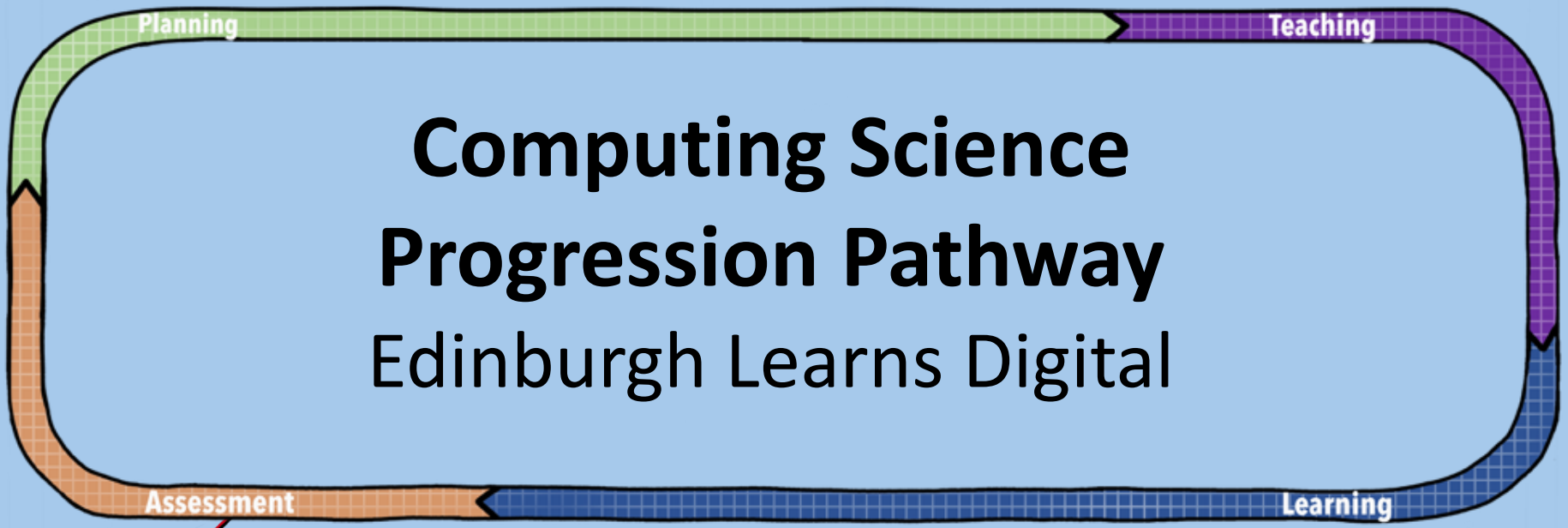
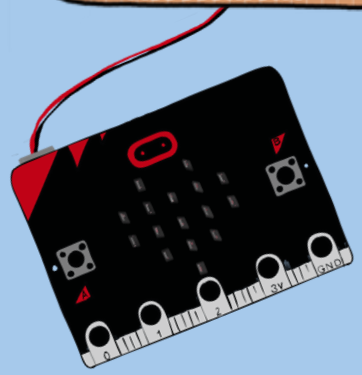
Computing Science Progression Pathway

Edinburgh Learns Digital

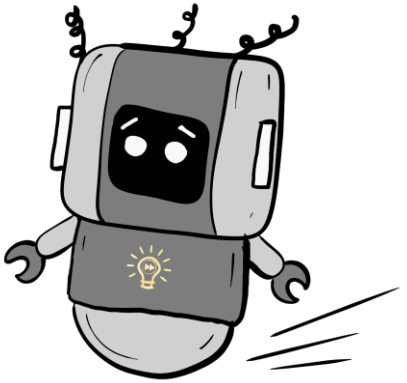




Contents

Background

This document has been created by Edinburgh Learns Digital to support schools to deliver progressive digital learning and teaching which should be embedded across the curriculum. The document aims to showcase a progression of digital skills organised alongside the Curriculum for Excellence's Technology Organisers and corresponding Benchmarks.



Devices have been provided to all (P6 - S6) learners and teachers ensuring equity of access to the transformative opportunities of digital learning: a shared model in P1 to P5, a 1:1 model from P6 to S6. Ubiquitous access to digital tools allows teachers to develop new pedagogical approaches and provides learners with new ways to access content and demonstrate their learning.

As educators we have a responsibility to support the development of digital skills for life, learning and work. This Progression Pathway provides assessment criteria through Early to Second Level, and suggested resources to support planning, delivery and assessment of learning.

What is Computing Science?

Source: *Teach Computing Science A Guide for Early Years and Primary Practitioners*

Understanding the World Through Computational Thinking
The underlying theory in the academic discipline of Computing Science. Theoretical concepts of Computing Science include the characteristics of information processes, identifying information, classifying and seeing patterns.

Understanding and Analysing Computing Technology
Insight into the hidden mechanisms of computers and the programs that run on them. It explores the different kinds of language, graphical and textual, used to represent processes and information. Some of these representations are used by people and others by machines. For example, a set of instructions could be represented as a verbal description, a sequence of blocks in a visual programming language such as Scratch, or as a series of 1s and 0s in binary.

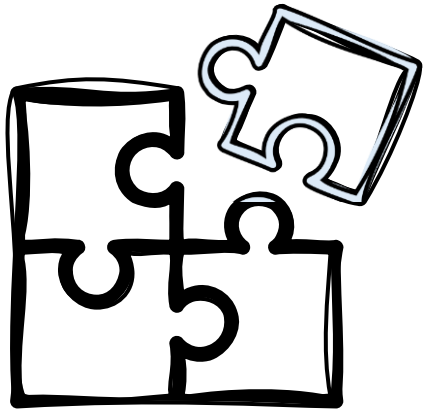
Designing, building and testing computing solutions
…taking the concepts and understanding from the first two Organisers and applying them. Learners will create solutions, perhaps by designing, building and testing solutions on a computer or by writing a computational process down on paper. In doing so, they will learn about modelling process and information from the real world in programs, and what makes a good model to represent or solve a particular problem.

