National Centre for Computing Education

Teacher Guide

Key Stage 4

A guide to the Teach Computing Curriculum

Funded by

Department for Education

National Centre for Computing 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 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/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, 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

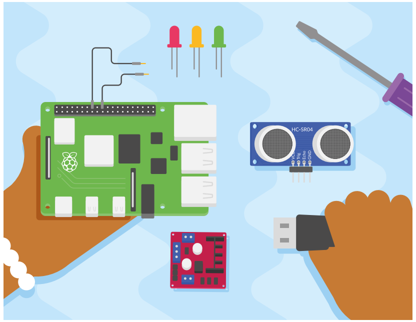
The taxonomy provides categories and an organised view of content to encapsulate the discipline of computing. Whilst all strands are present at all phases, they are not always taught explicitly.

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 – Developing physical computing projects
- 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

Brief GCSE overviews and suggested teaching orders

Year 10	Programming part 1 - Sequence	Programming part 2 - Selection	Programming part 3 - Iteration	Programming part 4 - Subroutines	Programming part 5 - Strings and lists	Algorithms part 2 - Searching and sorting (L4-12)
	Computer systems			Algorithms part 1	Data representations	
Year 11	Programming part 6 - Dictionaries and data files			Databases and SQL	HTML	
	Impacts of technology	Computer networks	Network security			
Optional	OOP					

Note that the top row of each section symbolises a double lesson

GCSE Unit summaries

Unit	Brief overview
Programming part 1 - Sequence (5 lessons)	Determine the need for translators. Use sequence, variables, and input in Python. Design programs using a flowchart.
Programming part 2 - Selection (6 lessons)	Use randomisation in programs. Work with arithmetic and logical expressions. Use selection and nested selection in Python.
Programming part 3 - Iteration (6 lessons)	Use a **while** loop and a **for** loop in Python. Perform validation checks on data entry. Design programs using pseudocode.
Programming part 4 - Subroutines (7 lessons)	Explain the differences between a procedure and a function. Describe scope of variables. Use functions and procedures as part of the structured approach to programming. Test a program for robustness.
Programming part 5 - Strings and lists (11 lessons)	Define the term 'graphical user interface' (GUI). Perform string handling operations. Describe the differences between a list and an array. Manipulate a list. Work with 2D lists.
Programming part 6 - Dictionaries and data files (12 lessons)	Use a record and a dictionary data structure. Access and modify external data files. Complete a complex programming project.

GCSE Unit summaries

Unit	Brief overview
Algorithms part 1 - The essentials (3 lessons)	Define the terms 'decomposition', 'abstraction', and 'algorithmic thinking'. Use trace tables.
Algorithms part 2 - Searching and sorting (9 lessons)	Describe a linear and binary search. Explain the key algorithms for a bubble, merge, and insertion sort.
Computer systems (13 lessons)	Describe the role of the CPU. Explain the processes of the fetch-decode-execute cycle. Determine the role of main memory and secondary storage. Construct truth tables for three input logic circuits. Write a program using assembly language (LMC).
Data representations (10 lessons)	Explain how numbers, text, images, and sound are represented using binary digits. Perform operations on binary digits. Convert between units of measurement.
Impacts of technology (8 lessons)	Determine the ethical, legal, environmental, and cultural impacts of technology.

GCSE Unit summaries

Unit	Brief overview
Computer networks (8 lessons)	Describe network components. Explain connectivity and distinguish between the various types. Describe the four layers of the TCP/IP model. Protect a network from threats.
Network security (7 lessons)	Describe the various ways that users and organisations can be affected by cyberattacks. Demonstrate how organisations can prevent cyberattacks.
Databases and SQL (5 lessons)	Describe a database and list its key terms. Determine the difference between a flat file and a relational database. Use structured query language (SQL) to retrieve and update data in a database.
HTML (8 lessons)	Create a website using HTML and CSS.
OOP (5 lessons)	Define and apply the principles of object-oriented programming. Create a class in Python and use its attributes and methods.

Brief non-GCSE overview and suggested teaching order

	Unit 1	Unit 2	Unit 3	Unit 4	Unit 5
Year 10/11	Online safety	IT and the world of work	Media	Spreadsheets	IT project management

Non-GCSE unit summaries

Unit	Brief overview
Online safety (10 lessons)	Recognise ways to build a positive online reputation. Discuss the ethics surrounding big data. Identify fake news and explain why it exists. Describe the laws governing online content. Recognise illegal content and describe how to report it.
IT and the world of work (6 lessons)	Examine modern technology tools that assist with inclusivity and accessibility. Evaluate effective online communication and collaboration. Create a positive work environment for remote working.
Media (7 lessons)	Create pre-production planning materials. Create raster and vector graphics. Utilise the software required for digital video creation. Create a multi-page website using open source tools.
Spreadsheets (6 lessons)	Use functions, formulas, and formatting in a spreadsheet. Develop a spreadsheet for a given scenario.
IT project management (10 lessons)	Identify why project management is important and recognise the common tools used. Manage a project for a given scenario.

