Key Stage 1 – *Discovering Programming* – Theme Guide

Children name the main external parts of a computer and explore how they work together. They explore programmable devices relating their understanding of inputs and outputs to digital systems. They use unplugged approaches and simple onscreen and physical devices to develop understanding of algorithms and programming. They develop their own skills in open programming time.

Learning objectives for the term
To begin to understand what a computer is and how it operates. Identify the main "parts" (screen, mouse, touchpad, touch screen, keyboard, base unit etc.)
To understand that we use many programmable and automated devices at school, home and in the wider world.
To understand that an algorithm is a set of precise instructions or rules to carry out a specific task or solve a problem. *Suggest suitable tasks for new algorithms.*
To understand we use logical reasoning help create algorithms.
To understand computers use programs written in programming languages and that there are many such languages. *Create their own language for a task.*
To understand digital devices are controlled using programs written in specific programming languages.
To understand precision and sequence are key to programming.
To know the repeat command can make programs more efficient.
To know collaborative exploration can support efficient programming.
On-going Learning Objectives
To understand the need to build up a program, step-by-step.
To use technologies safely and appropriately.
To talk about the choices they made. Revisit and refine their work.
To develop awareness of environmental issues related to technology.
To log on to the school system and save, locate and edit work.
To know to tell a trusted adult if anything they access or use makes them feel uncomfortable or worried.

Vocabulary – see Glossary for definitions (for terms in blue)	
• *unplugged* • hardware: visualiser, webcam, microscope, smartphone etc. • computer parts: keyboard, monitor, *base unit,* speakers etc. • *logical reasoning* • *algorithm,* programmable device	• *program,* • *debug*, *refine, predict* • *repeat (repetition,)* • precision, • sequence, • *decompose*

Possible resources for this theme (further resources are suggested with the explanatory notes below. Note that these are examples and not formal recommendations.)	
Online programming tools • Rapid Router (earlier units) https://www.codeforlife.education/rapidrouter/ • UK.Code.org https://uk.code.org/learn • 2Code (as part of 2Simple Purple Mash) • Espresso Coding • J2Code (as part of J2E) • JIT5 Turtle (as part of J2E)	**Apps** • Daisy the Dinosaur (iPad®) • Kodable (iPad, Android, Web) • Beebot (iPad) • Bluebot (iPad, Android) • Fix the Factory (iPad, Android) • ScratchJR (iPad, Android, Chrome) • A.L.E.X. (iPad, Android pad)
Physical Programmable Robots • Beebot • BlueBot • Contructa-Bot • Roamer Too	**Other** • MSW Logo (free Windows software)

Please note that with any online platform it is essential that you review the privacy policy and terms and conditions of the service. The school is responsible for the protection of data it holds and compliance with curent data protection legislation. Always assess both the data protection and safety of the service you are considering using, and ensure any necessary permissions are in place before using with pupils.

Free *Barefoot Computing* activities to support this theme.	
• **Crazy Character Algorithms Activity**. Link An unplugged activity where children create instructions for drawing a crazy character, and learn about algorithms. • **Spelling Rules Activity**. Link An unplugged activity where children create instructions for sounding phonemes, and learn about algorithms. Link to English. • **Sharing Sweets Algorithms Activity**. Link An unplugged activity where children create instructions for sharing sweets, and learn about algorithms. Link to English. • **Beebots Tinkering Activity**. Link Children explore the Beebot floor robot and learn how to program them. • **ScratchJR (app) Tinkering Activity**. Link Children explore the free ScratchJR app and learn how to use it to create simple programs. • **Beebots Basics Activity**. Link Children design and solve challenges using Beebots or similar programmable floor robots. • **Beebots 1,2,3 Programming Activity.** Link Children draw the shapes of numerals using programmable floor robots. • **Decomposition Unplugged Activity**. Link Children use hand-clapping sequences, breaking them down into individual steps (decomposition.) • **ScratchJR (app) Knock-knock Joke Activity.** Link Children program an animated knock-knock joke using the ScratchJR app.	*Barefoot Computing* provides freely available resources to support teachers in delivering the computer science aspects of the 2014 Primary National Curriculum for Computing. Free registration with the *Barefoot Computing* website is required to view and download these resources. To register, visit: http://barefootcas.org.uk References to these resources and the accompanying links are provided with permission from *Barefoot Computing*.

Primary Computing Scheme online materials that are referenced in this guide can be accessed from: http://www.hertsforlearning.co.uk/user/login

You will need to be logged into your school account and have a current subscription to the Primary Computing Scheme to gain access. The materials can be accessed from the *My Resources* link at the top/right of the screen, once you are logged in.