Where it fits

Digital Literacy
Using technologies confidently, responsibly and critically to access, create and communication information

Food and Textile
Practical skills and understanding of food, nutrition, textiles and materials to support healthy, sustainable and informed lifestyle choices.

Technological Developments in Society and Business
Explore how technologies are developed and applied, and how they impact society, the economy, business and the environment



Computing Science
Develop problem-solving and computational thinking through understanding how computers and digital systems work, including programming and data representation.

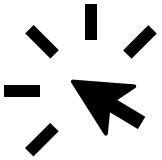
Edinburgh Learns Digital Strategy: Vision

In our education settings, digital learning is an intrinsic component of effective planning, teaching, learning and assessment. All learners and staff have ubiquitous access to high quality digital tools and resilient infrastructure to ensure fast and reliable online functionality. All learners and staff are digitally capable; they can create, communicate and collaborate digitally, transforming learning opportunities and enabling access to broader and more flexible curriculum pathways across all the city's education settings.

User Guide

- Each pathway is split across three phases to highlight progress through a level
- The pathways allow for responsive and flexible planning to support and challenge your learners, meaning they can and should move within and across progression levels. Some learners may be working before or after expected phases
- Progression pathways are skills-focussed with suggested resources included to allow staff the autonomy to select the most appropriate materials
- Many of the exemplified learning opportunities can be delivered unplugged, not requiring the use of computing technology
- Skills are highlighted in **bold** and can be used to support staff to form Learning Intentions and Success Criteria.
- For any enquiries regarding this pathway, please contact ELDigital@ea.edin.sch.uk
- The *Exemplified Learning & Resources* pages provides practical suggestions for developing these skills in your classroom and these have been categorised into ***Plugged*** and ***Unplugged*** activities.

Navigation - On each page, click the icons to jump to sections



Home/Contents Page	
Skills Progression inc. Benchmarks	Early First Second
Resources	RESOURCES
Glossary	A - Z

Benchmarks - Early

I can explore and comment on processes in the world around me making use of core computational thinking concepts and can organise information in a logical way **TCH 0-13a**	I understand that sequences of instructions are used to control computing technology. **TCH 0-14a** I can experiment with and identify uses of a range of computing technology in the world around me. **TCH 0-14b**	I can develop a sequence of instructions and run them using programmable devices or equivalent **TCH 0-15a**
• Identifies and sequences the main steps in an everyday task to create instructions/an algorithm for example, washing hands. • Classifies objects and groups them into simple categories for examples, groups toy bricks according to colour. • Identifies patterns, similarities and differences in objects or information such as colour, size and temperature and simple relationships between them.	• Demonstrates an understanding of how symbols can represent process and information. • Predicts what a device or person will do when presented with a sequence of instructions for example, arrows drawn on paper. • Identifies computing devices in the world (including those hidden in appliances and objects such as automatic doors).	• Designs a simple sequence of instructions/algorithm for programmable device to carry out a task for example directional instructions: forwards/backwards. • Identifies and corrects errors in a set of instructions.

Computing Science - Skills Progression

Organiser	Understanding the world through computational thinking	
E & O	I can explore computational thinking processes involved in a variety of everyday tasks and can identify patterns in objects or information	**TCH 0-13a**
Key Vocab	Instructions, Group, Similarities, Differences, Patterns, Classify, Sequence, Information	**RESOURCES**

Early

First

Second

Phase 1	Phase 2	Phase 3
Follow basic verbal/visual instructions regarding everyday tasks (e.g. washing hands) **Group** objects by simple given characteristics (e.g. colours, size) **Talk about** basic similarities and differences between objects (e.g. "these shapes are blue.")	**Follow** and **recall** basic instructions regarding everyday tasks (e.g. following daily routines) **Group** objects by multiple simple categories with support (e.g. all the green cars etc) **Identify** simple repeating patterns in visual sequences (e.g. dog, cat, dog, cat)	**Follow** and **sequence** basic instructions for everyday tasks (e.g. take off your coat *before* coming into the classroom) **Talk** about steps taken to complete everyday tasks (e.g. "What would you do first/then/next?") **Group** objects by multiple categories independently (e.g. animals with fur and four legs) **Continue** simple repeating patterns in sequences of information (e.g. blue, red, green, blue, red, green)

Computing Science - Skills Progression

Organiser	Understanding and analysing computing technology	
E & O	I understand **that sequences of instructions are used to control** computing technology / I can **experiment** with and **identify uses** of a **range of computing technology** in the world around me.	**TCH 0-14a/b**
Key Vocab	Symbols, Computing Device, Purpose, Sequence, Instructions, Function	**RESOURCES**

Early

First

Second

Phase 1	Phase 2	Phase 3
Start to **become aware** of everyday computing devices in their surroundings e.g. computers, traffic lights, torches, etc Start to **build** an awareness of different symbols/buttons and their actions (e.g. play button to play a video)	With adult support, **interact** with computing devices with an understanding of their purpose (e.g. taking a photo on an iPad) **Recognise** and **interact** with different symbols/buttons and describe their actions (e.g. power button switches on/off)	**Discuss** everyday computing devices and their purposes in their surroundings (e.g. tablets, projectors, smartboards) Begin to **interact** independently with **computing devices** to perform simple tasks (e.g. programme a Bee-Bot/Indi to follow a route) Begin to **identify** computing devices where the computer is hidden (e.g. a washing machine) **Explore** a sequenced set of instructions and **predict** the outcome (e.g. move forward two steps, backwards one and predict where you will stop)

Computing Science - Skills Progression

Organiser	Designing, building and testing computing solutions	
E & O	I can develop a sequence of instructions and run them using programmable devices or equivalent	**TCH 0-15a**
Key Vocab	Follow, Create, Identify	**RESOURCES**

