
- **Effective use of tools** — Use software tools to support computing work
- **Impact of technology** — Understand how individuals, systems, and society as a whole interact with computer systems
- **Programming** — Create software to allow computers to solve problems
- **Safety and security** — Understand risks when using technology, and how to protect individuals and systems

The taxonomy provides categories and an organised view of content to encapsulate the discipline of computing. Whilst all strands are present at all phases, they are not always taught explicitly.

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.

Physical computing

The Teach Computing Curriculum acknowledges that physical computing plays an important role in modern pedagogical approaches in computing, both as a tool to engage pupils and as a strategy to develop pupils' understanding in more creative ways. Additionally, physical computing supports and engages a diverse range of pupils in tangible and challenging tasks.

The physical computing units in the Teach Computing Curriculum are:

- Year 5 – Selection in physical computing, which uses a Crumble controller
- Year 6 – Sensing movement, which uses a micro:bit

Your local Computing Hub can loan you the kit you need to teach the physical computing units from our curriculum (ncce.io/hubs).

Online safety

The unit overviews for each unit show the links between the content of the lessons and the national curriculum and Education for a Connected World framework (ncce.io/efacw). These references have been provided to show where aspects relating to online safety, or digital citizenship, are covered within the Teach Computing Curriculum. Not all of the objectives in the Education for a Connected World framework are covered in the Teach Computing Curriculum, as some are better suited to personal, social, health, and economic (PSHE) education; spiritual, moral, social, and cultural (SMSC) development; and citizenship. However, the coverage required for the computing national curriculum is provided.

Schools should decide for themselves how they will ensure that online safety is being managed effectively in their setting, as the scope of this is much wider than just curriculum content.

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.

Research-informed

The subject of computing is much younger than many other subjects, and as such, there is still a lot more to learn about how to teach it effectively. To ensure that teachers are as prepared as possible, the Teach Computing Curriculum builds on a set of pedagogical principles (see the 'Pedagogy' section of this document), which are underpinned by the latest computing research, to demonstrate effective pedagogical strategies throughout. To remain up-to-date as research continues to develop, every aspect of the Teach Computing Curriculum is reviewed each year and changes are made as necessary.

Time-saving for teachers

The Teach Computing Curriculum has been designed to reduce teacher workload. To ensure this, the Teach Computing Curriculum includes all the resources a teacher needs, covering every aspect from planning, to progression mapping, to supporting materials.

Structure of the units of work

Every unit of work in the Teach Computing Curriculum contains: a unit overview; a learning graph, to show the progression of skills and concepts in a unit; lesson content — including a detailed lesson plan, slides for learners, and all the resources you will need; and formative and summative assessment opportunities.

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)

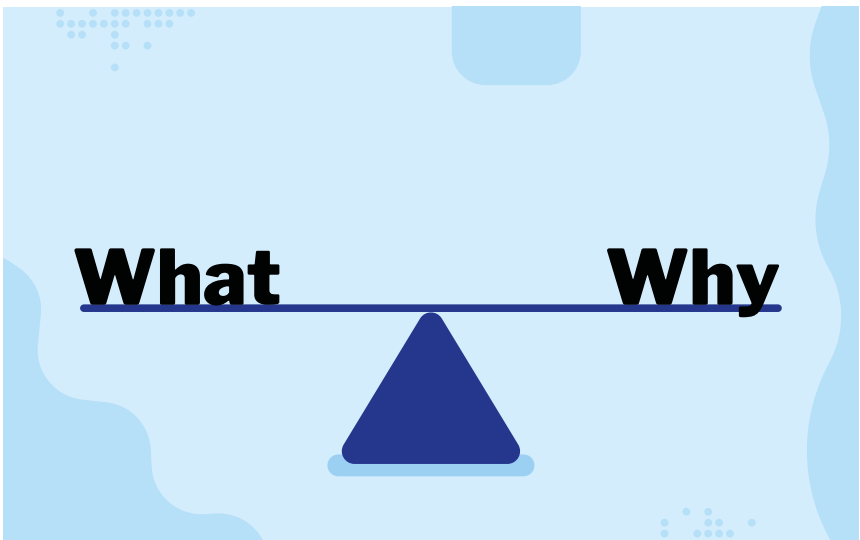
[1]Networks are not part of the key stage 1 national curriculum for computing but the title is used as a strand across primary.

