Teacher Guide

Key Stage 3

A guide to The Computing Curriculum

Raspberry Pi Foundation

Contents

Introduction

The Computing Curriculum is our complete bank of free lesson plans and other resources that offer you everything you need to teach computing lessons to all school-aged learners. It helps you cover the full breadth of computing, including computing systems, programming, creating media, data and information, and societal impacts of digital technology.

The 500 hours of free, downloadable resources within The Computing Curriculum include all the materials you need in your classroom: from lesson plans and slide decks to activity sheets, homework, and assessments. To our knowledge, this is the most comprehensive set of free teaching and learning materials for computing and digital skills in the world.

The aims of The Computing Curriculum are as follows:

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

The Computing Curriculum resources are regularly updated in response to teachers' feedback. You can share your feedback at http://the-cc.io/feedback or by email to resourcesfeedback@raspberrypi.org.

Curriculum design

The approach

Coherence and flexibility

The 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 a particular order, with the exception of units on programming, where concepts and skills rely on learners' prior learning and experiences.

Knowledge organisation

The Computing Curriculum uses our taxonomy of computing content to ensure comprehensive coverage of the subject. The taxonomy provides a way to look at and describe the subject of computing as a set of interconnected topics; it doesn't define standards or curricula. There are, of course, many ways of organising the subject matter, implemented through exam specifications, textbooks, schemes of learning, and various progression guides. For our computing taxonomy, we reviewed examples of each of these from England and beyond. All learning outcomes can be described through our computing taxonomy of eleven strands, ordered alphabetically as follows:

- **Algorithms** — Be able to comprehend, design, create, and evaluate algorithms
- **Artificial intelligence** — Developing computer systems that determine the relationships between inputs and output in order to make predictions rather than following programmed instructions
- **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

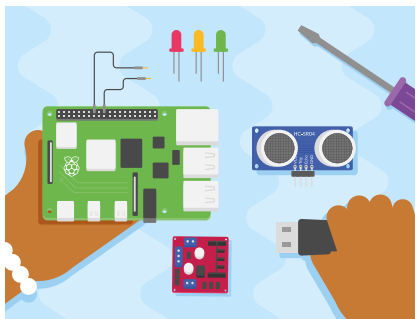
Our taxonomy provides categories and an organised view of content to encapsulate the discipline of computing. Whilst all strands are present at all phases, they are not always taught explicitly.

Physical computing

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

The physical computing units in The Computing Curriculum are:

- **Year 9** – Developing physical computing projects
- **GCSE** – Programming part 5 – Strings and lists
- **Non-GCSE** – Physical computing - Build a robot buggy



Online safety

The unit overviews for each unit show the links between the content of the lessons and England's national curriculum and the Education for a Connected World framework (the-cc.io/efacw). These references have been provided to show where aspects relating to online safety, or digital citizenship, are covered within The Computing Curriculum. Not all of the objectives in the Education for a Connected World framework are covered in The Computing Curriculum, as some are better suited to other subjects in England's education system. However, the coverage required for England's 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 Computing Curriculum has been written to support all learners. Each lesson is sequenced so that it builds on the learning from the previous lesson, and where appropriate, activities are scaffolded so that all learners can succeed and thrive. Scaffolded activities provide learners 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 learners to apply their learning in different contexts and make connections with other learning experiences.

As well as scaffolded activities, embedded within the lessons are a range of pedagogical strategies (defined in the 'Pedagogy' section of this document), which support making computing topics more accessible.

Research-informed

The subject of computing is much younger than many other subjects, and as such, there is still a lot more to learn about how to teach it effectively. To ensure that teachers are as prepared as possible, The 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 Computing Curriculum is reviewed each year and changes are made as necessary.

Time-saving for teachers

The Computing Curriculum has been designed to reduce teacher workload. To ensure this, The 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 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.