Early

First

Second

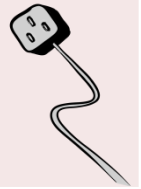
Phase 1	Phase 2	Phase 3
With adult support: **Follow** simple instructions (e.g. say "move forward or backwards") **Explore** basic directional language using visuals (e.g. chalk arrows for directions)	**Follow** simple directional instructions (e.g. "walk forward three steps") Begin to **use** directional language when playing to learn (e.g. "step, jump, forward, back") Begin to **explore** programmable/interactive toys when playing to learn (e.g. Bee-Bots, wind-up toys)	**Share** simple instructions in order (e.g. "move three steps forward, one step back") **Identify** and **fix** obvious errors in instructional language (e.g. five steps forward but not enough space) With adult support, begin to **explore** simple sequences with programmable toys (e.g. Bee-Bots)

Exemplified Learning & Resources

Early

	Exemplified Learning & Resources

I can explore computational thinking processes involved in a variety of everyday tasks and can identify patterns in objects or information
 	
TCH 0-13a
	Unplugged
	• In the classroom, use instructional language for everyday tasks (e.g. washing hands, daily routines)
	• Use toys to group items based on characteristics (e.g. group Duplo by colour and size)
	• Discuss patterns in sequences (e.g. nursery rhymes, fairy tales)
	• Ask learners to think about the similarities/differences between groups of objects (e.g. these are all Duplo blocks, but some are red, and others are blue)
	• Barefoot Computing
	  • Awesome Autumn - Exploring patterns
	  • Boats Ahoy - Six lessons exploring patterns and logic
	  • Busy Bodies - Simple algorithms
	  • People Who Help Us - Exploring patterns
	  • Super Space - Algorithms and spotting patterns
	  • Summer Fun - Grouping and sorting
	• Continue simple repeating patterns (e.g. songs, rhymes, marble runs, domino runs)
	
	Plugged
	• Watch short videos modelling daily routines (e.g. washing hands) and imitates steps
	• Play sorting games on an iPad or IWB (e.g. drag all blue shapes to one side).
	• Play sequence games on the iPad or IWB
	• Take photos on an iPad of similar or different objects
	• Record each other describing the steps and execution of a task using the iPad's voice recorder



Exemplified Learning & Resources

Early

	Exemplified Learning & Resources

I understand that sequences of instructions are used to control computing technology.
TCH 0-14a
 	
I can experiment with and identify uses of a range of computing technology in the world around me.
TCH 0-14b
	Unplugged
	• Tinker with play-based devices (e.g. torches, wind-up toys)
	• Recognise visual symbols and buttons (e.g. PE games with pause, play, stop, rewind)
	• Barefoot Computing
	  • Awesome Autumn - Exploring patterns
	  • People Who Help Us - Exploring patterns
	  • Boats Ahoy - Six lessons exploring patterns and logic
	  • Busy Bodies - Simple algorithms
	  • Super Space - Algorithms and spotting patterns
	  • Summer Fun - Grouping and sorting
	• Play with directional arrows to explore patterns, prediction and movement
	• Draw patterns and directional routes
	• Match direction words to arrow cards or pictures
	• Predict what happens before devices are used (e.g. pushing the power button on an interactive whiteboard)
	• Build the above into existing play to learn set-ups
	
	Plugged
	• Tinker with Bee-Bots or Sphero Indis in a play context - using direction cards
	• Understand that pushing a button can move a robot (e.g. Bee-Bot)
	• Discuss different devices in your setting and what they do (e.g. washing machines, iPads)
	• Take photographs on an iPad
	• Open apps on an iPad (e.g. voice memos app)
	• Tinker with different forms of technology (inc. torches, iPads, devices with on/off buttons)
	• Use an interactive whiteboard to play a variety of games
	• Use an interactive whiteboard drawing tools to create directional paths

Exemplified Learning & Resources

Early

	Exemplified Learning & Resources

I can develop a sequence of instructions and run them using programmable devices or equivalent
 	
TCH 0-15a
	Unplugged
	• Follow spoken instructions to move in a space ("Take two steps forward, turn left, then line-up")
	• Use chalk arrows on the playground to show directions and follows the path
	• Move toy figures in response to directional instructions
	• Simon Says style movement games reliant on directional instructions or sequences
	• Guide one another around classroom mazes (e.g. "Three steps forward, turn right, two steps forward"
	• Direct learners with a set of instructions to complete a task, then get them to do it again with the same instructions but in a different order. This highlights the importance of order.
	• Barefoot Computing
	  • Awesome Autumn - Exploring patterns
	  • Boats Ahoy - Six lessons exploring patterns and logic
	  • Busy Bodies - Simple algorithms
	  • People Who Help Us - Exploring patterns
	  • Super Space - Algorithms and spotting patterns
	  • Summer Fun - Grouping and sorting
	• Computing Science: Unplugged - Kidbot: Rescue Mission / Kidbot: Fitness Unplugged
	
	Plugged
	• Explore programmable toys e.g. program a Bee-Bot to follow a short route to reach a target and adjust if it crashes
	• Use Bee-Bots or wind-up toys to test how far it moves or in which direction it moves
	• Uses voice memo app to record or replay their set of instructions and discuss what went wrong
	• Explore Bee-Bot basics activities on Barefoot: Basics / Tinkering / 123 Programming

Benchmarks - First

I can explore and comment on processes in the world around me making use of core computational thinking concepts and can organise information in a logical way. **TCH 1-13a**	I understand the instructions of a visual programming language and can predict the outcome of a program written using the language. **TCH 1-14a** I understand how computers process information. **TCH 1-14b**	I can demonstrate a range of basic problem-solving skills by building simple programs to carry out a given task, using an appropriate language. **TCH 1-15a**
• Follows sequences of instructions/ algorithms from everyday situations for example, recipes or directions, including those with selection and repetition. • Identifies steps in a process and describes precisely the effect of each step. • Makes decisions based on logical thinking including IF, AND, OR and NOT for example, collecting balls in the gym hall but NOT basketballs, line up if you are left-handed OR have green eyes. • Collects, groups and orders information in a logical, organised way using my own and others‟ criteria (MNU 1-20a and b).	• Demonstrates an understanding of the meaning of individual instructions when using a visual programming language (including sequences, fixed repetition and selection). • Explains and predicts what a program in a visual programming language will do when it runs for example, what audio, visual or movement effect will result. • Demonstrates an understanding that computers take information as input, process and store that information and output the results.	• Simplifies problems by breaking them down into smaller more manageable parts. • Constructs a sequence of instructions to solve a task, explaining the expected output from each step and how each to contributes towards solving the task. • Creates programs to carry out activities (using selection and fixed repetition) in a visual programming language. • Identifies when a program does not do what was intended and can correct errors/bugs. • Evaluates solutions/programs and suggests improvements.