*The numbers in the brackets are a 'quick code' reference for each unit, e.g. 1.3 refers to the third Year 1 unit in the recommended teaching order.

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	✓			✓	✓		✓	✓	✓	✓		



Teaching order

The order in which to teach units within a school year is not prescribed, other than for the two 'Programming' units for each year group, which build on each other. It is recommended that the 'Programming' and 'Creating media' units be revisited in two different terms within the school year, so that the concepts and skills can be revisited and consolidated. Otherwise, schools can choose the order in which they teach the units, based on the needs of their pupils and other topics or events that are happening throughout the school year, to make use of cross-curricular links wherever possible.

Mixed year groups

The Teach Computing Curriculum is based on a learning progression from Year 1 through to Year 6 that fits into an overall progression including secondary school. In order to use this progression with mixed year groups, it is advisable for teachers to break up the content as they see fit, based on the learning graphs for the year groups that they are teaching.

Progression

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).

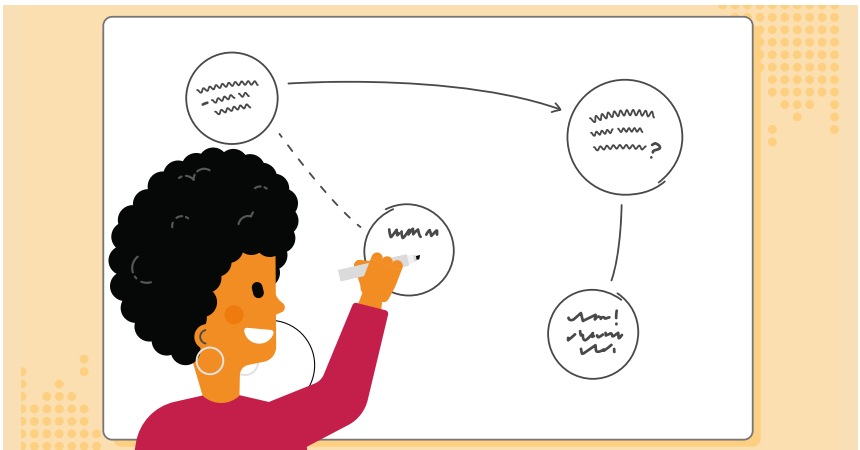
This approach allows us to use the spiral curriculum approach (see the 'Spiral curriculum' section for more information) to progress skills and concepts from one year group to the next.

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

Progression within a unit — learning graphs

Learning graphs are provided as part of each unit and demonstrate progression through concepts and skills. In order to learn some of those concepts and skills, pupils need prior knowledge of others, so the learning graphs show which concepts and skills need to be taught first and which could be taught at a different time.

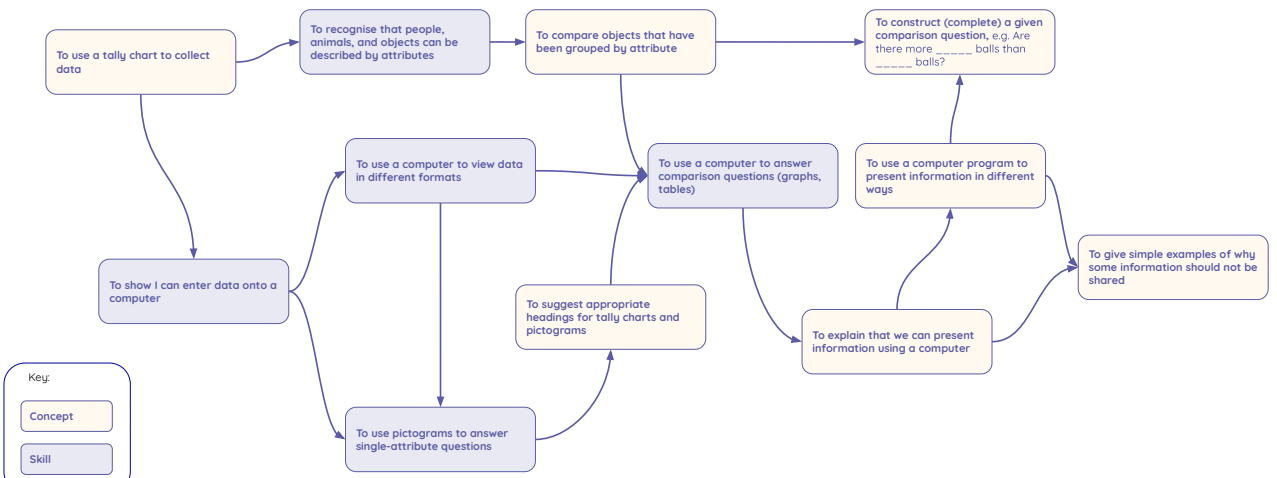
The learning graphs often show more statements than there are learning objectives. All of the skills and concepts learnt are included in the learning graphs. Some of these skills and concepts are milestones, which form learning objectives, while others are smaller steps towards these milestones, which form success criteria. Please note that the wording of the statements may be different in the learning graphs than in the lessons, as the learning graphs are designed for teachers, whereas the learning objectives and success criteria are age-appropriate so that they can be understood by pupils.