The Computing Curriculum overview

Brief overview

	Unit 1	Unit 2	Unit 3	Unit 4	Unit 5	Unit 6
Year 7	Impact of technology – Clear messaging in digital media (7.1)*	Networks – from semaphores to the internet (7.2)	Programming essentials in Scratch – part I (7.3)	Modelling data using spreadsheets (7.4)	Programming essentials in Scratch – part II (7.5)	Using media: gaining support for a cause (7.6)
Year 8	Media – Vector graphics (8.1)	Layers of computing systems (8.2)	Developing for the web (8.3)	Representations – from clay to silicon (8.4)	Mobile app development (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)	Introduction to cybersecurity (9.5)	Developing physical computing projects (9.6)

*The numbers in the brackets are a 'quick code' reference for each unit, e.g. 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	**Clear messaging in digital media** Combining the use of digital tools and online collaboration to produce media.	**Networks – from semaphores to the internet** Recognising networking hardware and explaining how networking components are used for communication.	**Programming essentials in Scratch – part I** Applying the programming constructs of sequence, selection, and iteration in Scratch.	**Modelling data using spreadsheets** Sorting and filtering data and using formulas and functions in spreadsheet software.	**Programming essentials in Scratch – part II** Using subroutines to decompose a problem that incorporates lists in Scratch.	**Using media: gaining support for a cause** Creating a digital product for a real-world cause.
Year 8	**Media – Vector graphics** Creating vector graphics through objects, layering, and path manipulation.	**Layers of computing systems** Exploring the fundamental elements that make up a computer system.	**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.	**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.	**Introduction to cybersecurity** Identifying how users and organisations can protect themselves from cyberattacks.	**Developing physical computing projects** Sensing and controlling with the micro:bit.

Coverage of England's national curriculum — Years 7 to 9	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	✓																✓	

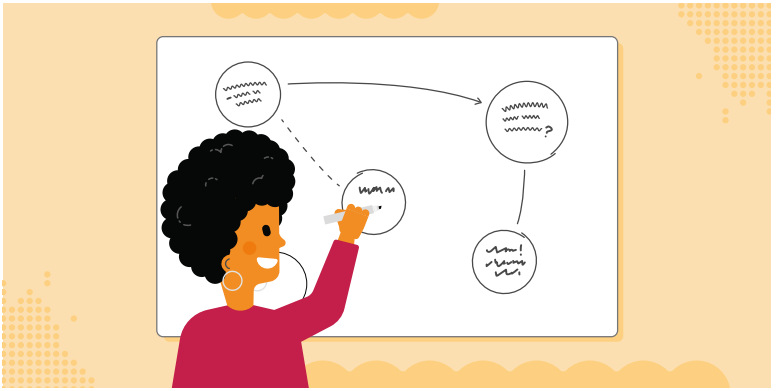
Progression

Progression across year groups

All learning objectives have been mapped to our computing taxonomy of eleven strands, which ensures that units build on each other from one year group to the next.

