## Foundation-2

## Programming and algorithms

|  |  |  |
| --- | --- | --- |
| Aspect of DT | Digital Technologies focus | Suggested DT Hub resources |
| Explore a simple algorithm | Explore Simple algorithm (step-by-step instructions) no branching. This could include giving and following instructions, ordering a set of instructions, writing (with support) a set of instructions. (Are the instructions clear and explicit which result in the expected outcome?) | Happy Maps<http://bit.ly/2pc8kgP>  CS Unplugged Programming languages  <http://bit.ly/2p3c4Gd>  Lesson 1: The Need for Programming Languages  <http://bit.ly/2qh1GKk> |
| Ways to improve an algorithm | Look at common algorithms (setting the table, cleaning your room) and consider how they could be improved. Relate to digital algorithms. | <https://tellagami.com/gami/5Q4X9R/> |
| Programming a robot or introductory software | Program a push button robotic device to move in an intended manner or apply directional language commands in an online environment. Continually review programming to complete a task in the shortest number of steps. | Bee-Bot lesson ideas  <http://bit.ly/2p3cpc0>  Bee-Bot Emulator: online games:  <http://bit.ly/2pcdSYv>  Bee-Bot app:  <http://apple.co/2pflday> |
| Explore programming fundamentals | Work through challenges to develop the fundamentals of programming, moving forwards, backwards, turn left, turn right, start, stop, go, ok, run program. Use supported challenges. | Code studio course 1  <http://bit.ly/2pG6w3b>  Blockly Games (Puzzle and Maze)  <http://bit.ly/2qKY4Ns>  Apps/Web-based course  The foos  <http://bit.ly/2qKS0Vx> |
| Create your own simple animation | Explore creating your own simple animation using an app suitable for Early Years students. The focus is on sequencing a series of commands using programming blocks to create an intended outcome. Look for ways to improve program to achieve shortest number of steps. | Apps  Scratch Jr  <http://bit.ly/2qKYC5Y>    Daisy the dinosaur.  <http://bit.ly/2q5rgCy> |

## Years 3-4

## Programming and algorithms

|  |  |  |
| --- | --- | --- |
| Aspect of DT | Digital Technologies focus | Suggested DT Hub resources |
| Explore a sequence of instructions (algorithms) | Write, order, follow and revise a series of instructions that explain the steps to complete a task. Focus on being explicit in their instructions. Follow instructions explicitly and look at steps where instructions are ambiguous or those that assume knowledge that is not provided in instructions eg draw a square. | Introducing algorithms  <http://bit.ly/2q7Hatw>  Lesson 1: The Need for Programming Languages  <http://bit.ly/2qh1GKk> |
| Define simple problems (using abstraction) | Decompose a problem and use abstraction (focus on essential information), when solving simple problems. Aim to reduce complexity and enable efficient design solutions. | Thinking Myself  <http://bit.ly/2pmg3xQ>  Abstraction  <http://bit.ly/2rgwM2L> |
| Use design tools when developing a digital solution | Create storyboards or flowcharts to record relationships or instructions when designing a digital solution to a problem. Consider ways to show decision making (branching). | Gliffy (design tool) requires free account set up  <https://www.gliffy.com/>  Storyboard template  <http://bit.ly/1JmOx1y> |
| Learn the fundamentals of visual programming | Provide self-paced learning opportunities to learn basic visual programming skills. Develop their understanding of decision making (branching) use if /then to do a specified action and if/else to do one action or the other. | Blockly maze  <http://bit.ly/2r2jpCu>  Lightbot  <http://lightbot.com/flash.html> |
| Implement a digital solution to a problem using visual programming | Use a programming language such as Scratch to implement a digital solution that incorporates user input and decision making. Programming may be integrated with a robotic device. | Scratch  <http://bit.ly/2qVoX54>  Scratch Resources for Schools and Students  <http://bit.ly/2rfThE2>  AN INTRODUCTORY  COMPUTING CURRICULUM  USING SCRATCH <http://bit.ly/2rfThE2>  Snap  <http://bit.ly/2q7IxZg>  Robo Bowling Dash and dot challenge example  <http://bit.ly/2q5fnvy>  Apps  Tickle, Hopscotch, Blockly,  OzoBlockly  <http://ozoblockly.com/>  Robotic devices such as Ollie, Sphero, Ozo bots |

## Years 5-6

## Programming and algorithms

|  |  |  |
| --- | --- | --- |
| Aspect of DT | Digital Technologies focus | Suggested DT Hub resources |
| Define simple problems (using abstraction) | Decompose a problem and use abstraction (focus on essential information and decide on what we can ignore), when solving simple problems. Use models to explain real-world systems. | Thinking Myself  <http://bit.ly/2pmg3xQ>  Abstraction  <http://bit.ly/2rgwM2L>  BBC Computational thinking  <http://bit.ly/2pE4ayu> |
| Build on the fundamentals of visual programming | Provide opportunities to build on their basic visual programming skills. Continue to refine use of branching. Emphasise how the use of repeating blocks of code is much more efficient than writing each command separately. | Lesson 5: Creating Functions  <http://bit.ly/2r6W1qR>  Lesson 6: Functions and Top-Down Design  <http://bit.ly/2qGZH2W>  Worksheet - Top-Down Design  <http://bit.ly/2qif3JR>  Video: Hour of Code: Saloni on the If/Else block  <http://ab.co/2rgrR1J>  GROK Learning  <http://bit.ly/2pOtov2>  For iOS  Swift playground Course 1 and 2 (Teacher guide)  <http://bit.ly/2qkUZ87> |
| Use design tools when developing a digital solution | Create storyboards or flowcharts to record relationships or instructions when designing a digital solution to a problem. Consider ways to show decision making (branching) and iteration (repetition) as well as inputs and outputs, for example ask the user for information that results in an onscreen action. Validate and refine the design. | Gliffy (design tool) requires free account set up  <https://www.gliffy.com/>  Storyboard Creator  http://bit.ly/1pEySGn  Making maths quizzes 1:  Plan and test our programs  <http://bit.ly/2r6rEgL> |
| Implement a digital solution | Create a range of digital solutions, such as games or quizzes and interactive stories and animations using relevant tools or visual programming language environments. The code written may not work as it’s intended, encourage debugging by working through their algorithm and looking for mistakes in the code. Take into consideration user interface design and user input. Evaluate the solution. | Making maths quizzes 2:  Plan and test our programs  <http://bit.ly/2qyCPCD>  PUT THE USER  IN USER INTERFACE  <http://bit.ly/2r6lrBE>  Programming software/apps: Scratch, Snap, Tynker, Hopscotch etc |
| Implement a digital solution that involves the control of a robotic device. | Program a robot to operate independently, incorporate loops and decision-making instructions to achieve the desired outcome. | RoboMind software <http://bit.ly/2qkVHCe>  Sphero education  <http://bit.ly/1QNgLcD>  Ozobot Lesson Library <http://bit.ly/2qidxr7>  Ozoblockly games  <http://games.ozoblockly.com/>  Tickle app: controls different robots: Sphero, Ollie etc  <http://bit.ly/2ql6rAJ>  Also search for: Lego WeDo, Lego Mindstorms |

# Further reading

A guide to programming languages for coding in class: <http://bit.ly/2qHeKK6>

CS Unplugged: Field guide: Programming Languages: <http://bit.ly/2rfKvpJ>

DT-Hub TOPIC: VISUAL PROGRAMMING <http://bit.ly/2rgUj3M>