Key learning objectives (some objectives might be used for more than one lesson)
To begin to understand what a computer is and how it operates. Identify the main "parts" (screen, mouse, touchpad, touch screen, keyboard, base unit etc.)
• Compare different computers (for example a tablet, smart phone, games console, desktop and laptop computer). • On a basic level, computers work through ***Input → Processing → Output*** • Discuss input and output. Input is the information which goes into a computer, through an input device, which is processed by the computer. Output is the information that comes out of a computer, through an output device. For example, and keyboard or microphone are input devices. A screen or speakers are output devices (though these days a screen may be a touch-screen, in which case it is also an input device.) • Role-play the parts working together to carry out a task. • Some devices and peripherals can have more than one correct name, for example it would be acceptable to call a computer screen a 'screen', or a 'monitor'. Agree with your team what terms you will use and use these consistently.

- Talk about whether the 'part' may be used for an input (e.g. mouse, keyboard, microphone) or output (e.g. screen, speakers.)
- See the accompanying Smart® Notebook® 15 file [*Computer Devices Labelling Activity*]

To understand that we use many programmable and automated devices at school, home and in the wider world.

- Talk about the different programmable / automated devices we have at home and at school. E.g. at home we have lots of devices especially in the kitchen (dishwasher, microwave etc.) At school, look around but also beyond the classroom. E.g. you may have things like electronic car park barriers that open automatically when a car is exiting the car park.
- Select two simple examples to investigate.
- In simple terms of input and output, think about how these things work. Think about how they are controlled and the instructions we might give them. How are these instructions given? (In a special 'language' that the device can understand.) Do these devices understand our spoken language?
- Talk about the fact that these devices do not understand our language, and have to follow 'programs' which are written in special languages for computers.
- See the accompanying Smart® Notebook® 15 file [*Input and Output Devices Sorting Activity*]

To understand that an algorithm is a set of precise instructions or rules to carry out a specific task or solve a problem. *Suggest suitable tasks for new algorithms.*

- Use every day examples of instructions that children follow, to introduce algorithms.
- As a class, create algorithms for some everyday tasks, (for example getting ready for school in the morning, cleaning teeth, lining up for assembly, solving a problem in maths etc.)
- Be clear that before creating an algorithm we need an objective to achieve or problem for it to solve – the purpose for the algorithm (e.g. as above.)
- See the accompanying sheet: [*10 ideas for teaching algorithms*]
- See the accompanying sheet: [*My amazing algorithm*]
- See the following *Barefoot Computing* links above:
    - Crazy Characters Algorithm Activity
    - Spelling Rules Activity
    - Sharing Sweets Algorithms Activity

To understand that we use logical reasoning to break tasks down into smaller steps (decompose) to help us create algorithms.

- Have different pupils working on different algorithms, not the whole class working on the same one.
- Pupils write their own algorithms by breaking a specified task down into steps. This is called decomposition.
- For each task, decompose it, in order to create the algorithm. Test, debug and refine each step, noting that there could be different solutions.
- Pupils can test their algorithms on each other to see how effective they are.
- They can then improve it by fixing any errors (*debugging,*) making it more efficient and as short as effectively possible (*refining*.) They can then test them again.
- Make sure you use the language throughout; *algorithm, debugging, refining, testing, decomposition.*
- See the following *Barefoot Computing* link above:
    - Decomposition Unplugged Activity.

To understand that computers use programs written in specific programming languages and that there are many such languages. *Create their own language for a task.*

- Just like people can speak different languages, computers also speak different languages. We cannot speak to computers in our own language because it doesn't understand, so we use a special language the computer does understand, called a *program*.
- As a class create a simple unplugged programming language using single words or symbols (clap, jump, ≠, ♦ etc.).
- Before creating your pretend, unplugged 'programming language' think about what exact commands the device will need. For example, for a robot (role-played by children or the teacher) to sharpen a pencil, what commands will it need? Make cards with these commands, which the children can arrange to create 'programs' for the robot. Remember, we can only use these commands in our program because if we use something that is not part of the language, the robot won't understand.
- Talk about how we use *algorithms* to plan the process using every day language, but we have to change it into a *program* before the device will understand it.

To understand that digital devices are controlled using programs written in specific programming languages.

