Teacher Guide

Key Stage 4

A guide to The Computing Curriculum

Raspberry Pi Foundation

Contents

1 Introduction

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

5 Structure of the units of work
5 The Computing Curriculum overview
5 Brief GCSE overviews and suggested teaching orders
6 GCSE Unit summaries
9 Non-GCSE unit summaries
11 National curriculum coverage
11 National curriculum coverage — GCSE
11 National curriculum coverage — non-GCSE
12 GCSE specification coverage
12 Teaching order
13 GCSE computer science
13 Non-GCSE computing

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

15 KS4 example learning graph

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

18 Assessment
18 Formative assessment
18 Summative assessment
19 Multiple choice quiz (MCQ)
19 Rubric
19 Adapting for your setting

20 Resources
20 Software and hardware
21 GCSE software and hardware
29 Non-GCSE software and hardware

33 Raspberry Pi Foundation

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

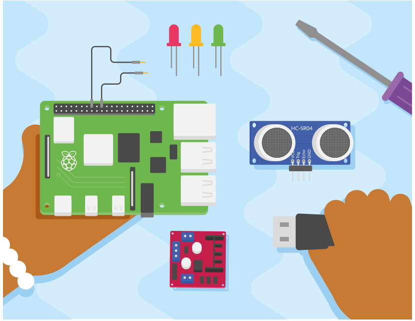
Our taxonomy provides categories and an organised view of content to encapsulate the discipline of computing. Whilst all strands are present at all phases, they are not always taught explicitly.

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.

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 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 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 (L1-3)	Data representations	
Year 11	Programming part 6 - Dictionaries and data files			Databases and SQL	HTML	
	Impacts of technology	Computer networks	Network security			
Optional	OOP	Physical computing - Build a robot buggy				

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 (15 lessons)	Use a record and a dictionary data structure. Access and modify external data files. Complete a complex programming project.

GCSE unit summaries continued

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 (18 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 continued

Unit	Brief overview
Computer networks (13 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 and CSS (8 lessons)	Create a website using HTML and CSS.
Object-oriented programming (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	Unit 6
Year 10/11	Online safety	IT and the world of work	Media	Physical computing - Build a robot buggy	Spreadsheets	Using IT in 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.

Non-GCSE unit summaries continued

Unit	Brief overview
Physical computing - Build a robot buggy (6 lessons)	Use a Raspberry Pi Pico microcontroller to explore inputs and outputs, and utilise a range of hardware components, including motors, reflective optical sensors, LEDs, and an ultrasonic sensor.
Spreadsheets (6 lessons)	Use functions, formulas, and formatting in a spreadsheet. Develop a spreadsheet for a given scenario.
Using IT in project management (10 lessons)	Identify why project management is important and recognise the common tools used. Manage a project for a given scenario.

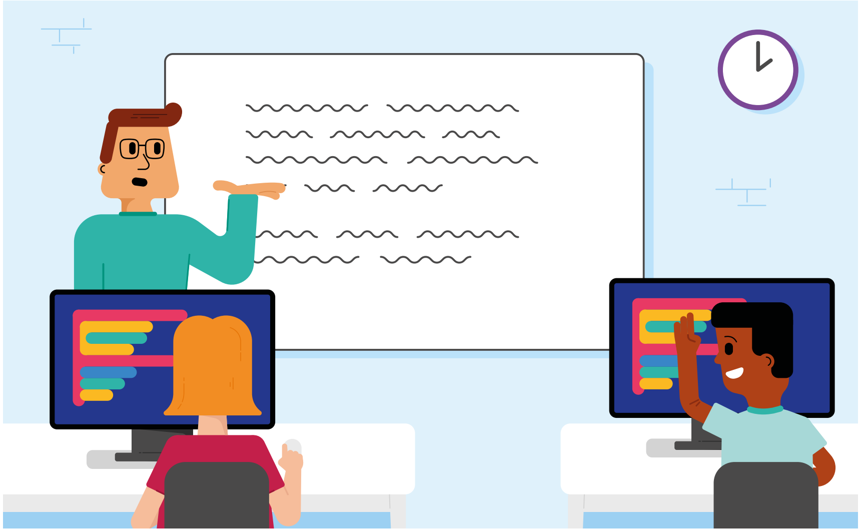
Coverage of England's national curriculum

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 - The Essentials	Algorithms part 2 - searching and sorting	Computer systems	Data representations	Impacts of technology	Computer networks	Network security	Databases and SQL	HTML	Object-oriented programming	Online safety	IT and the world of work	Media	Spreadsheets	Using IT in project management	Physical computing - Build a robot buggy
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 Computing Curriculum are not specific to an examination board, to ensure that learners 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				
Object-oriented programming				✓



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.

