# Digital Technologies – 3 and 4_Digital solutions Digital solutions

| Strand | | | Knowledge and understanding | | | | Processes and production skills | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | Digital systems | | Representation of data | | Collecting, managing and analysing data | | *Creating digital solutions by:* | | | | | | | |
| | | | | | | | | | Investigating and defining | | Producing and implementing | | Evaluating | | Collaborating and managing | |
| **Content Description** | | | Identify and explore a range of digital systems with peripheral devices for different purposes, and transmit different types of data (ACTDIK007 ) | | Recognise different types of data and explore how the same data can be represented in different ways (ACTDIK008 ) | | Collect, access and present different types of data using simple software to create information and solve problems (ACTDIP009) | | Define simple problems, and describe and follow a sequence of steps and decisions (algorithms) needed to solve them (ACTDIP010) | | Implement simple digital solutions as visual programs with algorithms involving branching (decisions) and user input (ACTDIP011) | | Explain how student solutions and existing information systems meet common personal, school or community needs (ACTDIP012) | | Plan, create and communicate ideas and information independently and with others, applying agreed ethical and social protocols (ACTDIP013) | |
| **Sequence of Lessons / Unit** | **Approx. time rq'd** | **Year** | CD | Achievement standard # | CD | Achievement standard # | CD | Achievement standard # | CD | Achievement standard # | CD | Achievement standard # | CD | Achievement standard # | CD | Achievement standard # |
| Programing project | 12 | 4 | ☐ | | ☐ | | ☐ | | ☑ | 3 | ☑ | 3 | ☑ | 4 | ☐ | |

| **Years F-2 Achievement Standard** | **Years 3 and 4 Achievement Standard** | **Years 5 and 6 Achievement Standard** |
|---|---|---|
| By the end of Year 2<br>• Students identify how common digital systems (hardware and software) are used to meet specific purposes. (1)<br>• They use digital systems to represent simple patterns in data in different ways. (2)<br>• Students design solutions to simple problems using a sequence of steps and decisions. (3)<br>• They collect familiar data and display them to convey meaning. (4)<br>• They create and organise ideas and information using information systems, and share information in safe online environments. (5) | By the end of Year 4<br>• Students describe how a range of digital systems (hardware and software) and their peripheral devices can be used for different purposes. (1)<br>• They explain how the same data sets can be represented in different ways. (2)<br>• Students define simple problems, design and implement digital solutions using algorithms that involve decision-making and user input. (3)<br>• They explain how the solutions meet their purposes. (4)<br>• They collect and manipulate different data when creating information and digital solutions. (5)<br>• They safely use and manage information systems for identified needs using agreed protocols and describe how information systems are used. (6) | By the end of Year 6:<br>• Students explain the fundamentals of digital system components (hardware, software and networks) and how digital systems are connected to form networks. (1)<br>• They explain how digital systems use whole numbers as a basis for representing a variety of data types. (2)<br>• Students define problems in terms of data and functional requirements and design solutions by developing algorithms to address the problems. (3)<br>• They incorporate decision-making, repetition and user interface design into their designs and implement their digital solutions, including a visual program. (4)<br>• They explain how information systems and their solutions meet needs and consider sustainability. (5)<br>• Students manage the creation and communication of ideas and information in collaborative digital projects using validated data and agreed protocols. (6) |

**Topic: Digital solutions**

**Units**

**Year 3** **Year 4**

| | |
|---|---|
| **Intro to programming** 8 hours<br>Follow the problem solving process to design and create a digital solution. | **Programing project** 12 hours<br>Develop an understanding of computer programming as a series of instructions |

**Programing project**
Year 4 TOPIC Digital solutions Time: 12 HOURS

Students should develop an understanding of computer programming as a series of instructions that can change depending on different user inputs or conditions. The focus is on how digital systems follow instructional pathways and how these can be described using flow charts or through the use of visual programming languages. These pathways can be hand drawn, displayed graphically, using cards or manipulated digitally using block-based programming languages.

