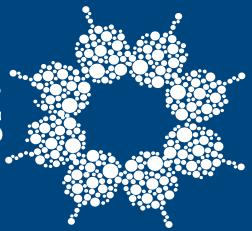


Year 5-6

Digital Technologies



food & agribusiness
SOLUTIONS



An educational resource for Year 5-6 Digital Technology

Programming for Water Management



Programming for Water Management

Year 5-6 Digital Technologies

Content Description

Examine how whole numbers are used to represent all data in digital systems	(ACTDIK015)
Define problems in terms of data and functional requirements drawing on previously solved problems	(ACTDIP017)
Design, modify and follow simple algorithms involving sequences of steps, branching, and iteration (repetition)	(ACTDIP019)
Implement digital solutions as simple visual programs involving branching, iteration (repetition), and user input	(ACTDIP020)
Plan, create and communicate ideas and information, including collaboratively online, applying agreed ethical, social and technical protocols	(ACTDIP022)

Source: Australian Curriculum v8.1

1 <http://www.australiancurriculum.edu.au/science/curriculum/f-10?layout=1-level9>

© Australian Curriculum, Assessment and Reporting Authority (ACARA) 2010 to present, unless otherwise indicated. This material was downloaded from the [Australian Curriculum](#) website (accessed 21 March 2016) and was not modified. The material is licensed under [CC BY 4.0](#). Version updates are tracked on the [Curriculum version history](#) page of the Australian Curriculum website.

ACARA does not endorse any product that uses the Australian Curriculum or make any representations as to the quality of such products. Any product that uses material published on his website should not be taken to be affiliated with ACARA or have the sponsorship or approval of ACARA. It is up to each person to make their own assessment of the product, taking into account matters including, but not limited to, the version number and the degree to which the materials align with the content descriptions (endorsed by all education Ministers), not the elaborations (examples provided by ACARA).



Learning Outcomes

At the end of the unit, students will be able to:

- Write computer programs using Scratch;
- Use algorithmic and computational thinking to solve introductory problems;
- Appreciate the importance of water in an agricultural context;
- Appreciate some of the challenges surrounding water conservation in producing food and fibre;
- Appreciate the concept of optimal water quantity; and
- Understand how computer programs can assist in agricultural processes.

Description

This unit focuses on programming concepts such as algorithms and branching to assist students build confidence in writing programs using the introductory programming language **Scratch**. Students subsequently apply this coding ability to create a program that assists them calculate the exact quantity of water required daily, to grow a plant, such as alfalfa. Alfalfa may be replaced with another plant, such as watercress or beans. It may be possible to integrate this unit with a school garden program. This provides the context of water conservation in Australian agriculture.

This foundation in coding, gives students basic skills in algorithms and computational thinking, as well as the ability to execute these skills in the **Scratch** programming environment. Teachers who wish to extend their class can set up **Scratch**-based homework tasks, such as a show and tell of Scratch programs at the end of each week, or encouraging “competitions” where students submit games to be played by their peers.

Teachers are not required to be experts in coding, computers or technology to undertake this unit. However, teachers with no prior programming experience should make sure they carefully read over the lessons and linked resources ahead of each activity to familiarise themselves with the concepts involved. Most of the computer activities in this unit require the teacher to adopt the role of ‘facilitator’ rather than ‘instructor’. Encourage students to problem solve, and help each other when they run into difficulties.

The activities provide opportunities for individual and group investigation.

Activity 1: Getting Started – My First Hour of Code

Activity 2: Getting Started – Scratching the Surface

Activity 3: Dealing with Drought – Water Management Challenge

Activity 4: Water Management – Scratch Program Basics

Activity 5: All about Algorithms

Activity 6: Plants, Programming, Prototypes

Activity 7: Crop Watch Data Report (ongoing)

Activity 8: Review /Assessment – What did I Learn?

The following is a suggested lesson flow, accompanied by the required resources. Please note that seeds and other resources should be organised ahead of time to ensure the successful running of the activities.