- Explore simple physical programmable devices and the languages in which they are programmed.
- Think about which commands your programmable device understands (usually directions at this stage.)
- Investigate the commands and write, test, debug and refine programs.
- Predict the outcomes of programs. Test their predictions.
- Can we ask it to do other things? Why not?
- Avoid just letting the pupils simply play with the devices (though this is OK to start with, for them to understand how they are operated.)
- Instead, use maps or grids and have the children plan the program (perhaps a series of arrows) that will be needed for the device to get to a certain destination. For example, use squared paper that the pupils can draw arrows on, in sequence.
- See the accompanying sheet: [*Show me the way activity*]
- See the following *Barefoot Computing* links above:
  - Beebots Tinkering Activity
  - Beebots Basics Activity
  - Beebots 1,2,3 Programming Activity

To understand that precision and sequence are key to programming.

- As above, have the pupils write the commands on paper, using an agreed format, to program the device to get to a specific destination on a map or grid.
- Test, debug and refine the program.
- Swap papers and pupils predict where the device will go (*predicting*.)
- Pupils can write the program on paper with one deliberate mistake. Can their friends find the mistake (*debugging*.)
- Try the programs out on the actual devices and/or onscreen object to test their efficiency. There are free apps and online activities for programming simple directional devices.
- What happens if we change the order of commands? Does the program still achieve its objective?
- You could consider introducing simple Logo programming commands at this stage. (See the accompanying sheet: *MSW Logo, the basics,* and the video guides, if you have not used Logo before.)
- See the following *Barefoot Computing* links above:
  - ScratchJR Tinkering Activity
  - ScratchJR Knock-knock Joke Activity

To know that the repeat command can make programs more efficient.

- If we need to use a command in an algorithm or program more than once in sequence, we can use a repeat command rather than using the same command many times. E.g. to sharpen a pencil we turn it repeatedly. Rather than write '*turn pencil*' ten times we could write it once and then write '*repeat 10*' or suchlike. This is more efficient.
- Use the repeat command (or repeat block if using block programming) in onscreen programming environments, for example in the ScratchJR app or in MSW Logo.
- See the video guide: [*Getting Started with Logo in Primary Schools – Part 1*]
- See the downloadable sheet: [*MSW Logo, The Basics*]
- See the video guide: [*A programming activity in ScratchJR (iPad)*]
- Think about where/how repeat commands might be used in the world around us. Relate to real world applications such as lifts, sirens etc.
- Consider showing video clips of, for example, robots building cars in factories or other automated system where tasks are repeated.
- See the following *Barefoot Computing* links above:
  - ScratchJR Tinkering Activity
  - ScratchJR Knock-knock Joke Activity

To know that collaborative exploration can support efficient programming.

- Provide open programming time during which the children work with learning partners to explore and tinker, developing their own ideas and creating their own programs.
- Much can be learned through exploring and tinkering with programming environments.
- Encourage children to explore different environments (e.g. a programming app on an iPad®/tablet and a PC based activity.)
- Have the children read and comment on programs created by others.

On-Going Learning Objectives
To understand the need to build up a program, step-by-step, testing and debugging each individual part.
To understand the need to use all technologies safely and appropriately.
To talk about the choices they made. Revisit and refine their work in the light of comments and suggestions from peers.
To develop awareness of environmental issues related to the use of technologies (energy use, responsible disposal of batteries, cartridges, old technologies etc.)
To log on to the school system and save, locate and edit work.
To know to tell a trusted adult if anything they access or use makes them feel uncomfortable or worried.

Suggested independent task – *any open-ended activity (2-3 sessions) enabling the children to demonstrate their computing capability around the knowledge and understanding provided in the term*

- Create a simple algorithm to achieve a specific objective or target.
- Write a program to instruct a physical and/or onscreen device to achieve an objective or target (This could link to their algorithm produced above.)
- Test, debug and refine the program and note how it has been improved and/or developed for accuracy or efficiency.
- Predict and test the outcome of a program written by a peer. Suggest improvements to the program.

Other considerations:

Does the task provide for children to work at different levels?
Is there support available for children to select if they wish?
Are there opportunities for the children to review and develop their work?
Is there an opportunity for the children to evaluate the finished task?

Depending on the resources available to you, the algorithm and program above could be linked or separate. Give children the opportunity to test the program on someone else, observe its efficiency and debug/refine it as necessary, before submitting a final version they are happy with.

Have the pupils create a program (e.g. directional commands to move a programmable robot to a specific destination on a map or grid) without stating what the objective is, so that their peers can predict what this is, and suggest improvements as necessary.

See the sample independent task in the assessment section, for this learning theme.