Computing Science - Skills Progression

Organiser	Understanding the world through computational thinking	
E & O	I can explore and comment on processes in the world around me making use of core computational thinking concepts and can organise information in a logical way.	**TCH 1-13a**
Key Vocab	Directions, Steps, Purpose, Sorting, Grouping, Create, Organise, Logical Decisions, IF, AND, OR, NOT Statements	**RESOURCES**

Early

First

Second

Phase 1	Phase 2	Phase 3
Create and **follow** a simple set of instructions (e.g. put blue pencils in the blue tray) **Describe steps** in a daily routine and **explain the purpose** (e.g. brushing teeth keeps them clean and healthy). With adult guidance, **play sorting games** using AND, OR, NOT (e.g. collect all red AND blue cars, but NOT the green cars) Follow instructions of **grouping** objects (e.g. all the red bricks, all the Lego figures)	**Create** and **follow** more complex instructions (e.g. make a smoothie) **Understand** that an IF statement is used to make a decision in response to real-life situations (e.g. IF it rains, AND I am outside, I will get wet) **Play sorting games** using IF, AND, OR, NOT (e.g. collect all the yellow AND blue LEGO bricks, IF they have 6 OR 4 dots, NOT 2) **Organise information** with simple charts (e.g. animals that live on land or water)	**Create** and **follow** instructions using concepts such as selection and repetition (e.g. Follow the instructions, repeating step four) **Sort with increasing complexity** using IF, AND, OR, NOT operators (e.g. Collect all items that are either round AND red OR square AND blue **Make logical decisions** using IF statements in real-life situations that require more than one condition to be met (AND, OR, NOT) (e.g. IF it is cold I will wear a hat AND gloves) **Create** simple sorting games using IF, AND, OR, NOT (e.g. Collect all pencils OR pens, but NOT markers) **Create** simple Venn diagrams to group and sort items (e.g. toy vehicles with two wheels or four wheels) **Organise information** e.g. research projects with headings, criteria specific to subjects

Computing Science - Skills Progression

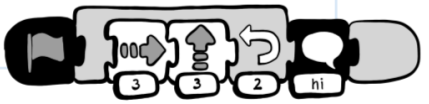
Organiser	Understanding and analysing computing technology	
E & O	I understand the instructions of a visual programming language and can predict the outcome of a program written using the language.	**TCH 1-14a**
Key Vocab	Visual Programming Language, Programming, Coding, Logical Operators	**RESOURCES**

Early

First

Second

Phase 1	Phase 2	Phase 3
With adult guidance, begin to **experiment** with the features of a visual programming environment (e.g. movement blocks, sound blocks, sprite designs in Scratch Jr) Start to **experiment** with code blocks and understand what their function on a visual programming environment With adult support, **talk about** what a program will do when it runs	Independently **navigate** the features of a visual programming environment (e.g. Scratch Jr) **Experiment** with code blocks on a visual programming environment with an understanding of their function **Predict** what will happen when a program runs (e.g. sprite moves)	**Navigate** more complex features of a visual programming language (e.g. creating an algorithm with fixed loop such as 'The sprite moves forward 5 steps') **Use** code blocks in a visual programming environment and explain their function **Explain** and **predict** what will happen when a program is run



Computing Science - Skills Progression

Organiser	Understanding and analysing computing technology	
E & O	I understand how computers process information.	**TCH 1-14b**
Key Vocab	Computing, Device, Input, Process, Output, Keyboard, Mouse, Screen, Monitor, Information.	**RESOURCES**

Phase 1	Phase 2	Phase 3
Build an awareness of the main parts of computing devices (inc. iPads/laptops) e.g. keyboard, screen etc. Begin to **explore** the idea that computers store and process information (e.g. a photo is saved to a gallery)	**Name** the main parts of computing devices e.g. keyboard, camera, screen etc. **Explore** input and output devices (e.g. microphones, cameras, iPad screens)	**Identify** the main parts of computing devices and **understand** their functions e.g. keyboard, camera, screen etc. Begin to **understand** the idea of input, process and output with regards to everyday computing devices (e.g. taking a photo, the device processing the image, and the output displaying on the screen) Begin to **understand** that information is processed and stored by computers

Early
First
Second

Computing Science - Skills Progression

Organiser	Designing, building and testing computing solutions	
E & O	I can demonstrate a range of basic problem-solving skills by building simple programs to carry out a given task, using an appropriate language.	**TCH 1-15a**
Key Vocab	Algorithm, sequence, loop, condition, debug, command, block, sprite	**RESOURCES**

Early

First

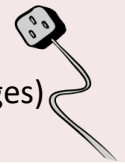
Second

Phase 1	Phase 2	Phase 3
With adult guidance, begin to **experiment** with the features of visual programming environments (e.g. Scratch Jr) With adult guidance, begin to **break down** small individual instructions (e.g. this block makes the sprite move, this block makes the sprite make a noise) Begin to **understand** the role of different blocks in a visual programming environments (e.g. Scratch Jr.)	**Simply problems** into smaller more manageable steps (e.g. in Scratch Jr, make the sprite move and then make a sound) Begin to **include repetition** commands ('fixed loops') in programs (e.g. sprite jumps four times) Run basic code and **check for errors** in a program including correct order (e.g. sprite stays still instead of moving)	**Simply more complex problems** into smaller more manageable steps (e.g. in Scratch Jr. decide whether a problem could be simplified with a repeat command) **Construct parallel algorithms** (e.g. make two sprites jump and make a sound simultaneously) **Create a program** in which a sprite performs multiple functions in a given project (e.g. move forward three steps and speak with 'repeat forever') **Explain** code solutions, **fix** any errors, and **describe** the improvements