| Flow of activities | | | | |
|---|---|---|---|---|
| | **Define a problem**<br>Define a problem drawing on computational thinking and draw some conclusions about its features or needs. | **Create a storyboard**<br>Create a storyboard or flow chart to record relationships between the content and processes. | **Implement a solution**<br>Use a visual programming language as part of the digital solution. | **Evaluate it**<br>Evaluate how well the solution met the desired outcome. |
| Questions to guide exploration | *What problem needs solving?* | *How do I plan a solution?* | *How do I create a quiz/story/game?* | *Was my digital solution successful?* |
| AC alignment | Investigating and defining (ACTDIP010) | Investigating and defining (ACTDIP010) | Producing and implementing (ACTDIP011) | Evaluating (ACTDIP012) |
| What's this about? | Students need to know what will solve the problem before they can design and create a solution.<br><br>Problem definition involves analysis, an understanding of each element of a situation/problem and how these elements are connected. Problem definition answers the 'what' questions – what is the cause of the problem and what would solve the problem?<br><br>Students define the problem so they can consider the steps involved in coming up with a relevant digital solution. | Computers operate by following a list of instructions, called a program, which has been written to carry out a particular task. Programs are written in languages that have been designed, with a limited set of instructions, to tell computers what to do. Some languages are more suitable for some purposes than others.<br>A fundamental skill required for programming is the creation of step-by-step instructions designed to solve a problem or complete a set task.<br><br>Planning before programming is an essential step that may involve creating a flow chart or storyboard. This step draws on students' computational skills and enables them to consider the sequence of the program and where branching is likely to occur.<br><br>A storyboard or flow chart depicting a choice of events within a plot is a fun way of visualising algorithms and can be an effective way to teach the concept of 'branching'. Branching involves making a decision between one of two or more actions, depending on sets of conditions and the data provided. | Using a programming language allows you to create an individual solution to a problem. At this level, students need to create a solution that lets the user provide input, such as an answer to a question, or state a direction as well as including choices or decisions (branching).<br><br>An interactive story, game or quiz that provides the user with a choice of paths or options is a fun way of visualising algorithms and can be an effective way to teach the concept of 'branching'.<br><br>Robotic devices also provide relevant learning opportunities that incorporate visual programming for a digital solution.<br><br>Branching allows for decisions to be made and allows actions to be changed based on their input. This input could be:<br>• user-input; for example, selecting an onscreen value or button, typing in an answer<br>• sensed from the immediate environment; for example, collected via a sensor on a robotic | When evaluating students consider how well the solution met the desired outcome and in particular the problem defined at the beginning of the process.<br><br>At this level, students need to judge not only how well their solutions meet the identified needs, but also how existing solutions meet personal, school or community needs.<br><br>In this unit students will evaluate their solution as well as an existing solution for a similar purpose. Where appropriate they will also determine how well an existing digital solution meets a community need. |

| | | | | |
|---|---|---|---|---|
| | | | device sensing a set speed is reached and being programmed to slow down. In the case of Ozobots, different colours when sensed will change the speed. | |
| The focus of the learning (in simple terms) | In this unit students will define a problem that already exists. For example, rubbish is not always being put in the correct bins (waste and recyclable). Before garbage collection day, someone has to sort the rubbish correctly, which takes time and effort.<br><br>The process of defining an **existing** problem can be guided by some key questions, such as:<br>• Who has the problem?<br>• Why does the problem exist?<br>• What would solve the problem?<br><br>**Note:** Students could find three or four existing solutions to a problem, such as recycling games or spelling or maths games. They could select one of these and respond to the above requirements, with adjustments. For example, for a spelling game the question could be:<br>• Who is the audience for this game? (age group, year level, reading level etc).<br>• Why does the problem exist (students are spelling common words incorrectly and they don't know common rules, or they find spelling boring or difficult).<br>• A solution that allows students to select the correct word in response to the spoken word would help improve spelling. | Once the students have a clear idea of the purpose and audience of the solution they can plan how to create the solution. Just like dressmakers follow a pattern to create a garment or chefs follow a recipe to make a meal, programmers follow a design to create a digital solution. One design method is an algorithm.<br><br>Provide students with the opportunity to create and follow an algorithm to complete a particular task, focusing on clear and precise language provided in the correct order.<br><br>Students develop instructions to construct a toy using building blocks. When they do this, they explore and use technical language in their instructions.<br><br>Students create instructions to draw a geometric shape; this is a great introduction to refining commands. Start with instructions to familiar shapes such as a square, triangle or rectangle. Progress to more complex geometric designs. Students work in pairs and take turns to implement each other's instructions.<br><br>*How can I make a digital story fun and more engaging?*<br><br>Integrate English with digital technologies. Students create a storyboard to plan a 'choose your own adventure' story, where the reader is provided with a number of decisions that lead to alternative endings.<br><br>*How can I help someone learn how to do maths problems?*<br><br>Integrate mathematics with digital technologies. Students | Students must convert their written plan (algorithm) into a 'live' solution by using a programming language – they are bringing the plan to life.<br><br>Provide a visual programming tool. Scratch is a commonly used tool with support provided via tutorials and an online community. Other relevant visual programming tools include Snap, which is web-based, and Pyonkee, which works on iPads. Tynker is another app that is web or iOS based.<br><br>Provide students with the opportunity to program a robotic device such as Sphero, Edison or Ozobots. Use the suggested mobile app to control the device using a visual programming language.<br><br>The BBC Micro:bit is a programmable device that can be used in all sorts of digital solutions. The online code editor uses block-based programming. | Brainstorm how existing digital solutions are used for personal uses, at school and in the community. For example, digital solutions can be used at school to take the roll, send out reports, take food orders at the canteen, learn new content and communicate daily school news. Students can identify one or two features of one of these existing solutions that they use and check it against their own solution.<br><br>When evaluating the digital solution, support students to refer back to the initial problem. What were they trying to solve with their digital solution?<br><br>For example:<br>• Quiz-based solution: Were they trying to help others learn about a topic or a language?<br>• Interactive story: Were they intending to make the story more engaging by providing alternative pathways?<br>• Game: Were they intending to help someone learn something using a game scenario?<br><br>How will they know if they were successful? Talk about ways to gather feedback from their target audience. |