National Curriculum Coverage	GCSE															Non-GCSE				
	Programming part 1	Programming part 2	Programming part 3	Programming part 4	Programming part 5	Programming part 6	Algorithms part 1	Algorithms part 2	Computer systems	Data representations	Impacts of technology	Computer networks	Network security	Databases and SQL	HTML	Online safety	IT and the world of work	Media	Spreadsheets	IT project management
Develop their capability, creativity and knowledge in computer science, digital media and information technology.	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓		✓	✓	✓	✓
Develop and apply their analytic, problem-solving, design, and computational thinking skills.	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓					✓				✓	✓
Understand how changes in technology affect safety, including new ways to protect their online privacy and identity, and how to report a range of concerns.									✓				✓			✓				

GCSE specification coverage

The GCSE units are closely linked to the specifications of the four main examination boards: AQA, OCR, Edexcel, and Eduqas. The GCSE units provide near full coverage of the specifications, regardless of the examination board that you have chosen for your school. Learning objectives set out in GCSE lessons in the Teach Computing Curriculum are not specific to an examination board, to ensure that pupils gain a broad understanding of computer science fundamentals.

Unit	AQA	OCR	Edexcel	Eduqas
Programming part 1 - Sequence	✓	✓	✓	✓
Programming part 2 - Selection	✓	✓	✓	✓
Programming part 3 - Iteration	✓	✓	✓	✓
Programming part 4 - Subroutines	✓	✓	✓	✓
Programming part 5 - Strings and lists	✓	✓	✓	✓
Programming part 6 - Dictionaries and data files	✓	✓	✓	✓
Algorithms part 1 - The essentials	✓	✓	✓	✓
Algorithms part 2 - Searching and sorting	✓	✓	✓	✓
Computer systems	✓	✓	✓	✓
Data representations	✓	✓	✓	✓
Impacts of technology	✓	✓	✓	✓
Computer networks	✓	✓	✓	✓
Network security	✓	✓	✓	✓
Databases and SQL	✓	✓		
HTML				
OOP				✓

Teaching order

GCSE computer science

The order in which to teach the GCSE units is not prescribed. You may wish to pick and choose the units that are suitable for your setting. However, a suggested teaching order has also been provided earlier in this document.

Non-GCSE computing

The order in which to teach the non-GCSE units is not prescribed. You may wish to pick and choose the units that are suitable for your setting. However, a suggested teaching order has also been provided earlier in this document.

Progression

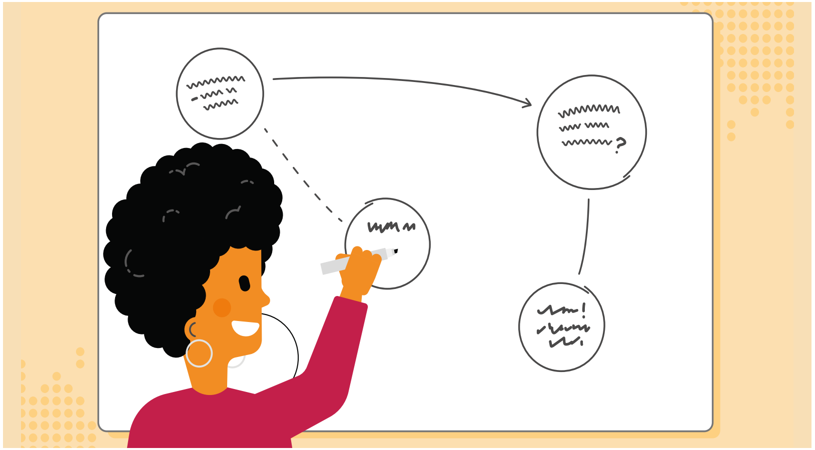
Progression across key stages

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.