Suggested Lesson Flow:

Week	Lesson / Activity	Description	Resources / Preparation
1	1. Hour of Code 2. Getting Started – “Scratching” the Surface	<i>Topic - Hour of Code</i> (70 min). Stimulate interest in and engage students with coding. <i>Topic - Scratch</i> (60 min) Introduction to Scratch Students create basic games and/or animations with Scratch using the online tutorials.	Computers (one per pair) with internet connection. Computers (as above). Registration for a Scratch teacher account (may take 24 hours to confirm). From this account, teachers can create student accounts and check student progress.
2	3. Dealing with Drought – Water Management Challenge	<i>Topic - Plant Information</i> (90 min) Water and agriculture – why is water important? BTN video “It’s all about the Rain” Farm management activity. Water Management Challenge – Interactive Game	A projector and computer with internet connection. Computers (one per pair) with internet connection (optional). Student Resources Sheets 1-3 for each student.
3	4. Water Management - Scratch basics	“Topic – Program Basics/Flow Diagram” – preparatory work (30min) Setting up of alfalfa plants.	Per student pair: one plastic cup, 4 sheets paper towel, clothes peg, 1/3 cup of soil or potting mix, 1 teaspoon alfalfa seeds (pre-soaked overnight).

Year 5-6

Digital Technologies

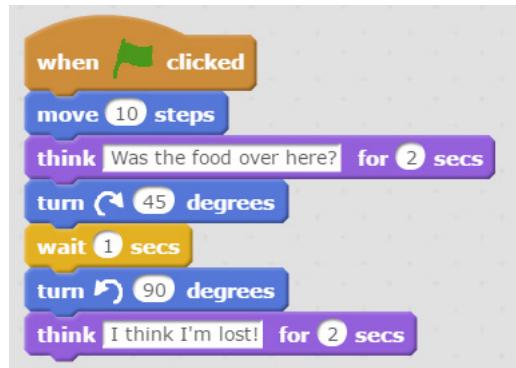
Week	Lesson / Activity	Description	Resources / Preparation
3	<i>5. All about Algorithms</i>	<p>Topic – Program Basics/ Flow Diagram (60min)</p> <p>Students practise flow diagrams.</p> <p>Students construct flow diagram of water needed on butcher's paper.</p> <p>Begin watering plants according to flow-chart.</p> <p>Topic – Algorithms (60min)</p> <p>Explain use of loops, if-statements and variables (score counter).</p> <p>Daily maintenance (10 mins per day)</p>	<p>Butcher's paper (optional).</p> <p>Average humidity and maximum temperature figures for your local region in the current month.</p> <p>A projector and computer with internet connection.</p> <p>Several spray bottles filled with water. Several computers with internet connection. Student activity 7 (sheet).</p>
4	<i>6. Plants Programming Prototypes</i>	<p>Topic – Scratch (60min)</p> <p>Students input variables to change the effect (e.g. customising their name).</p> <p>Students create a watering program from their flow charts (created in week 3).</p> <p>Daily maintenance (10 mins per day)</p>	<p>Computers (one per pair) with internet connection.</p> <p>Computers (one per pair) with internet connection.</p> <p>Several spray bottles filled with water. Several computers with internet connection. Student activity 7 (sheet).</p>
5	Monitor	Daily maintenance (10 mins per day)	Several spray bottles filled with water. Several computers with internet connection. Student activity 7 (sheet).
6	Monitor	Daily maintenance (10 mins per day)	Several spray bottles filled with water. Several computers with internet connection. Student activity 7 (sheet).

Week	Lesson / Activity	Description	Resources / Preparation
7	7. Crop Watch Data Report	Daily maintenance (10 mins per day) Harvest the alfalfa, measure the mass of each crop (yield), and maximum height of highest sprout. Results recorded.	Several spray bottles filled with water. Several computers with internet connection. Student activity 7 (sheet). Rulers/measuring tapes, precision scales. Student activity 7 (sheet).
8	Review – What did I Learn?	30 minutes. Review learning. Discuss variations in height, yield and water usage as a class.	Student activity 7 (sheet). Student activity 8 (sheet).

Key Terminology – Coding

Teacher Resource Sheet 1

Algorithm



A set of instructions for completing a task. For example, a recipe is an example of an Algorithm, being a clear set of instructions that if done correctly, (should) result in an identical output

Bug



Unusual behaviour in a computer program, because the programmer made a mistake somewhere in the code.