Computing Science - Skills Progression

First

	Exemplified Learning & Resources

I can explore and comment on processes in the world around me making use of core computational thinking concepts and can organise information in a logical way
 	
TCH 1-13a
	Unplugged
	• Share instructions for activities and routines (including those with repetition e.g. “repeat step four - wash your hands”)
	• Order daily routine picture cards and explains why steps matters (e.g. “Brushing teeth keeps them clean”)
	• Write or draw step-by-step instructions for a simple task (e.g. making snacks or a smoothie)
	• In real-life scenarios, incorporate the language of IF, AND, OR, NOT logic (e.g. “IF it rains, AND you are outside, you will get wet”)
	• Play (or design) sorting games with instructional cards (e.g. “Collect pencils OR pens, but NOT markers”).
	• Organise data in charts/Venn diagrams (e.g. topic related data - animals with fur and four legs)
	• Use hula hoops to create physical Venn diagrams to group objects
	• Organise information in a mini research project using headings and categories (e.g. “Habitat/Food/Threats”)
	• Barefoot Computing Activities
	  • World Map Logic
	  • Musical Sequences Activity
	 
	Plugged
	• Record a video explaining morning routines with visuals and captions using iPad camera and apps such as Keynote, ChatterKid or iMovie
	• Create a simple sorting game in Keynote (e.g. drag and sort the blue and red circle shapes)
	• Make a digital Venn diagram using Keynote using the circle shapes and text boxes
	• Organise a simple research project digitally with headings, tables, and media (e.g. in Keynote or Pages)
	• Barefoot Computing: Accessible Coding with Scratch
	 

Exemplified Learning & Resources

First

	Exemplified Learning & Resources

I understand the instructions of a visual programming language and can predict the outcome of a program written using the language.
 	
TCH 1-14a
 	
I understand how computers process information.
 	
TCH 1-14b
	Unplugged
	 
	• Play "human robot" games - one learner gives simple instructions and the other follows
	• Predict what will happen next when a set of instructions is followed and discuss/correct errors
	• Sequence story or picture cards to plan a series of actions (e.g. steps in a story/fairytale)
	• Name the main parts of computing devices and match with their function (e.g. keyboard, screen, camera, microphone) using real objects, IWB visuals or picture cards
	• Explore input-process-output in everyday environments (e.g. "When I press a button, the toy lights up")
	• Barefoot Unplugged Activities
	  • Investigating Input Devices
	  • Investigating Output Devices
	  • Classroom Sound Monitor
	
	Plugged
	 
	• Use Scratch Jr. to create a simple algorithm (e.g. a sprite jumps and then turns 180 degrees)
	• Create a short sequence using Scratch Jr to move a sprite in a set order
	• Builds a sequence in Scratch Jr, adding fixed loops and conditions ("Repeat 3 times") or IF statements ("IF touching wall → turn")
	• Tinker with different code blocks on platforms such as Scratch Jr. and MakeCode Arcade
	• Experiment with basic code on Micro:bits and the Micro:bit app
	• Run simple directional code on the Lightbot Hour iPad app
	• Use available technology such as iPads to explore real-life examples of input-process-output (e.g. taking a photo: camera captures → iPad processes → image appears on screen)

Exemplified Learning & Resources

First

	Exemplified Learning & Resources

I can demonstrate a range of basic problem solving skills by building simple programs to carry out a given task, using an appropriate language.
 	
TCH 1-15a
	Unplugged
	• Follow and create step-by-step instructions for simple activities (e.g. drawing a shape)
	• Build a team sequence, each person representing one "step" in a process, then run the "program" to see if it works as expected
	• Barefoot Unplugged Activities
	  • Make a Game
	  • Pizza Pickle Scratch Debugging
	• CS Unplugged: Parity Magic (5 to 7) - Explore debugging
	• Explore computing science ideas through video resources through ClickView and BBC Teach
	 
	Plugged
	• Use visual programming apps such Scratch Jr. to create simple algorithms and sequences (e.g. a sprite jumps and then turns 180 degrees)
	• Direct programmable toys with instructions, then repeat with the same instructions but in a different order. This highlights the importance of order.
	• Builds a sequence in Scratch Jr., MakeCode Arcade or Micro:bit adding fixed loops and conditions ("Repeat 3 times") or IF statements ("IF touching wall → turn")
	• Debug coding solutions in apps such as Micro:bit and Scratch Jr.
	• Predict and reflect on programming design choices, discuss how using certain blocks (movement, looks, sound, loops) creates a specific outcome
	• Experiment with visual programming code to direct a Sphero using the Sphero Edu app
	• Tinker with Scratch Jr. using Barefoot Scratch lessons
	 

Benchmarks - Second

I understand the operation of a process and its outcome. I can structure related items of information. **TCH 2-13a**	I can explain core programming language concepts in appropriate technical language. **TCH 2-14a** I understand how information is stored and how key components of computing technology connect and interact through networks. **TCH 2-14b**	I can create, develop and evaluate computing solutions in response to a design challenge **TCH 2-15a**
• Compares activities consisting of a single sequence of steps with those consisting of multiple parallel steps, for example, making tomato sauce and cooking pasta to be served at the same time. • Identifies algorithms/instructions that include repeated groups of instructions a fixed number of times and/or loops until a condition is met. • Identifies when a process is not predictable because it has a random element for example, a board game which uses dice. • Structures related items of information for example, a family tree (MNU 2- 20b). • Uses a recognised set of instructions/ an algorithm to sort real worlds objects for examples, books in a library or trading cards.	• Explains the meaning of individual instructions (including variables and conditional repetition) in a visual programming language • Predicts what a complete program in a visual programming language will do when it runs, including how the properties of objects for example, position, direction and appearance change as the program runs through each instruction. • Explains and predicts how parallel activities interact • Demonstrates an understanding that all computer data is represented in binary for example, numbers, text, black and white graphics. • Describes the purpose of the processor, memory and storage and the relationship between them • Demonstrates an understanding of how networks are connected and used to communicate and share information, for example the internet.	• Creates programs in a visual programming language including variables and conditional repetition. • Identifies patterns in problem solving and reuses aspects of previous solutions appropriately for example, reuse code for a timer, score counter or controlling arrow keys. • Identifies any mismatches between the task description and the programmed solution, and indicates how to fix them.