In each year group, there are two 'Programming' units of work, but only one 'Programming' learning graph. The second 'Programming' unit builds on the content that was taught in the first 'Programming' unit so closely that there is no specific divide where one ends and the other begins.

KS1 Example learning graph

Year 2 - Data and Information - Pictograms



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 (ncce.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.

Assessment

Formative assessment

Every lesson includes formative assessment opportunities for teachers to use. These opportunities are listed in the lesson plan and are included to ensure that misconceptions are recognised and addressed if they occur. They vary from teacher observation or questioning, to marked activities.

These assessments are vital to ensure that teachers are adapting their teaching to suit the needs of the pupils that they are working with, and you are encouraged to change parts of the lesson, such as how much time you spend on a specific activity, in response to these assessments.

The learning objective and success criteria are introduced in the slides at the beginning of every lesson. At the end of every lesson, pupils are invited to assess how well they feel they have met the learning objective using thumbs up, thumbs sideways, or thumbs down. This gives pupils a reminder of the content that has been covered, as well as a chance to reflect. It is also a chance for teachers to see how confident the class is feeling so that they can make changes to subsequent lessons accordingly.

Summative assessment

Pedagogically, when we assess, we want to ensure that we are assessing a pupil's understanding of computing concepts and skills, as opposed to their reading and writing skills. Therefore, we encourage observational assessment while pupils are still developing their literacy skills. We believe that this is the most reliable way to capture an accurate picture of learning.

Observing learning

To capture summative assessment data of KS1 pupils, we recommend using the success criteria in each lesson and capturing some of the following while the lesson is taking place:

- The work that pupils complete (marking)
- Notes on conversations or discussions that you have or hear during an activity
- Photographs of the work that pupils produce during an activity
- The pupils' self-assessments at the end of the lesson

This data is to support teachers' assessments of the pupils' understanding of the concepts and skills that were taught in the lesson. To help you make these assessments, you could also use one, or a combination of, the following strategies:

- Focussing on different pupils each lesson
- Creating checklists of what you expect to see
- Focussing on specific pupils

