Teach Computing

Teacher Guide

Key Stage 3

A guide to the Teach Computing Curriculum

National Centre for Computing Education

Raspberry Pi

Contents

1 Introduction

1 Curriculum design
1 The approach
1 Coherence and flexibility
1 Knowledge organisation
2 Physical computing
2 Online safety
3 Core principles
3 Inclusive and ambitious
3 Research-informed
3 Time-saving for teachers

3 Structure of the units of work
4 Teach Computing Curriculum overview
4 National curriculum coverage
4 Teaching order

4 Progression
4 Progression across key stages
4 Progression within a unit — learning graphs

4 Pedagogy
5 Lead with concepts
5 Structure lessons
5 Make concrete
5 Unplug, unpack, repack
5 Work together
5 Read and explore code first
5 Create projects
6 Model everything
6 Get hands-on
6 Challenge misconceptions
6 Add variety
6 Foster program comprehension

6 Assessment
6 Formative assessment
7 Summative assessment
7 Multiple choice quiz (MCQ)
7 Rubric
7 Adapting for your setting

7 Resources
7 Software and hardware

8 National Centre for Computing Education

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

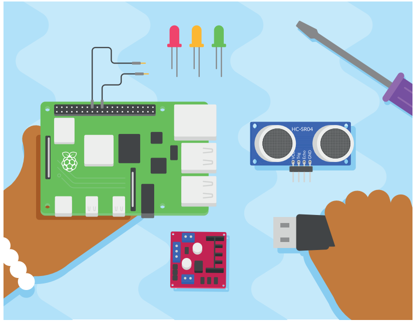
The taxonomy provides categories and an organised view of content to encapsulate the discipline of computing. Whilst all strands are present at all phases, they are not always taught explicitly.

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 9 – Physical computing
- GCSE – Programming part 5 – Strings and lists



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

Suggested teaching order

A suggested teaching order has been provided here. Within a year group, the order in which to teach units is not prescribed; however, the 'Programming' units should be taught in the order that is given in the suggested teaching order.

	Unit 1	Unit 2	Unit 3	Unit 4	Unit 5	Unit 6
Year 7	Impact of technology – Collaborating online respectfully (7.1)*	Networks: from semaphores to the internet (7.2)	Using media: gaining support for a cause (7.3)	Programming essentials in Scratch: part I (7.4)	Programming essentials in Scratch: part II (7.5)	Modelling data: spreadsheets (7.6)
Year 8	Developing for the web (8.1)	Representations: from clay to silicon (8.2)	Mobile app development (8.3)	Media: vector graphics (8.4)	Computing systems (8.5)	Introduction to Python programming (8.6)
Year 9	Python programming with sequences of data (9.1)	Media: animations (9.2)	Data science (9.3)	Representations: going audiovisual (9.4)	Cybersecurity (9.5)	Physical computing (9.6)

*The numbers in the brackets are a 'quick code' reference for each unit, eg 7.1 refers to the first Year 7 unit in the recommended teaching order.

Unit summaries

	Unit 1	Unit 2	Unit 3	Unit 4	Unit 5	Unit 6
Year 7	**Impact of technology: collaborating online respectfully** Identifying how to use online collaboration tools respectfully. An introduction to the computing lab.	**Networks: from semaphores to the internet** Recognising networking hardware and explaining how networking components are used for communication.	**Using media: gaining support for a cause** Creating a digital product for a real-world cause.	**Programming essentials in Scratch: part I** Applying the programming constructs of sequence, selection, and iteration in Scratch.	**Programming essentials in Scratch: part II** Using subroutines to decompose a problem that incorporates lists in Scratch.	**Modelling data: spreadsheets** Sorting and filtering data and using formulas and functions in spreadsheet software.
Year 8	**Developing for the web** Using HTML and CSS to create webpages.	**Representations: from clay to silicon** Representing numbers and text using binary digits.	**Mobile app development** Using event-driven programming to create an online gaming app.	**Media: vector graphics** Creating vector graphics through objects, layering, and path manipulation.	**Computing systems** Exploring the fundamental elements that make up a computer system.	**Introduction to Python programming** Applying the programming constructs of sequence, selection, and iteration in Python.
Year 9	**Python programming with sequences of data** Manipulating strings and lists. Creating a programming project.	**Media: animations** Creating 3D animations through object manipulation, and tweaking and adjusting lighting and camera angles.	**Data science** Using data to investigate problems and make real-world changes.	**Representations: going audiovisual** Representing images and sound using binary digits.	**Cybersecurity** Identifying how users and organisations can protect themselves from cyberattacks.	**Physical computing** Sensing and controlling with the micro:bit.