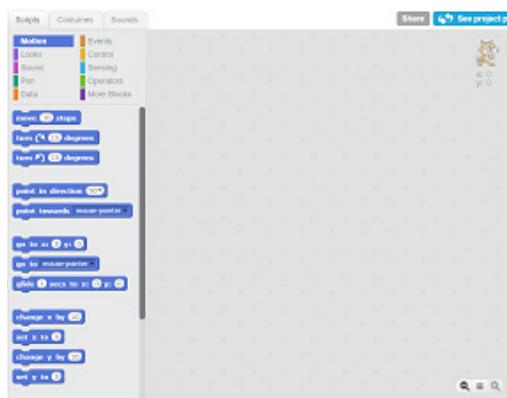
Branching



Branching can be thought of as whenever the computer has a choice to make. This is commonly seen in an "if" statement. So, for example:

IF it is raining outside THEN get an umbrella
ELSE do nothing

Interpreter



Some programming languages need Compilers, which translate the program into computer language. We will be using Scratch, which uses an Interpreter. Unlike a Compiler, an Interpreter can automatically read the program. This means that our program can be incomplete, and the Interpreter will still work.

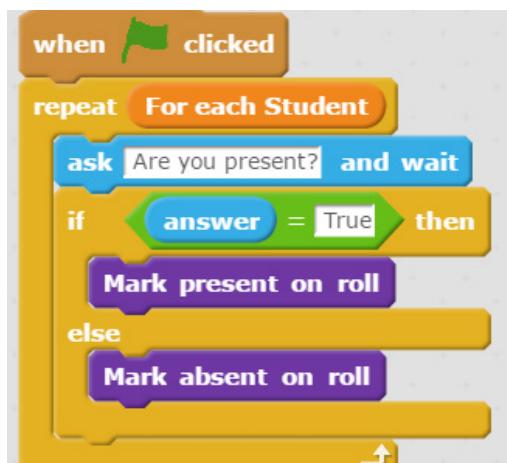
An example is thinking of a book. If it is in a foreign language, we need a Translator (Compiler) in order to read it, and if there is a part that the translator can't understand, the book will never be released in English. On the other hand, if it is in English, I can directly Interpret each page one at a time. It does not matter if the book has been finished or not, I can start reading the story!

Nested



When one thing is contained within another thing, such as a statement contained within an IF statement, it is said to be Nested.

Iteration



Repetition. A sequence of instructions to repeat. For example, if you check attendance of the entire class, you iterate over each item on the class roll and ask “Are you here?”.

Variable



A way to store a piece of information, such as a number or a word. For example, if we wanted a program to ask your name, and then say "Hello <name>", we need to make sure we can change what "<name>" is depending on who you say you are. Rather than create a different version of the program for every possible name imaginable

Boolean



A true or false, usually from comparing the value of two things. If the two things are equal, then it is True

Loop



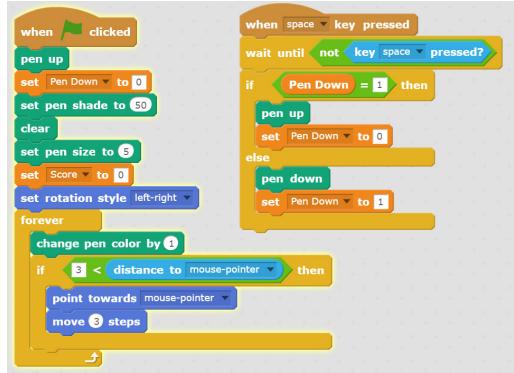
A sequence of instructions that is continuously iterated until it receives an order to stop e.g. it reaches a certain condition.

Sprite



Images that we add algorithms to in order to create projects. They can be user-created, uploaded, or found in the sprites library.

Project



A creation made in the Scratch Program. It can be about anything, from music to animations, art, games and simulations. They can be shared on the Scratch website.

Scratch (or Scratch Program):



A free educational programming language developed by the Lifelong Kindergarten Group at the Massachusetts Institute of Technology (MIT).

Scratch's own glossary - https://wiki.scratch.mit.edu/wiki/Scratch_Terms_Glossary

Lesson 1: Getting Started - My First Hour of Code

Lesson Time: 70 minutes (not including computer log in time etc.)



learn

Learning Outcome

Design, modify and follow simple algorithms involving sequences of steps, branching and iteration (repeating)	<u>(ACTDIP019)</u>
Implement digital solutions as simple visual programs involving branching, iteration (repetition), and user input	<u>ACTDIP020</u>

Description

In this introductory activity, students will learn the basics of programming, using the Hour of Code website at <https://code.org/learn>. The aim is to motivate students to understand the sequential nature of programming.

The Hour of Code activities will prepare students to use the online programming application Scratch in subsequent lessons, which has a similar appearance to the Hour of Code modules.

Teacher Background Information - Hour of What?

