National Centre for Computing Education

Teacher Guide

Key Stage 2

A guide to the Teach Computing Curriculum

Funded by
Department for Education

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 ca 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/tc-feedback or by email to info@teachcomputing.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, createand 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.

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 to 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 key stage 2 are:

- Year 5 – Programming A – Selection in physical computing
- Year 6 – Programming B – Sensing movement

Your local Computing Hub may be able 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.

Curricular knowledge
Knowledge of our learners
Subject matter content
Effective learning experience
Teaching experience
Pedagogical knowledge

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

	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)

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

Unit summaries

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

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 evaluation 3D computer models of physical objects.	**Sensing movement** Designing and coding a project that captures inputs from physical devices.

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.	✓	✓						✓	✓		✓	

Progression

Progression through the taxonomy

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.

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.

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			

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. To support this, it is useful to know how the four key themes progress.

Digital Literacy

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.

Progression throughout the four themes

With the curriculum organised into four key themes, a spiral approach can be adopted (see 'Spiral curriculum' section for more information). This ensures skills and concepts progress from one year group to the next.

Computer Systems and Networks

The Computer Systems and Networks strand is taught once a year, building progressively from one year group to the next, with subject specific knowledge introduced at age-appropriate points.

Computer Systems and Networks	
1	Technology around us
2	IT around us
3	Connecting Computers
4	The Internet
5	Systems and Searching
6	Communication and Collaboration

Data and Information

The Data and Information strand is again taught once a year, progressing in both skills and software. Key Stage 1 uses simplified age-appropriate software platforms, progressing to more industry focused software in upper Key Stage 2.

Data and Information	
1	Grouping data
2	Pictograms
3	Branching databases
4	Data logging
5	Flat file databases
6	Introduction to spreadsheets

Programming

The Programming stand is taught twice a year, with the same concept revisited and covered in more depth. The following year incorporates the previous skills, whilst progressing onto a new concept.

Creating Media

The Creating Media strand hosts a wide range of different media types, and therefore different skills. To support progression, this can be best categorised into four different key areas: text, graphics (the use of pictures and text), photo and video, and audio. The spiral curriculum covers each of these four areas over a phase (KS1, LKS2 and UKS2), rather than in every year group, with links across these areas made where possible.

	Programming	
1	Moving a Robot	Programming animations
2	Robot algorithms	Programming quizzes
3	Sequencing sounds	Events and actions in programs
4	Repetition in shapes	Repetition in games
5	Selection in physical computing	Selection in quizzes
6	Variables in games	Sensing movement

	Creating Media			
	Text	Graphics	Photo and Video	Audio
1	Digital writing	Digital painting		
2			Digital photography	Digital music
3	Desktop publishing	Stop frame animation		
4		Photo editing		Audio production
5		Introduction to vector graphics	Video production	
6	Web page creation	3D modelling		

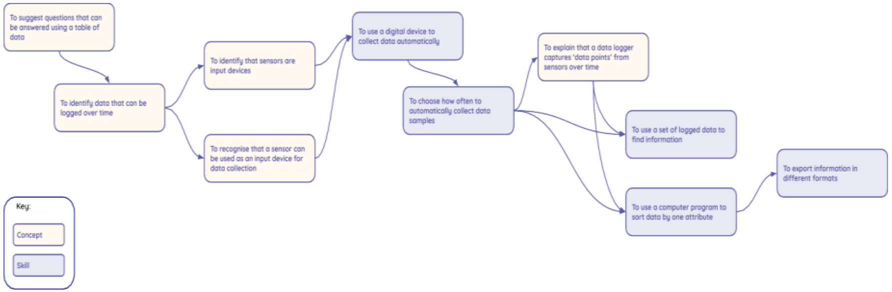
Progression within a unit — learning graphs

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.

KS2 example learning graph

Year 4 - Data and Information – Data logging


To suggest questions that can be answered using a table of data
To identify data that can be logged over time
To identify that sensors are input devices
To recognise that a sensor can be used as an input device for data collection
To use a digital device to collect data automatically
To choose how often to automatically collect data samples
To explain that a data logger captures 'data points' from sensors over time
To use a set of logged data to find information
To use a computer program to sort data by one attribute
To export information in different formats
Key:
Concept
Skill

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.

