Teacher Guide

Key Stage 2

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 students' 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 across all year groups in The Computing Curriculum materials, they are not always taught explicitly.

Spiral curriculum

The units for key stages 1 and 2 are based on a spiral curriculum. This means that each of the themes is revisited regularly (at least once in each year group), and learners revisit each theme through a new unit that consolidates and builds on prior learning within that theme.

This style of curriculum design reduces the amount of knowledge lost through forgetting, as topics are revisited yearly. It also ensures that connections are made even if different teachers are teaching the units within a theme in consecutive years.

Physical computing

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 5** – Selection in physical computing, which uses a Crumble controller
- **Year 6** – Sensing movement, which uses a micro:bit

Online safety

For each unit, the unit overview document shows 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 The Computing Curriculum. Not all of the objectives in the Education for a Connected World framework are covered in the 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 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

	Computing systems and networks	Creating media	Programming A	Data and information	Creating media	Programming B
Year 3	Connecting computers (3.1)	Stop-frame animation (3.2)	Sequencing sounds (3.3)	Branching databases (3.4)	Desktop publishing (3.5)	Events and actions in programs (3.6)
Year 4	The internet (4.1)	Audio production (4.2)	Repetition in shapes (4.3)	Data logging (4.4)	Photo editing (4.5)	Repetition in games (4.6)
Year 5	Systems and searching (5.1)	Video production (5.2)	Selection in physical computing (5.3)	Flat-file databases (5.4)	Introduction to vector graphics (5.5)	Selection in quizzes (5.6)
Year 6	Communication and collaboration (6.1)	Webpage creation (6.2)	Variables in games (6.3)	Introduction to spreadsheets (6.4)	3D modelling (6.5)	Sensing movement (6.6)

[1] Networks are not part of England's key stage 1 national curriculum for computing, but the title is used as a strand across primary.

*The numbers in the brackets are a 'quick code' reference for each unit, e.g. 1.3 refers to the third Year 1 unit in the recommended teaching order.

Unit summaries

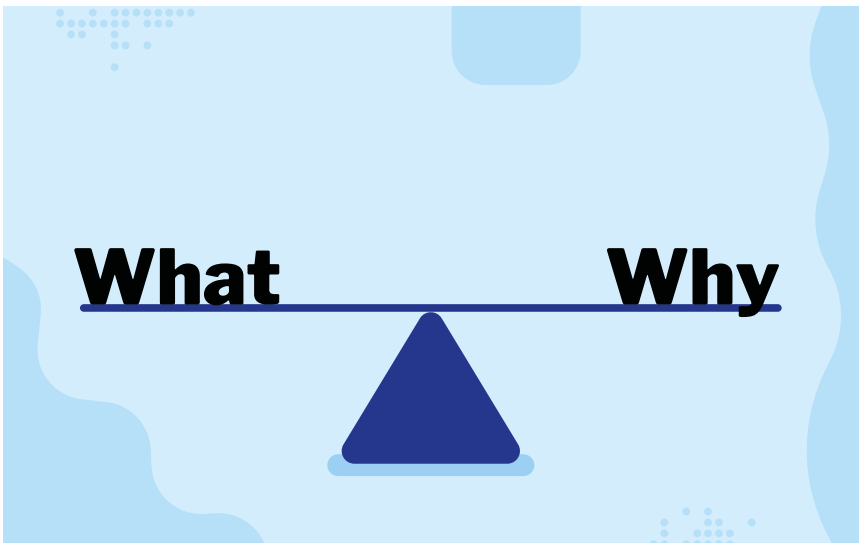
	Computing systems and networks	Creating media	Programming A	Data and information	Creating media	Programming B
Year 3	**Connecting computers** Identifying that digital devices have inputs, processes, and outputs, and how devices can be connected to make networks.	**Stop-frame animation** Capturing and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

Unit summaries

	Computing systems and networks	Creating media	Programming A	Data and information	Creating media	Programming B
Year 5	**Systems and searching** Recognising IT systems in the world and how some can enable searching on the internet.	**Video production** Planning, capturing, and editing video to produce a short film.	**Selection in physical computing** Exploring conditions and selection using a programmable microcontroller.	**Flat-file databases** Using a database to order data and create charts to answer questions.	**Introduction to vector graphics** Creating images in a drawing program by using layers and groups of objects.	**Selection in quizzes** Exploring selection in programming to design and code an interactive quiz.
Year 6	**Communication and collaboration** Exploring how data is transferred by working collaboratively online.	**Webpage creation** Designing and creating webpages, giving consideration to copyright, aesthetics, and navigation.	**Variables in games** Exploring variables when designing and coding a game.	**Introduction to spreadsheets** Answering questions by using spreadsheets to organise and calculate data.	**3D modelling** Planning, developing, and evaluating 3D computer models of physical objects.	**Sensing movement** Designing and coding a project that captures inputs from a physical device.