National Curriculum Coverage — Key Stage 3	7.1	7.2	7.3	7.4	7.5	7.6	8.1	8.2	8.3	8.4	8.5	8.6	9.1	9.2	9.3	9.4	9.5	9.6
Design, use and evaluate computational abstractions that model the state and behaviour of real-world problems and physical systems						✓			✓			✓	✓					✓
Understand several key algorithms that reflect computational thinking [for example, ones for sorting and searching]; use logical reasoning to compare the utility of alternative algorithms for the same problem				✓	✓				✓			✓	✓					✓
Use two or more programming languages, at least one of which is textual, to solve a variety of computational problems; make appropriate use of data structures [for example, lists, tables or arrays]; design and develop modular programs that use procedures or functions				✓	✓							✓	✓					✓
Understand simple Boolean logic [for example, AND, OR and NOT] and some of its uses in circuits and programming; understand how numbers can be represented in binary, and be able to carry out simple operations on binary numbers [for example, binary addition, and conversion between binary and decimal]				✓	✓						✓							

Table continues on next page

National Curriculum Coverage — Key Stage 3 (cont.)	7.1	7.2	7.3	7.4	7.5	7.6	8.1	8.2	8.3	8.4	8.5	8.6	9.1	9.2	9.3	9.4	9.5	9.6
Understand the hardware and software components that make up computer systems, and how they communicate with one another and with other systems		✓									✓							
Understand how instructions are stored and executed within a computer system; understand how data of various types (including text, sounds and pictures) can be represented and manipulated digitally, in the form of binary digits								✓			✓	✓	✓			✓		✓
Undertake creative projects that involve selecting, using, and combining multiple applications, preferably across a range of devices, to achieve challenging goals, including collecting and analysing data and meeting the needs of known users			✓			✓				✓					✓			
Create, reuse, revise and repurpose digital artefacts for a given audience, with attention to trustworthiness, design and usability	✓		✓	✓	✓		✓		✓	✓				✓				
Understand a range of ways to use technology safely, respectfully, responsibly and securely, including protecting their online identity and privacy; recognise inappropriate content, contact and conduct, and know how to report concerns	✓																✓	

Progression

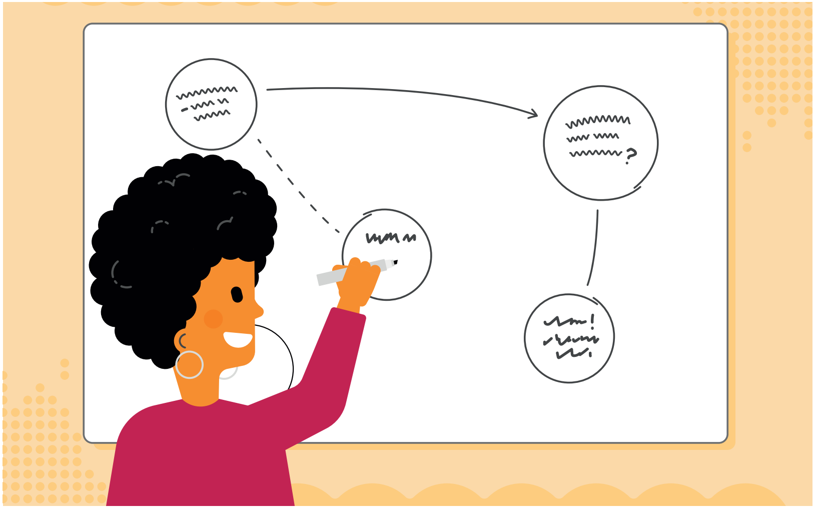
Progression across key stages

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.

