



# COMPUTING CURRICULUM

*'Everybody should learn how to program a computer, because it teaches you how to think.'*

*Steve Jobs*

# Computing: Intent

The Teach Computing Curriculum ([nccce.io/tcc](https://nccce.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 — [nccce.io/ogl](https://nccce.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:

- 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
- Provide a comprehensive set of resources which, in turn, helps to reduce teacher workload

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 which support making computing topics more accessible. 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 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. 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.



# Implementation:

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. 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. 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 experience

# When will this be taught?

## 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)	Web page creation (6.2)	Variables in games (6.3)	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><b>Connecting computers</b> Identifying that digital devices have inputs, processes, and outputs, and how devices can be connected to make networks</p>	<p><b>Stop-frame animation</b> Capturing and editing digital still images to produce a stop frame animation that tells a story</p>	<p><b>Sequencing sounds</b> Creating sequences in a block-based programming language to make music.</p>	<p><b>Branching databases</b> Building and using branching databases to group objects using yes/no questions.</p>	<p><b>Desktop publishing</b> Creating documents and modifying text, images and page layouts for a specific purpose.</p>	<p><b>Events and actions in programs</b> Writing algorithms and programs that use a range of events to trigger sequences of actions.</p>
Year 4	<p><b>The internet</b> Recognising that the internet is a network of networks including the WWW, and why we should evaluate online content.</p>	<p><b>Audio production</b> Capturing and editing audio to produce a podcast, ensuring that copyright is considered.</p>	<p><b>Repetition in shapes</b> Using a text-based programming language to explore count-controlled loops when drawing shapes.</p>	<p><b>Data logging</b> Recognising how and why data is collected over time, before using data loggers to carry out an investigation,</p>	<p><b>Photo editing</b> Manipulating digital images, and reflecting on the impact of the changes and whether the required purpose is fulfilled,</p>	<p><b>Repetition in games</b> 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	<p><b>Systems and searching</b> Recognising IT systems in the world and how some can enable searching on the internet.</p>	<p><b>Video production</b> Planning, capturing, and editing video to produce a short film.</p>	<p><b>Selection in physical computing</b> Exploring conditions and selection using a programmable microcontroller.</p>	<p><b>Flat-file databases</b> Using a database to order data and create charts to answer questions.</p>	<p><b>Introduction to vector graphics</b> Creating images in a drawing program by using layers and groups of objects.</p>	<p><b>Selection in quizzes</b> Exploring selection in programming to design and code an interactive quiz.</p>
Year 6	<p><b>Communication and collaboration</b> Exploring how data is transferred by working collaboratively online.</p>	<p><b>Webpage creation</b> Designing and creating webpages, giving consideration to copyright, aesthetics and navigation.</p>	<p><b>Variables in games</b> Exploring variables when designing and coding a game.</p>	<p><b>Introduction to spreadsheets</b> Answering questions by using spreadsheets to organise and calculate data.</p>	<p><b>3D modelling</b> Planning, developing, and evaluation 3D computer models of physical objects.</p>	<p><b>Sensing movement</b> Designing and coding a project that captures inputs from physical devices.</p>

## 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 database	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.	✓	✓						✓	✓		✓	

# Impact:

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). All learning objectives have been mapped to the strands, which ensures that units build on each other from one key stage to the next.

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.

All of the Teach Computing content is mapped to the ten-strand taxonomy, which covers the breadth of computing (see progression through the taxonomy). Within these strands, key elements of digital literacy have been identified, such as effective use of tools, impact of technology and safety and security. These strands are woven throughout the four key themes, with skills and knowledge applied across the teach computing curriculum.

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.

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 objectives 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 objectives using thumbs up, thumbs down, or thumbs sideways. This gives pupils a 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.

Every unit includes an optional summative assessment framework in the form of either a multiple choice quiz (MCQ) or a rubric. All units are designed to cover both skills and concepts from across the computing national curriculum. Units that focus more on conceptual development include a MCQ. Units that focus more on skills development end with a project and include a rubric. However, within the 'Programming' units, the assessment framework (MCQ or rubric) has been selected on a best fit basis. Each of the MCQ questions has been carefully chosen to represent the learning that should have been achieved within the unit. In writing the MCQs, we have followed the diagnostic assessment approach to ensure that the assessment of the unit is useful to determine both how well pupils have understood the content, and what pupils have misunderstood, if they have not achieved as expected. Each MCQ includes an answer sheet that highlights the misconceptions that pupils may have if they have chosen a wrong answer. This ensures teachers know which areas to return to in later units.

The rubric is a tool to help teachers assess project-based work. Each rubric covers the application of skills that have been taught directly across the unit, and highlights to teachers whether the pupil is approaching (emerging), achieving (expected), or exceeding the expectations for their age group. It allows teachers to assess projects that pupils have created, focussing on the appropriate application of computing skills and concepts. Pedagogically, we want to ensure that we are assessing pupils' understanding of computing concepts and skills, as opposed to their reading and writing skills. This has been carefully considered both in how the MCQs have been written (considerations such as the language used, the cultural experiences referenced, etc) and in the skills expected to be demonstrated in the rubric.