| | | | | |
|---|---|---|---|---|
| | Use a table template with column headings that match the guided questions. Alternatively, students create a mindmap that shows the questions and answers.<br><br>Once the problem is defined the next process is to design a solution. | create an algorithm by drawing a flow chart to solve a mathematical problem. | | |
| Supporting resources and tools and purpose/ context for use. | | Plan a 'choose your own adventure' story<br>Students create a storyboard to plan a 'choose your own adventure' story.<br><br>Have fun with flow charts<br>Create a flow chart to represent a sequence of (branching) steps and decisions needed to solve a mathematical problem.<br><br>Take a Lego building challenge<br>In pairs, explore giving and following a sequence of steps and decisions to build a Lego toy.<br><br>Logo workshop contents<br>Logo is a graphical programming language to move a 'turtle'. This website offers a comprehensive range of tasks for students to develop skills in designing instructions to create geometric shapes from simple to complex.<br><br>Turtle programming<br>Program using Logo. | Design a quiz: Convicts – Crime and punishment<br>Students design and create a simple game/quiz to demonstrate convict crimes and punishments.<br><br>Create a language-learning program<br>This quiz is an example of creating a computer program to learn an Aboriginal or Torres Strait Islander language.<br><br>Scratch<br>Scratch is an online tool that uses visual block-based programming language. It enables students to create their own interactive stories and games with how the code is structured providing a visual representation of simple pathways.<br><br>Snap<br>Snap is similar to Scratch. Scratch projects can be imported into Snap.<br><br>BBC Micro:bit<br>Micro:bit's online code editor makes it easy to program your Micro:bit in Blocks.<br><br>Create a board game that uses an Ozobot<br>Create a game board where the player is provided with a number of decisions.<br><br>Sphero and the chocolate factory<br>This activity allows students to use the visual programming software Lightning Lab to control Sphero to act out the role of a fictional character.<br><br>Code bug | |

| | | | | |
|---|---|---|---|---|
| | | | CodeBug is a programmable and wearable device designed to introduce simple programming and electronic concepts and is suitable for students at this age. CodeBug can display graphics and text, has touch sensitive inputs and you can power it with a watch battery. It is easy to program CodeBug using the online interface, which features colourful drag and drop blocks, an in-browser emulator. | |
| Assessment | **Suggested approaches**<br><br>For an existing problem, list:<br>• Who has the problem?<br>• Why does the problem exist?<br>• What would solve the problem?<br><br>For an existing solution, list:<br>• Who was the solution designed for?<br>• What is the purpose of the solution?<br>• Why does the solution solve the problem?<br><br>**Achievement standard**<br>**Define simple problems**, design and implement digital solutions using algorithms that involve decision-making and user input. | **Suggested approaches**<br>Show one example of branching and one example of user input in the algorithm.<br><br>Each student reads aloud part of another student's algorithm.<br>**Note:** Verbalising algorithms is an effective pedagogy for developing an understanding of creating clear and logical instructions. Being able to read aloud an algorithm demonstrates understanding. Of course, the algorithm could be wrong, but this should not affect the assessment of the reader, but rather the creator of the algorithm.<br><br>**Achievement standard**<br>Define simple problems**, design** and implement **digital solutions using algorithms that involve decision-making and user input.** | **Suggested approaches**<br>• Presentation or demonstration of a selected feature of the digital solution, such as user input or decision making.<br><br>• **Dr Scratch**<br>Dr Scratch is a free online analytical tool that provides feedback on Scratch (MIT) project progress.<br><br>**Achievement standard**<br>Define simple problems, design and **implement digital solutions using algorithms that involve decision-making and user input.** | **Suggested approaches**<br>• A list of three questions could be used to evaluate the students' solutions.<br><br>• Positives and negatives of an existing solution that meets a community need, such as a library borrowing system, an information kiosk, an app of local park facilities.<br><br>• Complete the sentences:<br>– My solution works well because …<br>– My solution is similar to an existing solution because …<br><br>**Achievement standard**<br>They explain how their solutions meet their purposes. |