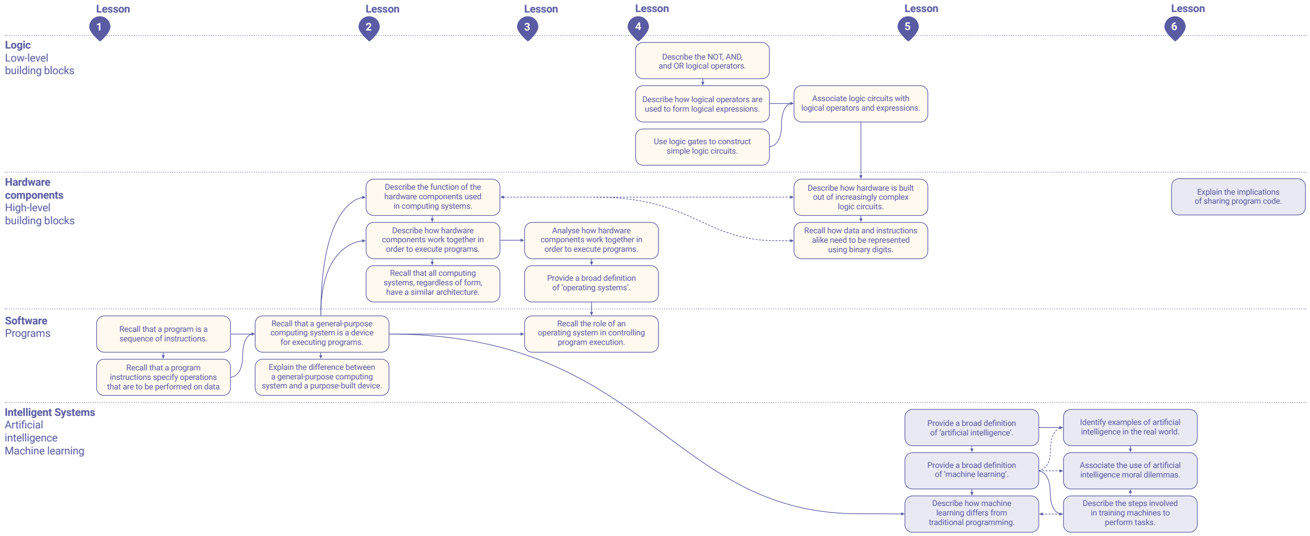
Progression within a unit — learning graphs

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.

Please note that the learning graphs often include statements with different wording than those shown in the lessons, as the learning graphs are designed for use by teachers, whereas the learning objectives are age-appropriate so that they can be understood by pupils.



Learning graph Year 8 – Computing Systems


Lesson 1
Lesson 2
Lesson 3
Lesson 4
Lesson 5
Lesson 6
Logic
Low-level building blocks
Describe the NOT, AND, and OR logical operators.
Describe how logical operators are used to form logical expressions.
Use logic gates to construct simple logic circuits.
Associate logic circuits with logical operators and expressions.
Hardware components
High-level building blocks
Describe the function of the hardware components used in computing systems.
Describe how hardware components work together in order to execute programs.
Recall that all computing systems, regardless of form, have a similar architecture.
Analyse how hardware components work together in order to execute programs.
Provide a broad definition of 'operating systems'.
Describe how hardware is built out of increasingly complex logic circuits.
Recall how data and instructions alike need to be represented using binary digits.
Explain the implications of sharing program code.
Software
Programs
Recall that a program is a sequence of instructions.
Recall that a program instructions specify operations that are to be performed on data
Recall that a general-purpose computing system is a device for executing programs.
Explain the difference between a general-purpose computing system and a purpose-built device.
Recall the role of an operating system in controlling program execution.
Intelligent Systems
Artificial intelligence
Machine learning
Provide a broad definition of 'artificial intelligence'.
Provide a broad definition of 'machine learning'.
Describe how machine learning differs from traditional programming.
Identify examples of artificial intelligence in the real world.
Associate the use of artificial intelligence moral dilemmas.
Describe the steps involved in training machines to perform tasks.