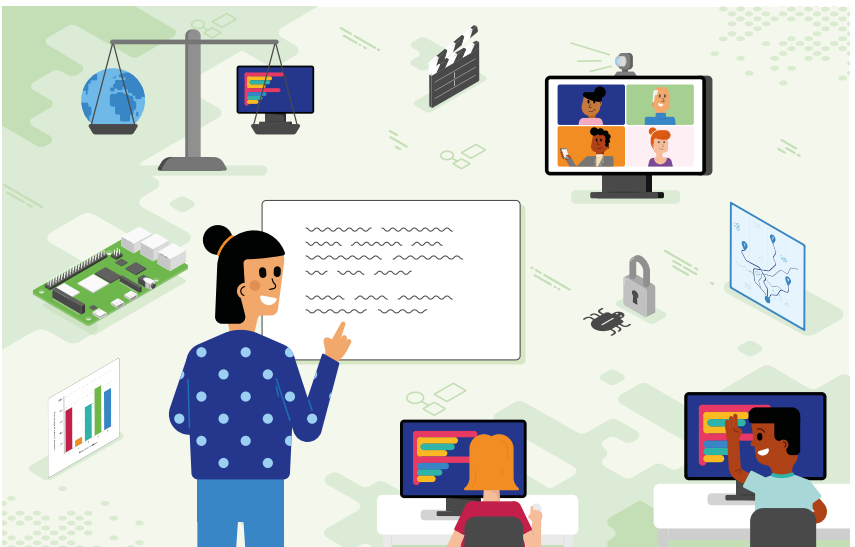
End of the unit

A pupil working at age-related expectations should be able to meet the success criteria for each lesson by the end of the unit. However, it should also be noted that some pupils may take longer to grasp certain skills and concepts and therefore may achieve a success criterion from a lesson at a later date.

At the end of a unit, you may wish to use the observations that you have made across each of the lessons to determine an overall snapshot of a pupil's understanding of the content from that unit.

Adapting for your setting

As there are no nationally agreed levels of assessment, the assessment materials provided are designed to be used and adapted by schools in a way that best suits their needs. The summative assessment materials will inform teacher judgements around what a pupil has understood in each computing unit, and could feed into a school's assessment process, to align with their approach to assessment in other foundation subjects.



Resources

Software and hardware

Computing is intrinsically linked to technology and therefore requires that pupils experience and use a range of digital tools and devices. As the Teach Computing Curriculum was being written, careful consideration was given to the hardware and software selected for the units. The primary consideration was how we felt a tool would best allow pupils to meet learning objectives; the learning always came first and the tool second.

To make the units of work more accessible to pupils and teachers, the materials include screenshots, videos, and instructions, and these are based on the tools listed in the table below. The list below should not be seen as an explicit requirement for schools. Schools may choose to use alternative tools that offer the same features as described in the units. All of the learning objectives can be met with alternative hardware and software, as the learning objectives are not designed to be tool-specific.

Software

If you do not wish to use the software recommended in the units, you could use an alternative piece of software that provides the same function. All learning objectives should be achievable using alternative software, however, there will be a lot less support for teachers, as screenshots and demonstration videos reflect the software referenced in the materials.

The units of work include the use of free software that would need to be installed on local machines, and software that is available as an online tool. Where software needs to be installed locally, schools will need to plan software installation in advance.

Several of the units that use online tools require schools to sign up to free services in order to access the tools. This also allows pupils the opportunity to save the projects that they are working on, and gives them the skills that they need to manage their own usernames and passwords as digital citizens. However, the school needs to ensure that they are comfortable using the software, and that it is in line with their policies about using online tools and how teachers will manage accounts.

Hardware

Pupils should experience a range of digital devices, which may include desktop, laptop, and tablet computers. Pupils should also experience hardware designed for specific purposes, e.g. data loggers, floor robots, and microcontrollers.

Several of the Teach Computing Curriculum units require the use of physical computing devices. This is in recognition of the growing importance of physical computing and digital making and was part of our curriculum design from the beginning. As we are aware that not all schools will have invested in this equipment, NCCE Computing Hubs (ncce.io/hubs) have a number of class sets of equipment, which will be loaned to schools in rotation, with some set aside for CPD sessions.

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

National Centre for Computing Education

The National Centre for Computing Education (NCCE) is funded by the Department for Education and marks a significant investment in improving the provision of computing education in England.

The NCCE is run by a consortium made up of STEM Learning, the Raspberry Pi Foundation, and BCS, The Chartered Institute for IT. Our vision is to achieve a world-leading computing education for every child in England.

The NCCE provides high-quality support for the teaching of computing in schools and colleges, from key stage 1 through to A level. Our extensive range of training, resources, and support covers elements of the curriculum at every key stage, catering for all levels of subject knowledge and experience.

For further information, visit: teachcomputing.org

Contributions: We would like to thank the many people who helped to create the Teach Computing Curriculum: our content writers, advisors, reviewers, pilot schools, and every teacher who has taken the time to send us feedback.