Computing Science - Skills Progression

Organiser	Understanding the world through computational thinking	
E & O	I understand the operation of a process and its outcome. I can structure related items of information.	**TCH 2-13a**
Key Vocab	Algorithm, sequence, loop, condition, debug, sprite, input/output	**RESOURCES**

Early

First

Second

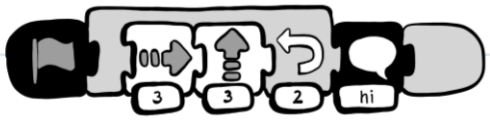
Phase 1	Phase 2	Phase 3
Identify basic tasks with one step at a time (e.g. brushing teeth) vs. tasks with two things happening at once (e.g. boiling pasta and making a sauce). **Spot** repeated patterns in simple instructions (e.g. "walk forward 3 steps, repeat 4 times"). **Understand** what is meant by predictable and random **Recognise** a random element in games (e.g. rolling a dice) **Explore** different ways of structuring information (e.g. family trees, flow charts, data tables, spreadsheets) **Use** simple instructions for sorting objects into categories (e.g. class library books by genre or trading cards by shared features)	**Explain** why tasks may need to be done in multiple parallel steps (e.g. listening to instructions, whilst tidying-up). **Recognises** loops with conditions (e.g. keep trying answers until correct in a maths puzzle) **Identify** processes with random elements and understand how this may change an outcome (e.g. flipping a coin to decide who starts a game) With adult support, **use** and **complete** diagrams (e.g. branching diagrams, Venn diagrams) to organise information. **Explore** more complex instructions (e.g. two sorting rules) for sorting objects into categories (e.g. card games suits - even numbers in the Clubs suit) 	**Organise** and **sequence** multi-step and parallel activities (e.g. cooking a simple meal with several elements ready together). **Identify** when instructions have fixed (e.g. clap your spelling words four times) or conditional loops (e.g. keep answering maths questions until you get three correct) **Identify** processes that contain either random or predictable element, understanding the impact of these. Independently **choose** and **use** different diagrams for structuring and organising information (e.g. sorting networks, branching diagrams) **Explore** and **use** more complex sorting algorithms, explaining the logical thinking used within the sorting process e.g. Bubble Sorting.

Computing Science - Skills Progression

Organiser	Understanding and analysing computing technology	
E & O	I can explain core programming language concepts in appropriate technical language	**TCH 2-14a**
Key Vocab	Input, output, storage, binary, data, hardware, software	**RESOURCES**

Early | First | Second

Phase 1	Phase 2	Phase 3
Begin to understand individual coding blocks make a sprite move, turn and repeat With adult support, **predict** what a program will do when it runs	**Use** and **understand** coding blocks with variables (e.g. scores) and conditional repetition (e.g. IF you collect a gem THEN you will receive a point) **Predict** what an ordered sequence in a visual programming language will do when it runs, with growing awareness of each step	**Use** and **understand** a range of coding blocks (e.g. variables, loops, and conditional repetition) in visual programming environments (e.g. Swift, Scratch, MakeCode) **Discuss** and **predict** the steps and outcome of a program (e.g. understanding elements such as sprite position, direction and appearance) **Predict** what will happen when two or more algorithms run at the same time (e.g. a sprite moves while sound plays)

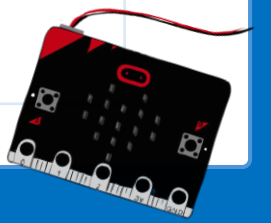


Computing Science - Skills Progression

Organiser	Understanding and analysing computing technology	
E & O	I understand how information is stored and how key components of computing technology connect and interact through networks	**TCH 2-14b**
Key Vocab	Network, internet, binary, memory, storage	**RESOURCES**

Early | First | Second

Phase 1	Phase 2	Phase 3
Understand that computers store numbers and text as data **Recognise** that networks are used to help devices to connect and share information (e.g. email, the internet)	**Understand** that computer data is represented as Binary Code (shown as 1s and 0s) **Describe** the function of a computer's processor, memory and storage (e.g. processor gives instructions, memory holds data, storage keeps data) Begin to **consider** the journey data takes and how devices communicate to share information on a network	**Demonstrate** an understanding of converting basic Binary Code into numbers and text **Identify** and **describe** the links between a computer's processor, memory and storage (e.g. opening a saved file requires the computer's memory and processor) **Demonstrate** an understanding of how devices communicate and share information via networks or online (e.g. safe internet searches, collaboration across Microsoft 365 apps)



Computing Science - Skills Progression

Organiser	Designing, building and testing computing solutions	
E & O	I can create, develop and evaluate computing solutions in response to a design challenge.	**TCH 2-15a**
Key Vocab	Debug, variable, condition, rule, solution, loops	**RESOURCES**

Early | First | Second

Phase 1	Phase 2	Phase 3
Understand that a variable is a 'box with a label' that stores information which can change (e.g. a score that goes up when you collect a gem) **Understand** the that conditional repetition is a rule that must be true for something to happen (e.g. "if you touch the wall, then the game ends"). **With guidance or tutorials:** • **Create** simple programs with code blocks including variables and conditional repetition • **Reuse** solutions for previous problems to solve a new problem (e.g. code for keeping scores can be copied) • **Identify** any errors ('bugs') in visual programming language (e.g. sprite moves too far, timer does not stop)	Using walkthroughs (e.g. MakeCode tutorials, Swift guides), **create** increasingly complex programs with variables (e.g. score counter) and conditional repetition (e.g. "repeat until key pressed") **Recognise** when a program could be enhanced from a repeated block pattern (e.g. same movement controls for different sprites) **Identify** errors in visual programming language and begin to plan solutions ('debugging') (e.g. the score isn't increasing when the sprite touches the gem, I need to add a variable block)	**Plan** and **create** more complex programs combining multiple variables, conditions, and loops to complete a design challenge (e.g. a simple game with levels and scoring) **Create** more complex programs reusing and adapting code (e.g. copying code for timers, score counters, sprite movement) **Identify** parallel algorithms interacting in a design challenge (e.g. movement, sound, scoring happening simultaneously) **Identify** and **fix** any mismatches between a task description and a planned solution (e.g. does your game contain all the required elements and do they function correctly) **Evaluates** how well a program meets a design brief and suggest improvements