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 (ncce.io/qr07), and displays, along with regular recall and revision, can support this approach.

Structure lessons

Use supportive frameworks when planning lessons, such as PRIMM (Predict, Run, Investigate, Modify, Make — ncce.io/qr11) and Use-Modify-Create. These frameworks are based on research and ensure that differentiation can be built in at various stages of the lesson.

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.

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' (ncce.io/qr06), can help pupils develop a secure understanding of complex concepts.

Work together

Encourage collaboration, specifically using pair programming (ncce.io/qr03) and peer instruction (ncce.io/qr04), and also structured group tasks. Working together stimulates classroom dialogue, articulation of concepts, and development of shared understanding.

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.

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.

Model everything

Model processes or practices — everything from debugging code to binary number conversions — using techniques such as worked examples (ncce.io/qr02) and live coding (ncce.io/qr05). Modelling is particularly beneficial to novices, providing scaffolding that can be gradually taken away.

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.

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.

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.

Foster program comprehension

Use a variety of activities to consolidate knowledge and understanding of the function and structure of programs (ncce.io/qr12), including debugging, tracing, and Parson's Problems. Regular comprehension activities will help secure understanding and build connections with new knowledge.

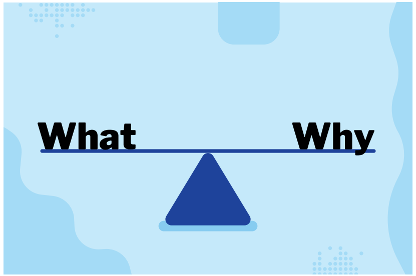
Assessment

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 are introduced in the slides at the beginning of every lesson. Every lesson has a starter activity and a plenary that can be used as an opportunity for formative assessment.

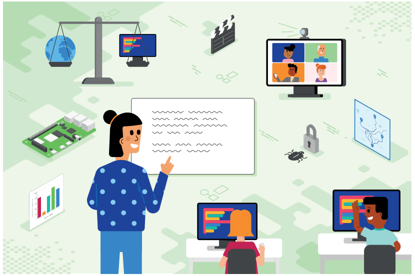


Summative assessment

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 an 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 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 that 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 directly taught 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 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 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.

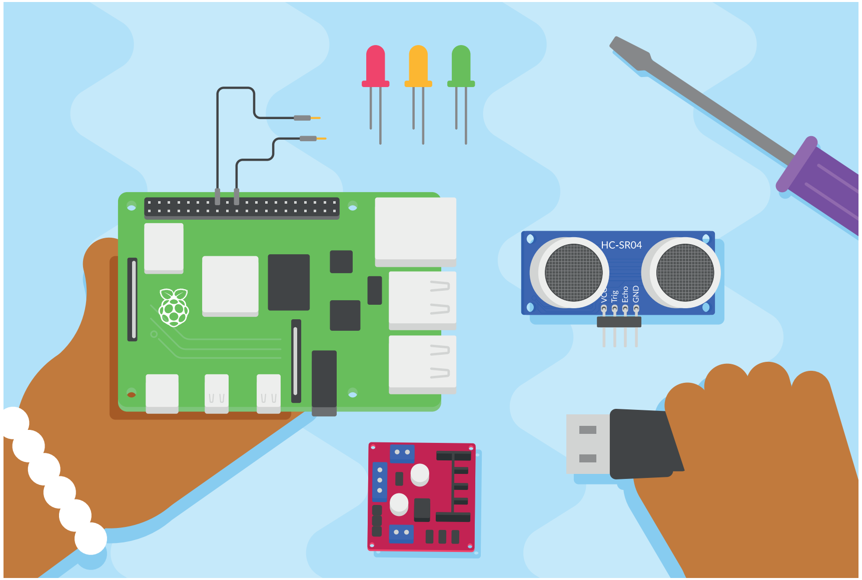
Resources

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.