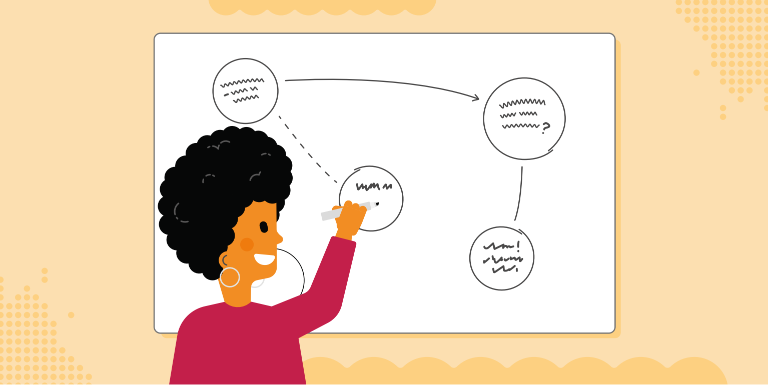
Progression

Progression across year groups

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

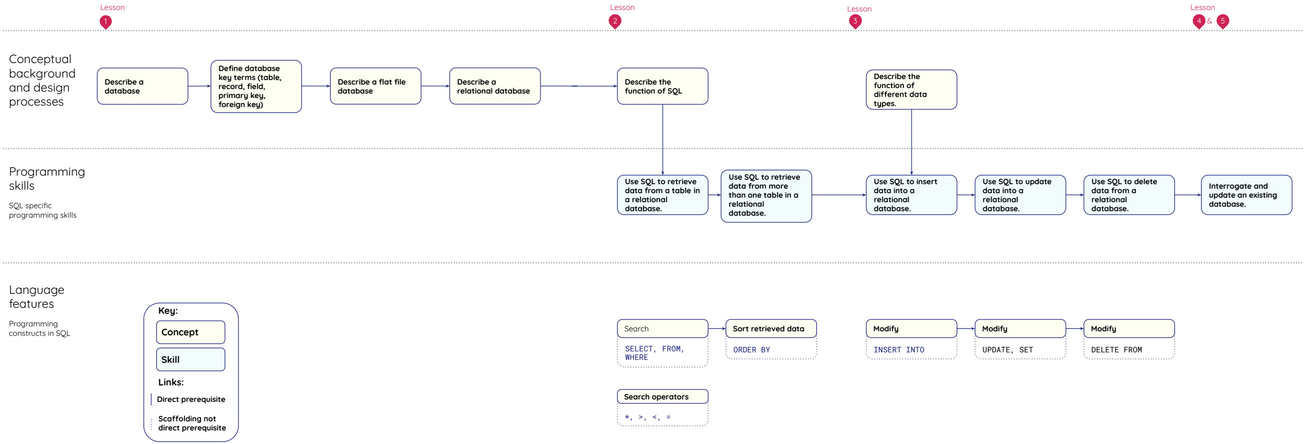
Progression within a unit — learning graphs

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 learners.

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`

Search operators

`*, >, <, =`

Modify

`INSERT INTO`

Modify

`UPDATE, SET`

Modify

`DELETE FROM`

Key:

Concept

Skill

Links:

Direct prerequisite

Scaffolding not direct prerequisite

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

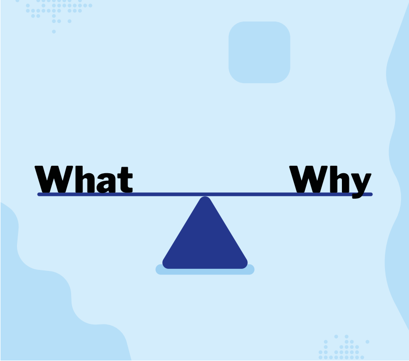
When teaching programming, focus first on code 'reading' activities, before code writing. With both block-based and text-based programming, encourage pupils to review and interpret blocks of code. Research has shown that being able to read, trace, and explain code augments pupils' ability to write code.

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.



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. 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.

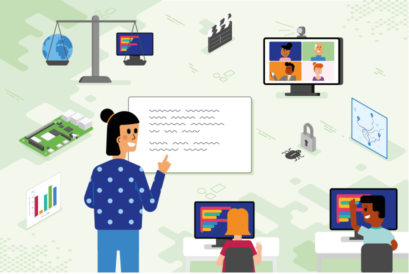
Each MCQ includes an answer sheet that highlights the alternate conceptions that learners may have if they have chosen a wrong answer. This ensures that you know which areas to return to in later units.

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 judgments 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.

GCSE software and hardware

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 learners need to access learners during a 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, e.g. Google Drawings or Google Slides)	• youtu.be/Og847HVwRSI • the-cc.io/ks4-scratchpath • repl.it • the-cc.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, e.g. 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) • the-cc.io/ks4-senseHAT-emulator • Flowgorithm (or software suitable to design flowcharts, e.g. Google Drawings or Google Slides)	• repl.it • scratch.mit.edu • the-cc.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, e.g. Google Drawings or Google Slides)	- repl.it - the-cc.io/ks4-rpgproject
Algorithms part 1 - The essentials	- Flowgorithm (or software suitable to design flowcharts, e.g. Google Drawings or Google Slides)	- repl.it
Algorithms part 2 - Searching and sorting	- Flowgorithm (or software suitable to design flowcharts, e.g. 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 • the-cc.io/qr04 • 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 • the-cc.io/csys-latch
Data representations	• Spreadsheet software (e.g. Google Sheets or Microsoft Excel) • Painting software (e.g. 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 • the-cc.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 • the-cc.io/impt-areyouready • help.instagram.com/519522125107875 • the-cc.io/L5ImpactsFacebook • www.bbc.co.uk/radio4/reith2005 • the-cc.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: the-cc.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
Object-oriented programming	• Python (we recommend the Mu IDE for desktop, or Repl.it for cloud-based)	

Non-GCSE software and hardware

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 learners need to access during a 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 (e.g. 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 (e.g. Google Sheets or Microsoft Excel)	
IT project management	- Spreadsheet software (e.g. Google Sheets or Microsoft Excel) - Drawing tool for creating a PERT chart (Google Slides is suitable for this) - Presentation software (e.g. Google Slides or Microsoft PowerPoint)	- the-cc.io/tenreasonswhy - the-cc.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
Physical computing - Build a robot buggy	- Thonny	

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