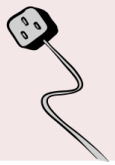
Exemplified Learning & Resources

Second

	Exemplified Learning & Resources	DLUs
I understand the operation of a process and its outcome. I can structure related items of information. **TCH 2-13a**	**Unplugged** • Create diagrams to show relationships, such as family trees, branching diagrams, or Venn diagrams • Organise and structure related information through research projects and activities • Follow and create instructions with repeated steps/conditions e.g. gymnastic sequences in PE lessons • Explore computing science ideas through video resources through ClickView and BBC Teach **Engage with CS Unplugged lessons:** • Sending a rocket to Mars- Algorithms • The Modulo Operator - Error detection • Reinforcing Numeracy through Sorting Networks - Sorting • Investigating variations with Sorting Networks - Sorting • Searching algorithms - Algorithms and searching **Engage with Barefoot Unplugged activities:** • Decomposition - Breaking down sequences • Logical Number Sequences 	• P5 - Computational Thinking - Unit 1 • P5 - Get Started Coding with Swift • P6 - Coding with Microsoft MakeCode • P6 - Learn to Code with Swift - Unit 1 • P7 - Learn to Code with Swift - Unit 2 • Micro:bit and IoT - Light Sensors • Micro:bit and IoT - Temperature Sensors

Exemplified Learning & Resources

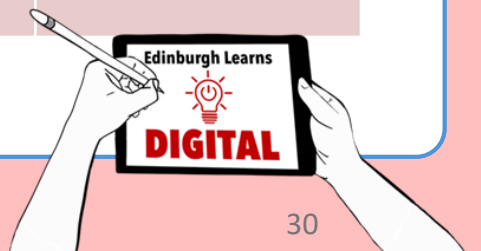
Second

	Exemplified Learning & Resources	DLUs
I understand the operation of a process and its outcome. I can structure related items of information. **TCH 2-13a**	**Plugged** • Progress through Swift Playgrounds: Get Started with Code and Learn to Code • Explore Swift Playgrounds App Design • Practice coding skillmaps available in Microsoft MakeCode Arcade • Explore Micro:bit coding activities found in Microsoft's library • Engage with Minecraft coding lessons found in MakeCode • Take part in Computing science and data handling lessons with IoT sensors from University of Edinburgh • Use Micro:bit and IoT sensors together to make Light and Temperature sensors • Engage with Computing Science - Digital Learning Units from Edinburgh Learns Digital • Explore Raspberry Pi coding projects • Explore coding lessons using Scratch • If available, utilise Spheros, Marty the Robot and other school specific coding tools. 	• P5 - Computational Thinking - Unit 1 • P5 - Get Started Coding with Swift • P6 - Coding with Microsoft MakeCode • P6 - Learn to Code with Swift - Unit 1 • P7 - Learn to Code with Swift - Unit 2 • Micro:bit and IoT - Light Sensors • Micro:bit and IoT - Temperature Sensors

Exemplified Learning & Resources

Second

	Exemplified Learning & Resources	DLUs
I can explain core programming language concepts in appropriate technical language. **TCH 2-14a** I understand how information is stored and how key components of computing technology connect and interact through networks. **TCH 2-14b**	**Unplugged** • Research how computers process and share information • Investigate the "journey of data" with real classroom examples devices labelled laptop, iPads, router, server, etc. • Investigate examples of input, process and output in your classroom. Compile these into a project. • Explore computing science ideas through video resources through ClickView and BBC Teach **Engage with Barefoot Unplugged activities:** • Crystal Flowers 1 and 2 - Exploring algorithms and loops • Network Hunt - Understanding computer networks • Patterns Unplugged - Reusing elements **Explore CS Unplugged lessons:** • How binary digits work • Reinforcing binary sequences • Binary codes for letter representation	• P6 - Coding with Microsoft MakeCode • P6 - Computational Thinking - Unit 2 • P7 - Learning Binary (Unplugged) • Micro:bit and IoT - Light Sensors • Micro:bit and IoT - Temperature Sensors

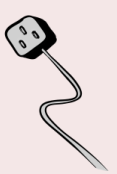


Exemplified Learning & Resources

Second

	Exemplified Learning & Resources	DLUs

I can explain core programming language concepts in appropriate technical language. **TCH 2-14a**
 	
I understand how information is stored and how key components of computing technology connect and interact through networks. **TCH 2-14b**
	Plugged
	• Understand how devices communicate through activities on Cyber Skills Live
	• Explore Barefoot's Scratch Lessons:
	  • Selection
	  • Variables
	  • Repetition
	  • Solar System simulation
	  • Viking Raid
	• Progress through Swift Playgrounds: Get Started with Code and Learn to Code
	• Explore Swift Playgrounds App Design
	• Explore coding skillmaps available in Microsoft MakeCode Arcade
	• Take part in Micro:bit coding activities found in Microsoft's library
	• Use Micro:bit and IoT sensors together to make Light and Temperature sensors
	• Engage with Minecraft coding lessons found in MakeCode
	• Take part in Computing science and data handling lessons with IoT sensors from University of Edinburgh
	• Engage with Computing Science - Digital Learning Units from Edinburgh Learns Digital
	• Explore Raspberry Pi coding projects
	• Explore coding lessons using Scratch
	• If available, use Spheros, Marty the Robot and other school specific coding tools.
		• P6 - Coding with Microsoft MakeCode
		• P6 - Computational Thinking - Unit 2
		• P7 - Learning Binary (Unplugged)
		• Micro:bit and IoT - Light Sensors
		• Micro:bit and IoT - Temperature Sensors