All software used is either free for educational settings or open source. We have also ensured that any physical computing devices that are used are low-cost devices.

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 recommended for the lessons. A detailed list of hardware and software requirements for each unit is included on the next page:



Software and hardware

The Teach Computing Curriculum units require the use of a combination of hardware, software, and websites. Outlined below are:

- Specific hardware requirements
- Software that requires installation on the school network, or online software that requires pupils to have an account
- Websites that will need to be accessed by pupils during the unit

Note: It may be useful to make the manager of your network aware of all hardware, software, and website requirements before delivering a unit to a class.

	Software or hardware	Websites
7.1 – Impact of technology: collaborating online respectfully	■ School email account ■ Presentation software with built-in online collaboration tools (eg Google Slides)	■ www.youtube.com/watch?time_continue=20&v=opRMrEfAIiI&feature=emb_logo ■ www.ncsc.gov.uk/cyberaware/home ■ www.security.org/how-secure-is-my-password ■ www.youtube.com/watch?v=OBg2YYV3Bts&feature=emb_logo ■ www.thinkuknow.co.uk/11_13/help/CEOP ■ www.childline.org.uk ■ www.thinkuknow.co.uk/11_13/help/Contact-social-sites ■ www.anti-bullyingalliance.org.uk ■ www.bullying.co.uk/cyberbullying ■ www.ditchthelabel.org

	Software or hardware	Websites
7.2 – Networks: from semaphores to the internet		■ www.bbc.co.uk/bitesize/guides/z36nb9q/revision/2 ■ www.nibusinessinfo.co.uk/content/benefits-computer-networks ■ www.speedtest.net ■ www.youtube.com/watch?v=Dxcc6ycZ73M ■ www.submarinecablemap.com ■ www.youtube.com/watch?v=ewrBalT_eBM ■ lifehacks.io/facts-about-the-internet ■ www.youtube.com/watch?v=ZTM9GA-4nBA ■ seotribunal.com/blog/google-stats-and-facts ■ www.bbc.co.uk ■ www.lifewire.com/most-common-tlds-internet-domain-extensions-817511 ■ www.yougetsignal.com/tools/network-location/
7.3 – Using media: gaining support for a cause	■ Word processing software (eg Google Docs or Microsoft Word) ■ Software to create a blog (eg word processing software, Google Sites, Microsoft Sway)	■ www.youtube.com/watch?v=q0VzUigrb_g&feature=emb_logo ■ creativecommons.org/choose ■ search.creativecommons.org ■ foodhero.com/en/blogs/reduce-meat-consumption ■ www.plasticpollutioncoalition.org ■ www.conserve-energy-future.com/various-deforestation-facts.php ■ news.sky.com/story/sheep-registered-as-pupils-in-bid-to-save-classes-at-french-alps-primary-school-11714338

	Software or hardware	Websites
7.3 – Using media: gaining support for a cause (cont.)		■ en.wikipedia.org/wiki/Wikipedia:About#Strengths,_weaknesses,_and_article_quality_in_Wikipedia ■ computer.howstuffworks.com/internet/basics/wiki1.htm ■ www.livescience.com/7946-wikipedia-accurate.html ■ climate.nasa.gov/evidence ■ www.realclimate.org/index.php/archives/2019/04/first-successful-model-simulation-of-the-past-3-million-years-of-climate-change
7.4 – Programming essentials in Scratch: part I	■ Scratch	■ scratch.mit.edu ■ en.wikipedia.org/wiki/Five_Little_Ducks ■ en.wikipedia.org/wiki/Software_bug
7.5 – Programming essentials in Scratch: part II	■ Scratch	■ scratch.mit.edu
7.6 – Modelling data: spreadsheets	■ Spreadsheet software (eg Google Sheets or Microsoft Excel) ■ Online form creator (eg Google Forms or Microsoft Forms)	■ en.wikipedia.org/wiki/2016_Summer_Olympics_medal_table ■ en.wikipedia.org/wiki/2018%E2%80%9319_Premier_League ■ socialblade.com/youtube ■ www.weatheronline.co.uk/weather/maps/current?LANG=en&CONT=euro&LAND=UK&REGION=0003&SORT=2&UD=0&INT=06&TYP=niederschlag&ART=tabelle&RUBRIK=akt&DATE=--&CEL=C&SI=mph