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

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 wth 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 move exploratory tasks. Adapting your instructions 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.

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 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 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

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.

Multiple choice quiz (MCQ)

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.

Rubric

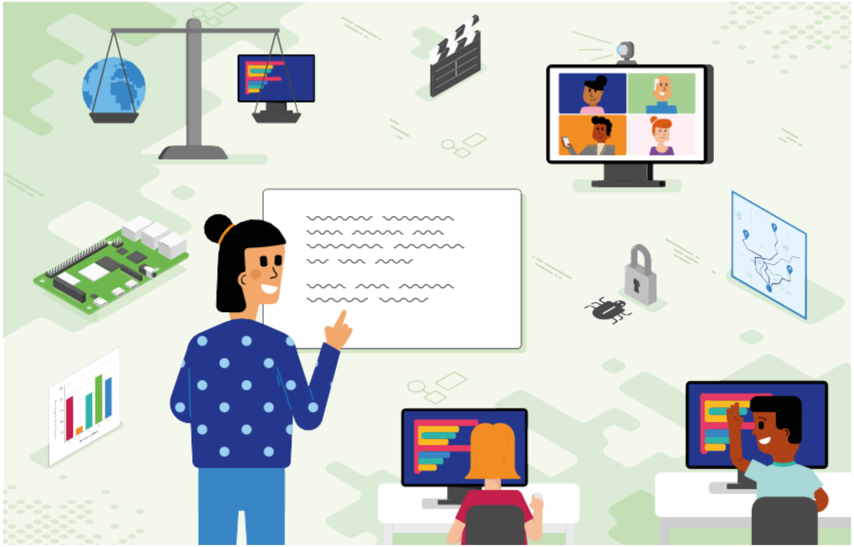
The rubric is a tool to help teachers assess project-based work. Each rubric covers the application of skills that have been taught directly across the unit, and highlights to teachers whether the pupil is approaching (emerging), achieving (expected), or exceeding the expectations for

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.

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 the 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 tip 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 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

✓Reflected in the unit screenshots ◆ Alternative software available

	Desktop or laptop	Chromebook	Tablet	Software or hardware
3.1 Connecting computers	✓	◆	◆	Painting program (any)
3.2 Stop-frame animation	◆	◆	✓	iMotion
3.3 Sequencing sounds	✓	✓	◆	Scratch
3.4 Branching databases	✓	✓	◆	J2data Branch and Pictogram
3.5 Desktop publishing	✓	✓	✓	Canva
3.6 Events and actions in programs	✓	✓	◆	Scratch
4.1 The internet	✓	✓	✓	Various websites
4.2 Audio production	✓	◆	◆	Audacity
4.3 Repetition in shapes	✓	◆	◆	FMS Logo
4.4 Data logging	✓	✓	✓	Data logger and associated software
4.5 Photo editing	✓	◆	◆	Paint.NET
4.6 Repetition in games	✓	✓	◆	Scratch

Software and hardware overview

Requirements for pupils - below

✓ Reflected in the unit screenshots

◆ Alternative software available

	Desktop or laptop	Chromebook	Tablet	Software or hardware
5.1 Systems and searching	✓	✓	◆	Google slides
5.2 Video production	✓	◆	◆	Any video editing software e.g iMovie or Canva
5.3 Selection in physical computing	✓	✓		Crumble controller starter kit + motors
5.4 Flat-file databases	✓	✓	◆	J2data Database
5.5 Introduction to vector graphics	✓	✓	◆	Google drawings or Microsoft PowerPoint
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

Adapting the curriculum for pupils with SEND

The Teach Computing Curriculum has been written to support all pupils, with units containing a number of scaffolding activities and utilising effective pedagogies to ensure high quality teaching. However, you may still need to adapt resources to enable some of your pupils, for example those with special educational needs and disabilities (SEND), to access lessons fully.

The following principles will help you make adaptations that benefit all learners, and these will be more effective if you identify clearly what it is your individual pupils need help with - do they have poor working memory that means that following instructions is more difficult, or do they need help to stay focussed when completing projects?