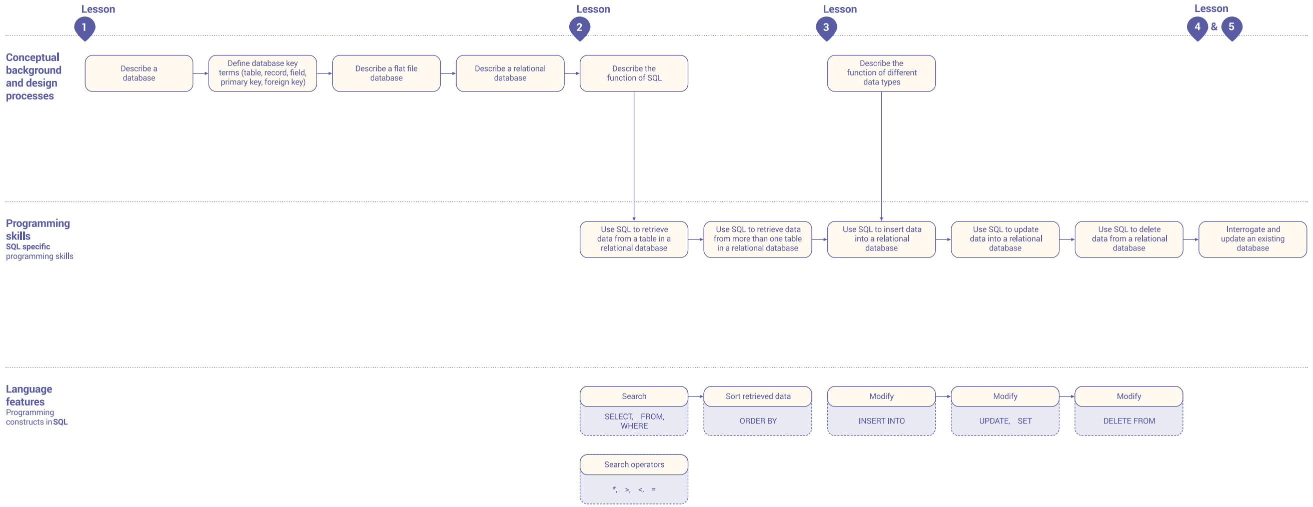
Progression within a unit — learning graphs

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 – Databases and SQL – KS4


Lesson 1
Lesson 2
Lesson 3
Lesson 4 & 5
Conceptual background and design processes
Describe a database
Define database key terms (table, record, field, primary key, foreign key)
Describe a flat file database
Describe a relational database
Describe the function of SQL
Describe the function of different data types
Programming skills
SQL specific programming skills
Use SQL to retrieve data from a table in a relational database
Use SQL to retrieve data from more than one table in a relational database
Use SQL to insert data into a relational database
Use SQL to update data into a relational database
Use SQL to delete data from a relational database
Interrogate and update an existing database
Language features
Programming constructs in SQL
Search
SELECT, FROM, WHERE
Sort retrieved data
ORDER BY
Modify
INSERT INTO
Modify
UPDATE, SET
Modify
DELETE FROM
Search operators
*, >, <, =

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.

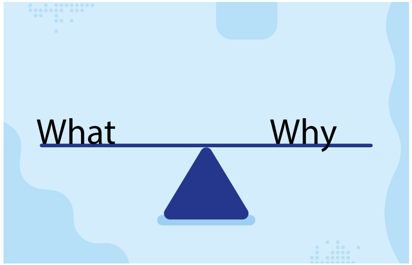
Assessment

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:

GCSE 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**
Programming part 1 - Sequence	■ Python (we recommend the Mu IDE for desktop, or Repl.it for cloud-based) ■ Flowgorithm (or software suitable to design flowcharts, eg Google Drawings or Google Slides)	■ youtu.be/Og847HVwRSI ■ ncce.io/ks4-scratchpath ■ repl.it ■ ncce.io/py-nameconventions
Programming part 2 - Selection	■ Python (we recommend the Mu IDE for desktop, or Repl.it for cloud-based)	■ repl.it ■ docs.python.org/3.7

	Software or hardware	Websites
Programming part 3 - Iteration	■ Python (we recommend the Mu IDE for desktop, or Repl.it for cloud-based)	■ repl.it ■ pythontutor.com/visualize.html
Programming part 4 - Subroutines	■ Python (we recommend the Mu IDE for desktop, or Repl.it for cloud-based) ■ Flowgorithm (or software suitable to design flowcharts, eg Google Drawings or Google Slides)	■ repl.it ■ www.python-course.eu/python3_global_vs_local_variables.php
Programming part 5 - Strings and lists	■ Python (we recommend the Mu IDE for desktop, or Repl.it for cloud-based) ■ ncce.io/ks4-senseHAT-emulator ■ Flowgorithm (or software suitable to design flowcharts, eg Google Drawings or Google Slides)	■ repl.it ■ scratch.mit.edu ■ ncce.io/guizero