	Software or hardware	Websites
8.1 – Developing for the web	■ A plain text editor for writing HTML and CSS (eg Windows Notepad, or Repl.it as an online alternative)	■ www.w3schools.com/html ■ www.w3schools.com/css ■ www.w3schools.com/cssref
8.2 – Representations: from clay to silicon		■ scratch.mit.edu ■ en.wikipedia.org ■ teachinglondoncomputing.org/lego-braille ■ csunplugged.org/en ■ csfieldguide.org.nz/en ■ archive.org/details/advancementofl00baco/page/256 ■ curriculum.code.org ■ www.cs4fn.org ■ apcentral.collegeboard.org/pdf/ap-computer-science-principles-course-and-exam-description.pdf?course=ap-computer-science-principles ■ denninginstitute.com/pjd/GP/GP-site/welcome.html ■ www.youtube.com/watch?v=1GSjbWt0c9M&list=PL8dPuuaLjXtNlUrzyH5r6jN9ulIgZBpdo&index=6&t=0s ■ www.futurelearn.com/courses/how-computers-work

	Software or hardware	Websites
8.3 – Mobile app development	■ App Lab from Code.org (pupils will need accounts, which can be created by the teacher in advance)	■ code.org/educate/applab ■ support.code.org/hc/en-us/articles/115000488132-Creating-a-classroom-section ■ www.youtube.com/watch?v=EhkxDIr0y2U ■ www.youtube.com/watch?v=e1St8LB4VJA ■ www.youtube.com/watch?v=fypSGGZZfzM
8.4 – Media: vector graphics	■ Vector graphics editor (the resources in this unit have been written for Inkscape, which is open source and cross-platform: inkscape.org)	■ inkscape.org
8.5 – Computing systems		■ scratch.mit.edu ■ www.computerhistory.org ■ teachinglondoncomputing.org/resources/inspiring-unplugged-classroom-activities/the-intelligent-piece-of-paper-activity ■ thecrashcourse.com/courses/computerscience ■ www.youtube.com/watch?v=5ocq6_3-nEw ■ jessecrossen.github.io/ttsim ■ www.khanacademy.org/computing/computer-science#how-computers--work ■ en.wikipedia.org ■ youtu.be/DFBbSTvtpy4 ■ youtu.be/CO67EQ0ZWgA

	Software or hardware	Websites
8.5 – Computing systems (cont.)		■ youtu.be/n-zeeRLBgd0 ■ teachablemachine.withgoogle.com ■ experiments.withgoogle.com/collection/ai ■ quickdraw.withgoogle.com ■ machinelearningforkids.co.uk ■ projects.raspberrypi.org ■ code.org/oceans ■ royalsociety.org
8.6 – Introduction to Python programming	■ Python (we recommend the Mu IDE for desktop, or Repl.it for cloud-based)	■ repl.it ■ blog.teachcomputing.org/tag/pedagogy ■ pythontutor.com/visualize.html ■ trinket.io ■ projects.raspberrypi.org ■ docs.python.org/3
9.1 – Python programming with sequences of data	■ Python (we recommend the Mu IDE for desktop, or Repl.it for cloud-based)	■ repl.it ■ blog.teachcomputing.org/tag/pedagogy ■ pythontutor.com/visualize.html ■ trinket.io

	Software or hardware	**Websites**
9.1 – Python programming with sequences of data (cont.)		■ projects.raspberrypi.org ■ docs.python.org/3 ■ www.gutenberg.org/ebooks/345
9.2 – Media: animations	■ Blender (free open source 3D creation software)	■ www.youtube.com/channel/UCZFUrFoqvqlN8seaAeEwjlw ■ en.wikipedia.org/wiki/File:Agathaumas.ogv
9.3 – Data science		■ www.datawrapper.de ■ www.youtube.com/watch?v=f_6IEKqS2l0 ■ www.gapminder.org ■ berkeleyearth.lbl.gov/country-list ■ codap.concord.org ■ datashine.org.uk ■ naei.beis.gov.uk/emissionsapp ■ www.gaugemap.co.uk