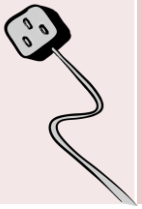
Exemplified Learning & Resources

Second

	Exemplified Learning & Resources	DLUs
I can create, develop and evaluate computing solutions in response to a design challenge. **TCH 2-15a**	**Unplugged** • Explore variables by using physical boxes or envelopes to hold changing values (e.g. keeping score in a class game, changing total when a coin is added or removed). • Model "IF… THEN…" conditions, e.g. *"If you hear a clap, take one step forward; if you hear a bell, turn around."* • Model repeated patterns or loops in movement sequences (e.g. "jump-clap-turn, repeat three times") to explore repetition and conditional repetition ("repeat until you reach the wall"). • Debug physical sequences (e.g. identifying where an instruction went wrong in a dance or LEGO build) and correct them • Evaluate instructions to explore if a sequence met its goal. How could it be improved to run more smoothly or efficiently? • Explore computing science ideas through video resources through ClickView and BBC Teach • Use CS Uplugged's Parity Magic (8 to 10) lesson to explore debugging **Explore Barefoot Unplugged activities:** • Bug in the Water Cycle - Debugging • Code Cracking - Algorithms and Debugging • Make a Game Project • Variables Unplugged	• P5 - Get Started Coding with Swift • P6 - Coding with Microsoft MakeCode • P6 - Learn to Code with Swift - Unit 1 • P7 - Learn to Code with Swift - Unit 2 • Micro:bit and IoT - Light Sensors • Micro:bit and IoT - Temperature Sensors

Exemplified Learning & Resources

Second

	Exemplified Learning & Resources	DLUs
I can create, develop and evaluate computing solutions in response to a design challenge. **TCH 2-15a**	**Plugged** • Progress through Swift Playgrounds: Get Started with Code and Learn to Code • Explore Swift Playgrounds App Design • Explore coding skillmaps available in Microsoft MakeCode Arcade • Take part in Micro:bit coding activities found in Microsoft's library • Engage with Minecraft coding lessons found in MakeCode • Take part in Computing science and data handling lessons with IoT sensors from University of Edinburgh • Use Micro:bit and IoT sensors together to make Light and Temperature sensors • Engage with Computing Science - Digital Learning Units from Edinburgh Learns Digital • Explore Raspberry Pi coding projects • Explore coding lessons using Scratch • If available, utilise Spheros, Marty the Robot and other school specific coding tools. 	• P5 - Get Started Coding with Swift • P6 - Coding with Microsoft MakeCode • P6 - Learn to Code with Swift - Unit 1 • P7 - Learn to Code with Swift - Unit 2 • Micro:bit and IoT - Light Sensors • Micro:bit and IoT - Temperature Sensors

Glossary

A - Z

Algorithm	A clear, ordered set of instructions to solve a problem or complete a task.
App	A software application designed to perform a specific function, often used on mobile devices.
Bee-Bot	A programmable floor robot used to teach directional language, sequencing, and basic coding.
Bug	An error or fault in a program that causes it to behave unexpectedly.
Button	A clickable or tappable element that triggers an action in software or on a device.
Code	A set of written instructions that a computer or robot can follow.
Command	A specific instruction given to a device or program to perform an action.
Computer	An electronic device that processes data and follows instructions to perform tasks.
Conditions	A rule in coding such as If, Then, AND, OR, NOT.
Conditional Loop	A loop that repeats code until a condition is met OR until a condition is no longer met.
Debug	The process of identifying and fixing errors in a program.
Device	A piece of technology used to interact with digital content (e.g., iPad, laptop).

Glossary

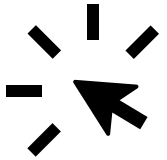
A - Z

Direction	A movement instruction such as forward, backward, left, or right.
Download	The process of transferring data from the internet to a local device.
Input	Data or signals entered into a system, such as touch, typing, or voice.
Internet	A global network that connects millions of devices and allows sharing of information.
Internet of Things (IoT)	Everyday objects connected to the internet that can send and receive data (e.g., smart thermostats, wearable tech).
iPad	A touchscreen device commonly used in classrooms for interactive learning and coding activities.
Fixed Loop	A loop that repeats for a specific number of times as instructed.
Loop	A programming structure that repeats a set of instructions until a condition is met or the conditions are no longer met.
Move	A basic instruction used to change the position of a robot or character.
Output	The result produced by a computer or device, such as sound, movement, or visual display.
parallel Algorithms	A set of coded instructions that work alongside each other at the same time.
Pattern	A repeated sequence or design, often used in coding and problem-solving.

Glossary

Pixel	The smallest unit of a digital image or display.
Program	A complete set of coded instructions that tells a computer or robot what to do.
Robot	A machine that can be programmed to carry out tasks automatically.
Scratch Jr	A visual programming app designed for young learners to create interactive stories and games.
Sequence	The specific order in which instructions are executed in a program.
Sphero	A spherical robot that can be programmed to move, light up, and interact with its environment.
Sprite	A character or object in a coding environment that can be programmed to act.
Step	A single action or instruction within a sequence or algorithm.
Swift	A programming language used to develop apps, particularly for Apple devices.
Technology	Tools and systems designed to solve problems or enhance learning and communication.
Upload	The process of sending data from a local device to the internet or another system.
Variable	A 'box' with a label that stores information which can change i.e. x 2.

Further Resources



Edinburgh Learns Digital

Digitips

Digital Learning Units

Digital Lending Library

Staff & Learner Skill Courses

Edinburgh Learns Digital SharePoint

Digital CLPL Opportunities

Digital Strategy Document

Highly Effective Digital Practice

ClickView Topic - Computing Science

[Coding and Programming | ClickView]

[Hardware and Software | ClickView]

[National Coding Week | ClickView]

[Networks and the Internet | ClickView]