National curriculum coverage - Years 3 and 4	3.1 Connecting computers	3.2 Stop-frame animation	3.3 Sequencing sounds	3.4 Branching databases	3.5 Desktop publishing	3.6 Events and actions in programs	4.1 The internet	4.2 Audio production	4.3 Repetition in shapes	4.4 Data logging	4.5 Photo editing	4.6 Repetition in games
Design, write, and debug programs that accomplish specific goals, including controlling or simulating physical systems; solve problems by decomposing them into smaller parts			✓			✓			✓			✓
Use sequence, selection, and repetition in programs; work with variables and various forms of input and output	✓		✓			✓			✓	✓		✓
Use logical reasoning to explain how some simple algorithms work and to detect and correct errors in algorithms and programs			✓			✓			✓			✓
Understand computer networks, including the internet; how they can provide multiple services, such as the World Wide Web; and the opportunities they offer for communication and collaboration	✓						✓					
Use search technologies effectively, appreciate how results are selected and ranked, and be discerning in evaluating digital content					✓		✓	✓			✓	
Select, use, and combine a variety of software (including internet services) on a range of digital devices to design and create a range of programs, systems, and content that accomplish given goals, including collecting, analysing, evaluating, and presenting data and information	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Use technology safely, respectfully, and responsibly; recognise acceptable/unacceptable behaviour; identify a range of ways to report concerns about content and contact		✓		✓			✓	✓			✓	

National curriculum coverage - Years 5 and 6	5.1 Systems and searching	5.2 Video production	5.3 Selection in physical computing	5.4 Flat-file databases	5.5 Introduction to vector graphics	5.6 Selection in quizzes	6.1 Communication and collaboration	6.2 Webpage creation	6.3 Variables in games	6.4 Introduction to spreadsheets	6.5 3D modelling	6.6 Sensing movement
Design, write, and debug programs that accomplish specific goals, including controlling or simulating physical systems; solve problems by decomposing them into smaller parts			✓			✓	✓		✓			✓
Use sequence, selection, and repetition in programs; work with variables and various forms of input and output			✓			✓			✓			✓
Use logical reasoning to explain how some simple algorithms work and to detect and correct errors in algorithms and programs			✓			✓			✓			✓
Understand computer networks, including the internet; how they can provide multiple services, such as the World Wide Web; and the opportunities they offer for communication and collaboration	✓						✓					
Use search technologies effectively, appreciate how results are selected and ranked, and be discerning in evaluating digital content		✓		✓				✓				
Select, use, and combine a variety of software (including internet services) on a range of digital devices to design and create a range of programs, systems, and content that accomplish given goals, including collecting, analysing, evaluating, and presenting data and information	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Use technology safely, respectfully, and responsibly; recognise acceptable/unacceptable behaviour; identify a range of ways to report concerns about content and contact	✓	✓						✓	✓		✓	



Teaching order

The order in which to teach units within a school year is not prescribed, other than for the two 'Programming' units for each year group, which build on each other. It is recommended that the 'Programming' and 'Creating media' units be revisited in two different terms within the school year, so that the concepts and skills can be revisited and consolidated. Otherwise, schools can choose the order in which they teach the units, based on the needs of their learners and other topics or events that are happening throughout the school year, to make use of cross-curricular links wherever possible.

Mixed year groups

There are many different approaches to organising learners in school – one of which is mixed year group classes. The content throughout The Computing Curriculum is based on a learning progression from Year 1 through to Year 11 (ages 5–16). In order to use this progression with mixed year groups, , or any other school organisation system, we advise teachers to use the learning graphs for the age group which they are teaching to break up the content as they see fit.

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 across all year groups.

Within The Computing Curriculum materials for primary schools, every year group learns through units within the same four themes, which combine ten strands of the taxonomy (see table, right).