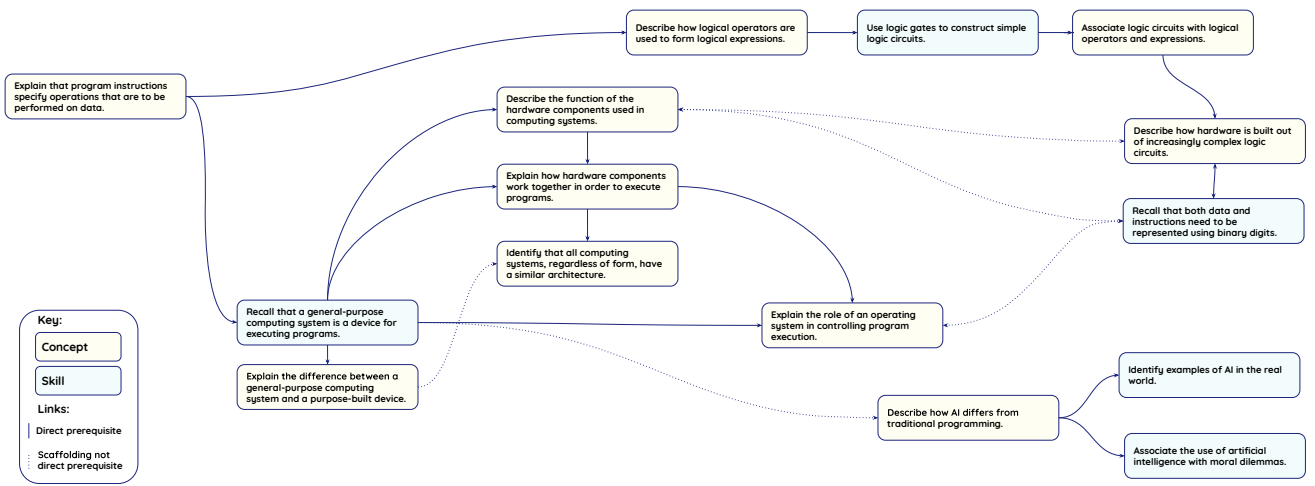
Progression within a unit — learning graphs

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, learners 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.

Learning graph Year 8 – Layers of computing systems



Pedagogy

Computing is a broad discipline, and computing teachers require a range of strategies to deliver effective lessons to their learners. Our 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 learners.

These 12 principles are embodied by The Computing Curriculum, and you can find examples of their application 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 Big Book of Computing Pedagogy* we have collated (the-cc.io/pedagogy).

Lead with concepts

Support learners 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 (the-cc.io/qr07), and displays, along with regular recall and revision, can support this approach.

Work together

Encourage collaboration, specifically using pair programming (the-cc.io/qr03) and peer instruction (the-cc.io/qr04), 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 learners 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' (the-cc.io/qr06), can help learners 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 (the-cc.io/qr02) and live coding (the-cc.io/qr05). 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 (the-cc.io/qr12), including debugging, tracing, and Parson's Problems. Regular comprehension activities will help secure understanding and build connections with new knowledge.

Create projects

Use project-based learning activities to provide learners with the opportunity to apply and consolidate their knowledge and understanding. Design is an important, often overlooked aspect of computing. Learners can consider how to develop an artefact for a particular user or function, and evaluate it against a set of criteria.

Add variety

Provide activities with different levels of direction, scaffolding, and support that promote learning, ranging from highly structured to more exploratory tasks. Adapting your instruction to suit different objectives will help keep all learners engaged and encourage greater independence.

Challenge misconceptions

Use formative questioning to uncover misconceptions and adapt teaching to address them as they occur. Awareness of common misconceptions alongside discussion, concept mapping, peer instruction, or simple quizzes can help identify areas of confusion.

Make concrete

Bring abstract concepts to life with real-world, contextual examples, and a focus on interdependencies with other curriculum subjects. This can be achieved through the use of unplugged activities, proposing analogies, storytelling around concepts, and finding examples of the concepts in learners' lives.

Structure lessons

Use supportive frameworks when planning lessons, such as PRIMM (Predict, Run, Investigate, Modify, Make — the-cc.io/qr11) and UMC (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 learners to review and interpret blocks of code. Research has shown that being able to read, trace, and explain code augments learners' ability to write code.

Assessment

Formative assessment

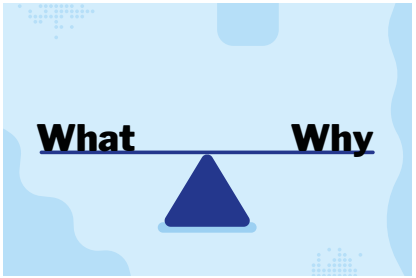
Every lesson includes formative assessment opportunities for you to use, and they are listed in the lesson plan. The formative assessments may be, for example, observations, questioning, or marked activities. We include these in every lesson to ensure that you can recognise and address learners' alternate conceptions if they occur. You can use the assessments to decide whether and how to adapt your teaching to suit the needs of the learners you are working with.