	Software or hardware	Websites
Programming part 6 - Dictionaries and data files	■ Python (we recommend the Mu IDE for desktop, or Repl.it for cloud-based) ■ Flowgorithm (or software suitable to design flowcharts, eg Google Drawings or Google Slides)	■ repl.it ■ ncce.io/ks4-rpgproject
Algorithms part 1 - The essentials	■ Flowgorithm (or software suitable to design flowcharts, eg Google Drawings or Google Slides)	■ repl.it
Algorithms part 2 - Searching and sorting	■ Flowgorithm (or software suitable to design flowcharts, eg Google Drawings or Google Slides) ■ Python (we recommend the Mu IDE for desktop, or Repl.it for cloud-based)	■ repl.it
Computer systems	■ peterhigginson.co.uk/lmc	■ www.teachertoolkit.co.uk/2019/04/08/dual-coding ■ theteacher.info/index.php/fundamentals-of-cs/1-hardware-and-communication/topics/2599-registers-and-the-fetch-decode-execute-cycle ■ www.101computing.net/LMC ■ www.peterhigginson.co.uk/AQA

	Software or hardware	Websites
Computer systems (cont.)	■ peterhigginson.co.uk/lmc	■ www.youtube.com/watch?v=zDAYZU4A3w0 ■ blog.teachcomputing.org/quick-read-4-peer-instruction ■ pcpartpicker.com ■ repl.it ■ primming.wordpress.com ■ www.britannica.com/biography/George-Boole ■ en.wikipedia.org/wiki/A_Symbolic_Analysis_of_Relay_and_Switching_Circuits ■ ncce.io/csys-latch
Data representations	■ Spreadsheet software (eg Google Sheets or Microsoft Excel) ■ Painting software (eg Microsoft Paint 3D)	■ www.youtube.com/watch?v=yMfayttx6uw ■ www.w3schools.com/html/html_colors.asp ■ hexed.it ■ www.w3schools.com/html/html_colors_hex.asp ■ www.google.com/search?q=color+picker ■ www.youtube.com/watch?v=I-pQH_krD0M ■ www.unicode.org/charts/PDF/U2600.pdf ■ ncce.io/scratch-sound

	Software or hardware	Websites
Impacts of technology	■ No specific software or hardware requirements	■ youtu.be/a8fHgx9mE5U ■ youtu.be/s-HoCVAqcx4 ■ www.bbc.co.uk/news/magazine-30645383 ■ ico.org.uk/for-organisations/guide-to-freedom-of-information/refusing-a-request ■ edps.europa.eu/data-protection/data-protection/legislation/history-general-data-protection-regulation_en ■ news.bbc.co.uk/1/hi/education/1258446.stm ■ www.youtube.com/watch?v=2CVTH3_Bf-I ■ ncce.io/impt-areyouready ■ help.instagram.com/519522125107875 ■ ncce.io/L5ImpactsFacebook ■ www.bbc.co.uk/radio4/reith2005 ■ ncce.io/impt-theworldcounts ■ www.raspberrypi.org/blog/upgrade-culture ■ youtu.be/PUk4udSQGE0 ■ youtu.be/4sjS030x4Ig (note that the slide deck uses timings to avoid potentially inappropriate content) ■ www.moralmachine.net ■ www.theatlantic.com/technology/archive/2014/06/everything-we-know-about-facebooks-secret-mood-manipulation-experiment/373648 ■ www.youtube.com/watch?v=2ePf9rue1Ao

	Software or hardware	**Websites**
Computer networks	■ You can carry out network simulation using Packet Tracer: www.netacad.com/courses/packet-tracer	■ youtu.be/Cg_XeRSD6Rg ■ en.wikipedia.org/wiki/LoRa ■ www.ieee802.org/3 ■ www.ieee802.org/11 ■ en.wikipedia.org/wiki/Local_area_network ■ en.wikipedia.org/wiki/Wide_area_network ■ en.wikipedia.org/wiki/Personal_area_network ■ en.wikipedia.org/wiki/Bus_network ■ en.wikipedia.org/wiki/Ring_network ■ en.wikipedia.org/wiki/Star_network ■ en.wikipedia.org/wiki/Mesh_networking ■ en.wikipedia.org/wiki/Peer-to-peer ■ en.wikipedia.org/wiki/Client%E2%80%93server_model ■ en.wikipedia.org/wiki/Dynamic_Host_Configuration_Protocol ■ youtu.be/Dxcc6ycZ73M ■ youtu.be/fK7oAc_V-Kk ■ youtu.be/tT4AaelwvV4 ■ youtu.be/AEaKrq3SpW8