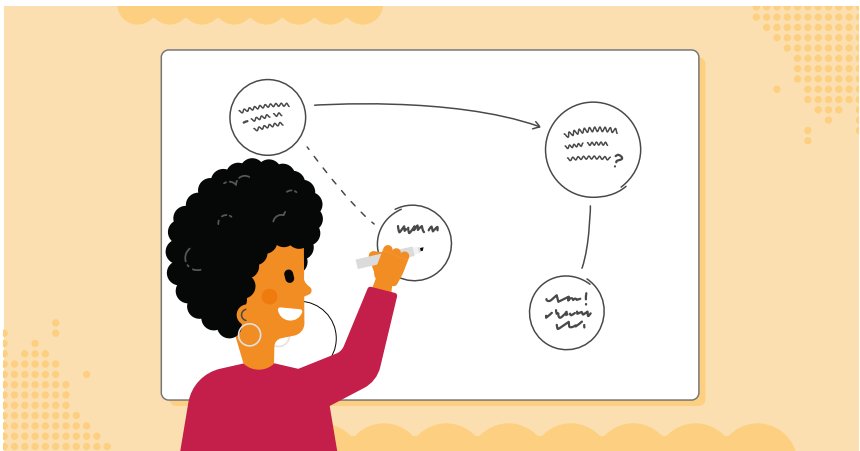
This approach allows us to use the spiral curriculum approach (see the 'Spiral curriculum' section for more information) to progress skills and concepts from one year group to the next.

Primary themes	Computing systems and networks	Programming	Data and information	Creating media
Taxonomy strands	Computer systems Computer networks	Programming Algorithms Design and development	Data and information	Creating media Design and development
	Effective use of tools			
	Impact of technology			
	Safety and security			

Progression within a unit — learning graphs

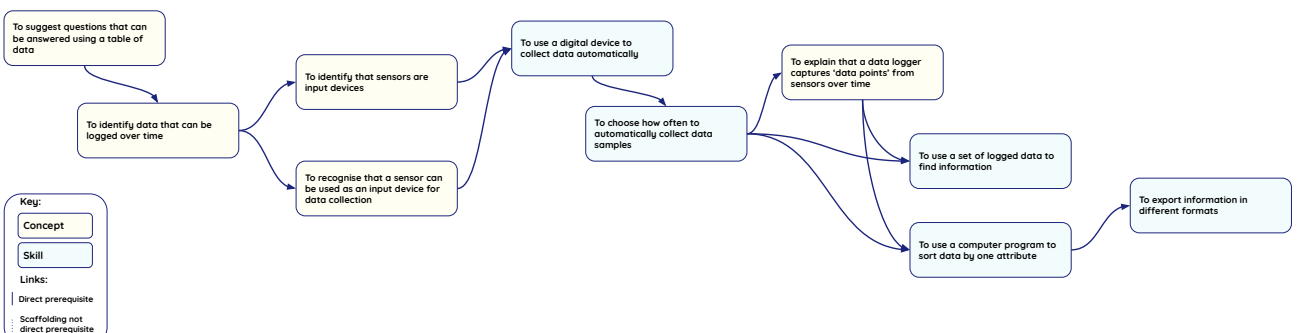
Learning graphs are provided as part of each unit and demonstrate progression through concepts and skills. In order to learn some of those concepts and skills, learners need prior knowledge of others, so the learning graphs show which concepts and skills need to be taught first and which could be taught 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 learners. In each year group, there are two 'Programming' units of work, but only one 'Programming' learning graph. The second 'Programming' unit builds on the content in the first 'Programming' unit so closely that there is no specific divide where one ends and the other begins.



KS2 Example learning graph

Year 4 - Data and information - Data logging



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 pupils' 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 pupils to review and interpret blocks of code. Research has shown that being able to read, trace, and explain code augments pupils' ability to write code.

Assessment

Formative assessment

Every lesson includes formative assessment opportunities for 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. At the end of every lesson, learners are invited to assess how well they feel they have met the learning objective using thumbs up, thumbs sideways, or thumbs down. This gives learners a reminder of the content that they have covered, as well as a chance to reflect. It is also a chance for you to see how confident your class is feeling so that you can make changes to subsequent lessons accordingly.

Summative assessment

Every unit includes an optional summative assessment framework in the form of either a multiple choice quiz (MCQ) or a rubric. 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.

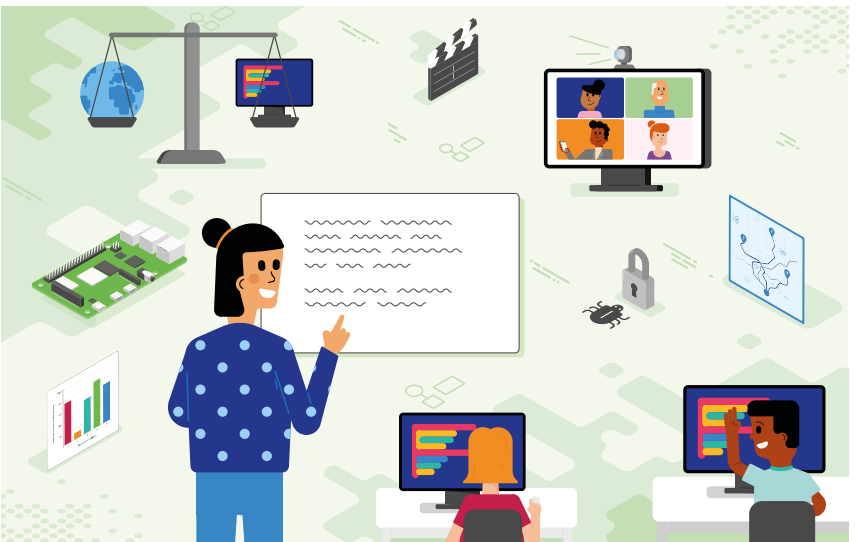
Each MCQ includes an answer sheet that highlights the alternate conceptions that learners may have if they have chosen a wrong answer. This ensures that you know which areas to return to in later units.