At the beginning of every lesson, the learning objective and success criteria are introduced in the slides.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. The summative assessment materials can inform your judgement around what a learner has understood in each computing unit, and could feed into your school's assessment process, to align with its approach to assessment in other foundation subjects.

All units in The Computing Curriculum are designed to cover both skills and concepts from across England's computing national curriculum. Units that focus more on conceptual development include MCQs as the optional summative assessment framework. Units that focus more on skills development end with a project and include a rubric. Within the 'Programming' units, we have selected the assessment framework (MCQs or rubric) on a best-fit basis.



The summative assessments are meant to give you insight into your learners' understanding of computing concepts and skills, as opposed to their reading and writing skills. To this end, we have created the MCQs and rubrics with great care. For the MCQs this involved, for example, carefully choosing the wording and cultural references. For the rubrics it involved making them focused on the purpose of application instead of the specific lesson context.

Multiple choice quiz (MCQ)

Each question in the MCQ has been designed to represent learning that learners are meant to achieve within the unit. In writing the MCQs, we have followed the diagnostic assessment approach to ensure that the assessment of the unit is useful for you to determine both how well your learners have understood the content, and what learners have misunderstood, if they have not achieved as expected.

Each MCQ includes an answer sheet that highlights the alternate conceptions that learners may have if they have chosen a wrong answer. This ensures that you know which areas to return to in later units.

Rubric

The rubric is a tool to help you assess project-based work. Each rubric covers the application of skills that have been directly taught across a unit, and highlights to you whether the learner is approaching (emerging), achieving (expected), or exceeding the expectations for their age group. The rubric allows you to assess whether learners have appropriately applied computing skills and concepts while they created their projects.

Adapting for your setting

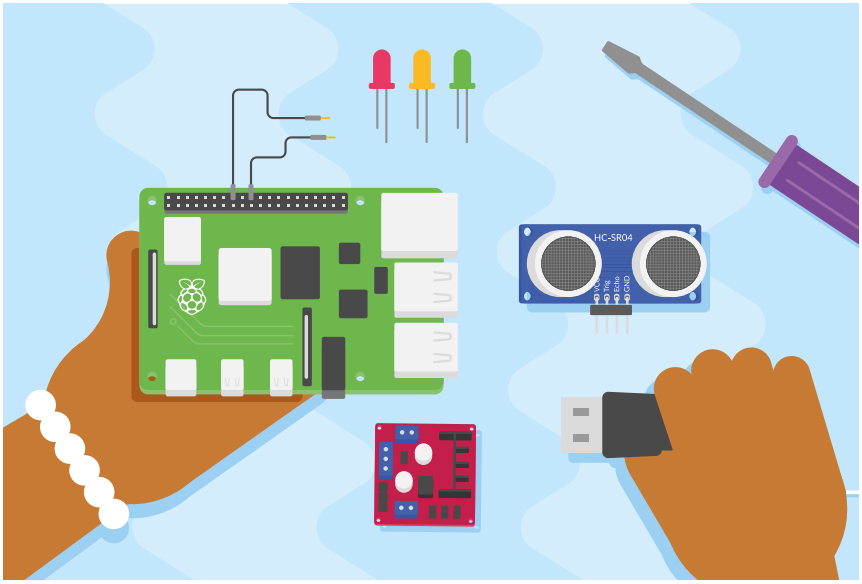
As there are no universally agreed levels of assessment, the assessment materials provided are designed to be used and adapted by schools in a way that best suits their needs. The summative assessment materials will inform your judgements around what a learner has understood in each unit, and could feed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

Resources

Software and hardware

Computing is intrinsically linked to technology and therefore requires that learners experience and use a range of digital tools and devices. As we wrote The Computing Curriculum, we carefully considered the hardware and software selected for the units. Our primary consideration was how we felt a tool would best allow learners to meet learning objectives; the learning always came first and the tool second. The learning objectives are not designed to be tool-specific.