	Software or hardware	**Websites**
Network security	■ No specific software or hardware requirements	■ assets.publishing.service.gov.uk/government/uploads/system/uploads/attachment_data/file/813599/Cyber_Security_Breaches_Survey_2019_-_Main_Report.pdf ■ threatmap.checkpoint.com ■ www.bbc.co.uk/news/av/technology-48707033/ransomware-cyber-attacks-are-targeting-large-companies-and-demanding-huge-payments ■ www.youtube.com/watch?v=wWIwlfo9_OM ■ www.hacksplaining.com/exercises/sql-injection ■ www.youtube.com/watch?v=5v5gtycGTps ■ www.youtube.com/watch?v=88jkB1V6N9w&t=694s ■ inventwithpython.com/cipherwheel ■ cryptii.com/pipes/caesar-cipher-decoder ■ www.sporcle.com/games/g/passwords ■ www.securitymagazine.com/articles/89694-the-top-100-worst-passwords ■ www.youtube.com/watch?time_continue=91&v=o0RlmLuMP_s&feature=emb_logo ■ www.youtube.com/watch?v=rPlxvo3c_Zk ■ youtu.be/NlzEsMYW36A ■ youtu.be/l2iPGiiV2M4 ■ youtu.be/PWVN3Rq4gzw

	Software or hardware	**Websites**
Databases and SQL	■ Database software with SQL capabilities (SQLite is recommended: ncce.io/ks4-sqlitebrowser)	■ www.draw.io ■ www.w3schools.com/sql/exercise.asp
HTML	■ A suitable code editor such as Windows Notepad, TextEdit (macOS), or Geany (Raspberry Pi) ■ www.w3schools.com/html/html_editors.asp	■ www.w3schools.com ■ www.bbc.co.uk ■ validator.w3.org/#validate_by_input ■ developer.mozilla.org/en-US/docs/Web/CSS/CSS_Box_Model ■ developer.mozilla.org/en-US/docs/Web/CSS/float
OOP	■ Python (we recommend the Mu IDE for desktop, or Repl.it for cloud-based)	

Non-GCSE 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**
Online safety	■ No specific hardware or software requirements	■ www.dpocentre.com/difference-dpa-2018-and-gdpr ■ www.gov.uk/report-terrorism ■ www.report-it.org.uk/home ■ report.iwf.org.uk/en ■ torproject.org ■ vpnoverview.com

	Software or hardware	Websites
IT and the world of work	■ Word processing software with built-in collaboration tools (eg Google Docs)	■ informationisbeautiful.net ■ gsuite.google.co.uk ■ slack.com/intl/en-gb/features ■ evernote.com ■ www.cloudconnect.net ■ www.skype.com/en/business ■ www.webex.com/team-collaboration.html ■ www.dropbox.com/en_GB/features/content-collaboration ■ trello.com/en-GB ■ https://www.centenaryuniversity.edu/academics/academic-resources-advising/online-coursework-tips/online-etiquette/ ■ www.infoworld.com/article/2683784/what-is-cloud-computing.html ■ www.idc-online.com/technical_references/pdfs/data_communications/Ad_Hoc_Network.pdf ■ www.posturite.co.uk/help-advice/useful-resources/learning-guides/what-is-ergonomics

	Software or hardware	Websites
Media	■ Mind mapping software (choose one): ☐ docs.google.com/drawings ☐ www.mindmup.com ☐ www.edrawsoft.com/mindmaster ■ Mood board software (choose one): ☐ docs.google.com/drawings ☐ www.libreoffice.org (LibreOffice Draw) ☐ www.gimp.org ■ Optional: Storyboard software: ☐ wonderunit.com/storyboarder ■ Graphics editing software: ☐ www.gimp.org ☐ inkscape.org ■ Video editing software: ☐ www.openshot.org ■ Text editor: ☐ atom.io	■ dictionary.cambridge.org/dictionary/english/copyright ■ en.wikipedia.org/wiki/Creative_Commons_license ■ search.creativecommons.org

	Software or hardware	Websites
Spreadsheets	■ Spreadsheet software (eg Google Sheets or Microsoft Excel)	
IT project management	■ Spreadsheet software (eg Google Sheets or Microsoft Excel) ■ Drawing tool for creating a PERT chart (Google Slides is suitable for this) ■ Presentation software (eg Google Slides or Microsoft PowerPoint)	■ ncce.io/tenreasonswhy ■ ncce.io/workamjig ■ www.paymoapp.com/blog/what-is-a-gantt-chart ■ https://www.investopedia.com/terms/p/pert-chart.asp ■ www.visual-paradigm.com/project-management/gantt-chart-vs-pert-chart

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

This resource is available online at ncce.io/tcc.

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.

Funded by

Department for Education