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 projects that learners have created, focussing on the appropriate application of computing skills and concepts.

Adapting for your setting

As there are no universally agreed levels of assessment, the assessment materials in The Computing Curriculum are designed to be used and adapted by schools in a way that best suits their needs. As mentioned above, the summative assessment materials 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 listed in the table below. The list should not be seen as an explicit requirement for schools. Schools may choose to use alternative tools that offer the same features as described in the units. All of the learning objectives can be met with alternative hardware and software, as the learning objectives are not designed to be tool-specific.

Hardware

Learners should experience using a range of digital devices, which may include desktop, laptop, and tablet computers. Learners should also experience using hardware designed for specific purposes, e.g. data loggers, floor robots, and microcontrollers.

Several of The Computing Curriculum units require the use of physical computing devices. This is in recognition of the growing importance of physical computing and digital making and was part of our curriculum design from the beginning.

Software

If you do not wish to use the software recommended in the units, you could use an alternative piece of software that provides the same function. All learning objectives should be achievable using alternative software, however, The Computing Curriculum will contain a lot less support if you choose an alterative, as screenshots and demonstration videos reflect the software referenced in the materials.

The units of work include the use of free software that would need to be installed on local computers, and software that is available as an online tool. Where software needs to be installed locally, schools will need to plan software installation in advance.

Several of the units that use online tools require schools to sign up to free services in order to access the tools. This also allows learners the opportunity to save the projects that they are working on, and gives them the skills that they need to manage their own usernames and passwords as digital citizens. However, you need to ensure that you are comfortable using the software and managing accounts, and that the software is in line with your school's policies about using online tools and how teachers will manage accounts.

Software and hardware overview

Requirements for learners — below

	Desktop or laptop	Chromebook	Tablet	Software or hardware
3.1 Connecting computers	✓	●	●	Painting program (any)
3.2 Stop-frame animation	●	●	✓	iMotion (app for iOS)
3.3 Sequencing sounds	✓	✓	●	Scratch
3.4 Branching databases	✓	✓	●	j2data Branch and Pictogram
3.5 Desktop publishing	✓	●	●	Canva.com
3.6 Events and actions in programs	✓	✓	●	Scratch
4.1 The internet	✓	✓	✓	Various websites
4.2 Audio production	✓			Audacity
4.3 Repetition in shapes	✓	●	●	FMSLogo
4.4 Data logging	✓	+	+	Data logger and associated software
4.5 Photo editing	✓	●		Paint.NET (for Microsoft Windows)
4.6 Repetition in games	✓	✓	●	Scratch

✓ Used for the unit — reflected in screenshots ● Could be used as an alternative + Data loggers that work with Chromebooks or tablets are available. Check with suppliers.

Software and hardware overview, cont.

Requirements for learners — below

	Desktop or laptop	Chromebook	Tablet	Software or hardware
5.1 Systems and Searching	✓	✓		Google Slides
5.2 Video production	✓	●	●	Microsoft Photos (for Microsoft Windows 10)
5.3 Selection in physical computing	✓	✓		Crumble controller + starter kit + motor
5.4 Flat-file databases	✓	✓	●	j2data Database
5.5 Introduction to vector graphics	✓	●		Google Drawings
5.6 Selection in quizzes	✓	✓		Scratch
6.1 Communication and collaboration	✓	✓		Google Slides
6.2 Webpage creation	✓	✓		Google Sites
6.3 Variables in games	✓	✓		Scratch
6.4 Introduction to spreadsheets	✓	✓	●	Google Sheets or Microsoft Excel
6.5 3D modelling	✓	✓	●	Tinkercad
6.6 Sensing movement	✓	✓	●	micro:bit and Microsoft MakeCode

✓ Used for the unit — reflected in screenshots ● Could be used as an alternative

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.

This resource is licensed by the Raspberry Pi Foundation under a Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International licence. To view a copy of this license, visit, see creativecommons.org/licenses/by-nc-sa/4.0/.

Raspberry Pi Foundation