Digital Technologies – Years F - 2 _ Sequences

	Strand		Knowledge and understanding				Strand: Processes and production skills							
			Digital systems		Representation of data		Collecting, managing and analysing data		*Creating digital solutions by:*					
									Investigating and defining		Evaluating		Collaborating and managing	
	Content Description		Recognise and explore digital systems (hardware and software components) for a purpose (ACTDIK001)		Recognise and explore patterns in data and represent data as pictures, symbols and diagrams (ACTDIK002)		Collect, explore and sort data, and use digital systems to present the data creatively (ACTDIP003)		Follow, describe and represent a sequence of steps and decisions (algorithms) needed to solve simple problems (ACTDIP004)		Explore how people safely use common information systems to meet information, communication and recreation needs (ACTDIP005)		Create and organise ideas and information using information systems independently and with others, and share these with known people in safe online environments (ACTDIP006)	
Sequence of Lessons / Unit	**Approx. time rq'd**	**Year A or B**	CD	Achievement standard #	CD	Achievement standard #	CD	Achievement standard #	CD	Achievement standard #	CD	Achievement standard #	CD	Achievement standard #
Pre-programming		2	☑	1	☐		☐		☑	3	☐		☐	

Years F-2 Achievement Standard	**Years 3 and 4 Achievement Standard**
By the end of Year 2 • Students identify how common digital systems (hardware and software) are used to meet specific purposes. (1) • They use digital systems to represent simple patterns in data in different ways. (2) • Students design solutions to simple problems using a sequence of steps and decisions. (3) • They collect familiar data and display them to convey meaning. (4) • They create and organise ideas and information using information systems, and share information in safe online environments. (5)	By the end of Year 4 • Students describe how a range of digital systems (hardware and software) and their peripheral devices can be used for different purposes. (1) • They explain how the same data sets can be represented in different ways. (2) • Students define simple problems, design and implement digital solutions using algorithms that involve decision-making and user input. (3) • They explain how the solutions meet their purposes. (4) • They collect and manipulate different data when creating information and digital solutions. (5) • They safely use and manage information systems for identified needs using agreed protocols and describe how information systems are used. (6)

Pre-programming
At the F–2 level, where learning at the pre-programming stage is the expectation, there is no requirement to learn a particular programming language. However, students do learn some basic computational skills such as working out steps and decisions required to solve simple problems. For example, they can instruct a robotic toy to move in a certain direction. The focus at this level is on designing a sequence of steps. Some students may be ready to learn to use a simple visual programming language specifically designed for young children. An app that enables the user to drag and drop programming blocks can be used to create some simple animations.