	Software or hardware	Websites
9.4 – Representations: going audiovisual	■ Image manipulation software (the resources in this unit have been tailored for GIMP, which is free and open source) ■ Sound editing software (the resources in this unit have been tailored for Audacity, which is free and open source) ■ Python ■ Optional: Raspberry Pi with resistors, breadboard, jumper wires, 1 RGB LED ■ Optional: Sonic Pi	■ curriculum.code.org ■ csfieldguide.org.nz ■ www.cs4fn.org ■ www.youtube.com/watch?v=7Jr0SFMQ4Rs ■ pippin.gimp.org/image_processing/chap_dir.html ■ projects.raspberrypi.org ■ trinket.io/sense-hat ■ pythonhosted.org/sense-hat/api/#led-matrix ■ scratch.mit.edu ■ sonic-pi.net ■ soundcloud.com/the-british-library/first-recording-of-computer-music-1951-copeland-long-restoration ■ commons.wikimedia.org/wiki/File:Chopin_-_Nocturne_in_F_Minor_variation.mid ■ www.vintagecomputermusic.com/mp3/s2t9_Computer_Speech_Demonstration.mp3 ■ web.archive.org/web/20000407081031/http://www.bell-labs.com/news/1997/march/5/2.html ■ en.wikipedia.org/wiki/MIDI ■ parametric.press/issue-01/unraveling-the-jpeg ■ classic.csunplugged.org ■ www.bbc.co.uk/bitesize/clips/zc2svcw ■ teachinglondoncomputing.org/compression-code-puzzles

	Software or hardware	Websites
9.5 – Cybersecurity		■ threatmap.checkpoint.com ■ scratch.mit.edu ■ forbusiness.snapchat.com/advertising#targeting ■ www.snap.com/en-GB/privacy/privacy-policy ■ help.instagram.com/519522125107875/?helpref=hc_fnav&bc[0]=Instagram%20Help&bc[1]=Privacy%20and%20Safety%20Center ■ policies.google.com/privacy#infocollect ■ en-gb.facebook.com/policy.php ■ www.ncsc.gov.uk ■ en.wikipedia.org/wiki/Hacktivism ■ en.wikipedia.org/wiki/2016_Dyn_cyberattack ■ www.cps.gov.uk/legal-guidance/computer-misuse ■ en.wikipedia.org/wiki/Computer_virus ■ en.wikipedia.org/wiki/Computer_virus#First_examples ■ us.norton.com/internetsecurity-malware-what-is-a-computer-virus.html ■ en.wikipedia.org/wiki/Computer_worm ■ us.norton.com/internetsecurity-malware-what-is-a-computer-worm.html ■ antivirus.comodo.com/blog/computer-safety/computer-worm-definition ■ en.wikipedia.org/wiki/Ransomware ■ www.malwarebytes.com/ransomware ■ uk.norton.com/internetsecurity-malware-ransomware-5-dos-and-donts.html

	Software or hardware	Websites
9.5 – Cybersecurity (cont.)		■ en.wikipedia.org/wiki/Trojan_horse_(computing) ■ us.norton.com/internetsecurity-malware-what-is-a-trojan.html ■ www.kaspersky.co.uk/resource-center/threats/trojans ■ www.cybersecuritychallenge.org.uk/resources/careers
9.6 – Physical computing	■ Python (we recommend the Mu IDE for desktop, or python.microbit.org for cloud-based)	■ microbit.org ■ python.microbit.org ■ microbit-micropython.readthedocs.io/en/v1.0.1 ■ www.arm.com/resources/education/schools/content ■ blog.teachcomputing.org/tag/pedagogy ■ youtu.be/oNLf6aFYVoU

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.