To make the units of work more accessible to learners and teachers, the materials include screenshots, videos, and instructions, and these are based on the tools recommended for the lessons.



Software and hardware overview

The 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 learners 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 – Clear messaging in digital media	• Canva - Pupil accounts • Presentation software with built-in online collaboration tools (e.g. 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 – Programming essentials in Scratch: part I	• Scratch	• scratch.mit.edu • en.wikipedia.org/wiki/Five_Little_Ducks • en.wikipedia.org/wiki/Software_bug

	Software or hardware	Websites
7.4 – Modelling data using spreadsheets	• Spreadsheet software (e.g. Google Sheets or Microsoft Excel) • Online form creator (e.g. 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
7.5 – Programming essentials in Scratch - part II	• Scratch	• scratch.mit.edu
7.6 – Using media - Gaining support for a cause (cont.)	• Word processing software (e.g. Google Docs or Microsoft Word) • Software to create a blog (e.g. 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 • 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

	Software or hardware	Websites
8.1 – Mobile app development	• App Lab from Code.org (learners 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=EhkxDlr0y2U • www.youtube.com/watch?v=e1St8LB4VJA • www.youtube.com/watch?v=fypSGGZZfzM
8.2 – 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
8.3 – Layers of 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 • youtu.be/n-zeeRLBgd0

	Software or hardware	Websites
8.3 – Layers of computing systems (cont.)		• teachablemachine.withgoogle.com • experiments.withgoogle.com/collection/ai • quickdraw.withgoogle.com • machinelearningforkids.co.uk • projects.raspberrypi.org • code.org/oceans • royalsociety.org
8.4 – Developing for the web	• A plain text editor for writing HTML and CSS (e.g. Windows Notepad, or Repl.it as an online alternative)	• www.w3schools.com/html • www.w3schools.com/css • www.w3schools.com/cssref
8.5 – 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

	Software or hardware	Websites
8.5 – Representations - from clay to silicon (cont.)		• 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
8.6 – Introduction to Python programming	• Python (we recommend the Mu IDE for desktop, or Repl.it for cloud-based)	• repl.it • 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 • pythontutor.com/visualize.html • trinket.io • projects.raspberrypi.org • docs.python.org/3 • www.gutenberg.org/ebooks/345

	Software or hardware	Websites
9.2 – Media - Animations	• Blender (free open source 3D creation software)	• www.youtube.com/channel/UCZFUrFoqvqIN8seaAeEwjlw • 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 – Developing physical computing projects	• 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 • youtu.be/oNLf6aFYVoU

Raspberry Pi Foundation

The Raspberry Pi Foundation is a UK-based charity with the mission to enable young people to realise their full potential through the power of computing and digital technologies.

Our vision is that every young person develops:

- The knowledge, skills, and confidence to use computers and digital technologies effectively in their work, community, and personal life; to solve problems and to express themselves creatively
- Sufficient understanding of societal and ethical issues to be able to critically evaluate digital technologies and their application, and to design and use technology for good
- The mindsets that enable them to confidently engage with technological change and to continue learning about new and emerging technologies

Our long-term goals

- Education: To enable any school to teach students about computing and how to create with digital technologies, through providing the best possible curriculum, resources, and training for teachers
- Non-formal learning: To engage millions of young people in learning about computing and how to create with digital technologies outside of school, through online resources and apps, clubs, competitions, and partnerships with youth organisations
- Research: To deepen our understanding of how young people learn about computing and how to create with digital technologies, and to use that knowledge to increase the impact of our work and advance the field of computing education

For more free support for teachers, including online courses to enhance your understanding of computing content and pedagogy, visit: raspberrypi.org/teach.

Resources are updated regularly - the latest version is available at: the-cc.io/curriculum.

Raspberry Pi Foundation