Flow of activities				
Short text	**Revisit algorithms** Identify the steps involved in completing a task and create instructions for someone to follow.	**Create algorithms** Incorporate the design of an algorithm for a meaningful purpose.	**Instructing a robotic device** Provide meaningful ways to incorporate the programming of robotic devices.	**Algorithmic thinking** Use a simple visual programming language specifically designed for young children.
Questions to guide exploration	*Can you describe and represent steps to complete a task?*	*How can I design an algorithm for a particular task?*	*How can I program a robot?*	*How can you program a series of steps using programming blocks?*
	Investigating and defining (ACTDIP004)	Investigating and defining (ACTDIP004)	Investigating and defining (ACTDIP004) Digital systems (ACTDIP001)	Investigating and defining (ACTDIP004)
What's this about?	Students continue to refine their understanding of algorithms. They should be able to describe, follow and represent algorithms. Typically, algorithms can be represented in text and graphic forms, such as photographs, 'flowcharts' and instructional cards.	Students consider the most suitable algorithmic representation for a specific task, such as directions to move an object from one position to another; a sequence of dance steps; or a basketball sequence to move to the goal. Representation options could include ordered photographs, a marked floor grid with directions and steps, a sticky note sequence, or a PowerPoint presentation.	A robot needs instructions to know what to do. Students experienced in using Bee-Bots will know that the programming is input by push buttons. An Ozobot robot has a visual sensor to gather information about its surroundings. An Ozobot can follow visual commands, which are made up of a series of colours.	While there is no requirement to learn a particular programming language at F–2, some students will be ready to learn to use a simple visual programming language specifically designed for young children.
The focus of the learning (in simple terms)	Revisit algorithms by looking at familiar activities or tasks. Identify the steps involved and create instructions for someone to follow. Pair up students and ask each student to read aloud the instructions created by their partner. Students see and verbalise algorithmic representations. This is an effective pedagogy for developing students' understanding of algorithmic thinking. If there are errors in a representation, students can consider together how to change the sequence or instructions so the task can be completed as intended.	Students design an algorithm for a meaningful purpose. For example, the class could imagine that a new student has joined the class. They could create algorithms that show the new student how to find their way to and from locations in the school. Photographs of each location can be incorporated. Look for opportunities for students to work with a programming challenge. Have older students design algorithms and represent them for their audience. For example, as a cross-age task, students can design and build a robot and design a way to command the robot to complete a series of tasks. Where appropriate, students represent an algorithm that can be carried out by a robotic device. Note: Students are not required to use a programming language at this level, but many students are able to issue instructions through a controller.	Explore how Bee-Bot robots work. Using the buttons students can identify the simple user interface and how it works. The Bee-Bots themselves represent hardware that the students are exploring. You can provide meaningful ways to integrate various subject areas as the students program the Bee-Bot. Students can create or select visual commands to instruct an Ozobot robot to complete a task. Discuss this robot as a piece of hardware and the fact that it gathers data through sensors as its input. Relate this to the output of relevant movement or action.	Provide access to an app that uses visual programming blocks as a way to animate an onscreen character. Students construct a sequence of steps (an algorithm) by arranging various blocks in a logical sequence. In Scratch J,r for example, an algorithm will have a start (eg press the green flag) and an end point (red end block). These blocks execute one after the other. Students can check that their code executes in the correct order by following the code and checking the visual animation.
Supporting resources and tools and purpose/context for use.	Introducing algorithms This lesson has a range of activities to introduce or extend students' understanding of algorithms. Thinking myself This simple problem-solving game introduces basic coding language and skills to early learners. Students progress through four categories: decompose, patterns, abstract and algorithms, and solve a problem in each. Each category contains a step-by-step tutorial followed by a simple task.	Cross-age making a robot In this cross-age project, students collaborate on a code for an unplugged robot. They design, test and modify the robot and create instruction manuals.	What's the buzz? Students create a map for a bee to follow. The bee pathway can be followed by a Bee-Bot. Three little pigs Retell the story of the three little pigs using a light sensing robot such as Ozobot.	Scratch Jr Scratch Jr is an introductory programming language that enables young students (aged 5–7) to create their own interactive stories and games. Daisy the dinosaur This is a free iPad app featuring a dinosaur that can be programmed to complete a series of simple tasks including instructions to produce a desired outcome.
Assessment	**Suggested approaches** There does not need to be any formal assessment – just formative to address any misunderstandings before they are applied to a specific task.	**Suggested approaches** • Demonstrate two steps or instructions in the algorithm.	**Suggested approaches** • Checklist for how the robot moved as per instructions in the algorithm.	**Suggested approaches** • Present or demonstrate branching or user input in a digital solution.

	Achievement standard **Design** solutions to simple problems using a sequence of steps and decisions.	• Verbalise some instructions and compare them to the stated algorithm. (This shows understanding by the reader and any errors, if appropriate, by the creator of the algorithm.) **Achievement standard** **Design** solutions to simple problems using a sequence of steps and decisions.	• Demonstrate two steps of a robotic solution. • Label a diagram of a robotic device. **Achievement standard** **Design** solutions to simple problems using a sequence of steps and decisions. **Identify** how common digital systems (hardware and software) are used to meet specific purposes.	• Scratch Jr: Assessment: This resource assesses students' understanding of the programming blocks. **Achievement standard** **Design** solutions to simple problems using a sequence of steps and decisions.