1. Identify essential learning and misconceptions: Determine the key learning in each unit that every child should know. Provide repeated opportunities for pupils to revisit this content in different ways. Identify any likely misconceptions and address these explicitly in lessons. For example, in the year 3 Animation unit, pupils tend to move characters too far between frames, so ensure this is highlighted and modelled well.

2. Pre-teach key vocabulary: Pre-teach the essential vocabulary for each unit, provide learners with a word list supported by images and use the vocabulary regularly throughout the unit with a consistent definition. Concentrate on a small number of terms and consider using a graphic organiser to highlight relationships between concepts, e.g. the Frayer model.

3. Create step-by-step instructions: Break down complex tasks and routine skills for using software and hardware into smaller steps and create pictorial instructions for children to follow. For example, in year 4 Audio Editing, you could create a handout with a sequence of instructions for trimming audio clips in Audacity based on the video guide.

4. Provide templates: In Creating Media or Data & Information unts, support task completion by providing a template for pupils to modify – removing the fear of the blank page and helping to build confidence. For example, in the year 6 Web Page Creation unit, you could set up a simple site with pages and navigation for pupils to fill in the content.

5. Consider non-computing barriers: Consider if difficulties in other areas, such as writing or maths, present barriers to completing a task and if so, modify the task to help mitigate these. For example, in the year 6 Communication and Collaboration unit, during lesson 3 where pupils work collaboratively to create slides, they could dictate content into the document rather than type it.

6. Use the PRIMM framework or Parson's problems: In programming units, add extra scaffolding using PRIMM and Parson's problems. Some pupils may not be able to create a program, but they can practise reading and exploring code in a working program, then modify it to make it more personalised. For example, in the year 4 Repetition in Games unit, for the final task learners can modify the bat catching game by changing the backdrop, adding a new costume to the sprites to change their appearance, and adding a different sound. The aim is to remove these scaffolds as children develop their skills, but some learners may not become fully independent.

7. Harness pupils' special interests: Increase engagement and make learning more relevant by incorporating pupils' special interests. This is also important in terms of culturally relevant pedagogy. For example, in the year 5 Vector Drawing unit, pupils could use what they have learnt to create a logo for their favourite sports team.

8. Use unplugged activities and the semantic wave: We can use unplugged activities to help make computing concepts more relevant and understandable for learners. However, it is very important to 'repack' the knowledge of the abstract concept so that learners understand what it means in a wider context and they can use the technical language. For example, in the year 5 Selection in Physical Computing unit pupils consider examples of selection in everyday life. Children then need to see the link between these examples and how this is used in a program and have the opportunity to use the key language in context. Pupils with SEND may need repeated examples and smaller steps to repack the knowledge, e.g. you could provide learners with some printed selection blocks from the Crumble software to add everyday examples to.

9. Support planning: Break down the planning process into smaller parts which can be ticked off as each one is completed, and provide a planning scaffold for learners where required. For example, when planning out algorithms for Scratch, provide printed versions of the blocks to manipulate and order, to help pupils to focus on only the code required

10. Reinforce digital skills: A significant barrier to accessing the whole computing curriculum is a lack of key digital skills, for example being able to log on to a computer and use the keyboard effectively. Time spent revisiting digital skills across all units is important to develop fluency. Some pupils may also benefit from extra time to practise these skills in small groups, or may need image-supported help sheets to support specific repeated tasks, such as saving work.

It is important that your adaptations are informed by effective formative assessment to identify any gaps in learning and the approach which may support with these.

A further resource which can support you is the Universal Design for Learning Framework from CAST which outlines a number of approaches to include all learners in lessons by providing multiple means of engagement, representation, action and expression.

For support from fellow teachers with individual units, head over to the STEM community and join the discussions about adapting units for pupils with SEND.

Finally, there is CPD available to support you further. Complete this online course to improve your knowledge: Creating an Inclusive Classroom: Approaches to Supporting Learners with SEND in Computing or attend the face-to-face course: Inclusive Computing in Primary Schools.

National Centre for Computing Education

The National Centre for Computing Education is the leading provider of support for computing education in England.

Funded by the Department for Education and operated by STEM Learning, our vision is to achieve a world-leading education for every child in England.

We provide 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 create the Teach Computing Curriculum: Our content writers, advisors, reviewers, pilot schools and every teacher who has taken the time to send us feedback.

Funded by

Department for Education