Hour of code is a US-based website that provides free resources for teaching the basics of coding. The resources are rich and well-constructed, allowing teachers to act as a facilitator rather than an instructor. Each activity on the website is designed to take roughly an hour for students to complete.

The website contains lesson plans to accompany the activities, although each activity is designed to be suitable for a stand-alone activity conducted by a student at home, so student self-pacing is encouraged rather than moving as a class unit.

An effective way for students to learn coding is to work in pairs, where one student is in charge of the computer, and the other is in charge of making sure there are no mistakes, with students swapping in between “levels”. This is referred to as “pair programming”.

Hour of code can be completed outside of school by the students, if teachers feel students have the necessary capability. **It is strongly recommended that teachers complete an hour of code activity prior to this lesson to familiarise themselves with the concepts involved.**



discuss

Setting the Scene

Introduce the Hour of Code activity by having students view one of the following videos.

Large Group: What Most Schools Don't Teach - <https://www.youtube.com/watch?v=nKlu9yen5nc>

Large Group: Pair Programming - <https://www.youtube.com/watch?v=vgkahOzFH2Q>

Explain the concept of pair programming and importance of team work and communication.

Explain to students that no one has all the answers, but pair programming lets us try out two people's ideas. If one doesn't work, we can try the other person's idea.



Student Activity 1:

Getting Started - My First Hour of Code

It is advisable that students work in pairs to ensure maximum engagement with the activity. If there are insufficient computers, rotate over two sessions. One advantage of pair programming is that if a student is unsure how to continue, the other may be able to assist.

Demonstrate to students how to navigate to the hour of code website, <https://code.org/learn>, and select any of the games in **Learn an Hour of Code**.

Teacher's notes are available underneath each game. Teachers may wish to restrict which games students can choose in order to be more familiar with the content, but most students will be able to work through the exercises with little assistance. The picture below indicates with the red arrow where the "hour of code" exercises are located.

The screenshot shows the homepage of the Code.org website. At the top, there is a teal navigation bar with the letters 'C', 'O', 'D', and 'E' arranged in a 2x2 grid. To the right of the grid are links for 'LEARN', 'TEACH', 'STATS', 'HELP US', 'ABOUT', and 'Sign in'. Below the navigation bar, there are four main tabs: 'Learn an Hour of Code' (which is highlighted in orange), 'Beyond an Hour', 'Find Local Schools + Camps', and 'Code Studio'. Underneath these tabs, there are three game cards:

- Star Wars: Building a Galaxy with Code**: Features characters from Star Wars. Description: 'Learn to program droids, and create your own Star Wars game in a galaxy far, far away. (Ages 6-106)'. Buttons: 'Teacher's Notes' (link: <https://hourofcode.com/star>) and 'Go'.
- Minecraft Hour of Code**: Features Steve and Alex from Minecraft. Description: 'Use blocks of code to take Steve or Alex on an adventure through this Minecraft world. (Ages 6-106)'. Buttons: 'Teacher's Notes' (link: <https://hourofcode.com/mc>) and 'Go'.
- Code with Anna and Elsa**: Features Anna and Elsa from Frozen. Description: 'Let's use code to join Anna and Elsa as they explore the magic and beauty of ice. (Ages 8-108)'. Buttons: 'Teacher's Notes' (link: <https://hourofcode.com/frzn>) and 'Go'.

Below the game cards, there is a section titled 'Partner tutorials that teach JavaScript' featuring a Khan Academy tutorial on 'An introduction to JavaScript'.



learn

Lesson 2: Getting Started – “Scratching” the Surface

Lesson Time: 60 minutes.

Learning Outcome:

Design, modify and follow simple algorithms involving sequences of steps, branching and iteration (repeating)	<u>ACTDIP019</u>
Implement digital solutions as simple visual programs involving branching, iteration (repetition), and user input	<u>ACTDIP020</u>

Description

In this lesson, students are asked to create their first Scratch program. Step-by-step web tutorials are used, with the teacher’s primary objective to facilitate learning and confirm understanding, rather than direct instruction.

Setting the Scene - Pre-Class work (to be completed several days in advance):

1. Create a Scratch teacher account. Go to <https://scratch.mit.edu/educators/register> and follow the simple steps.

It may take up to 24 hours to gain full teacher access, as each application is personally considered.

2. Create a student account for each programming pair. You may decide to call these SchoolNameStudent1 etc. Instructions on managing your students are found at <https://www.youtube.com/watch?v=7HI9GxA1zwQ> or search for “Scratch teacher accounts” on YouTube.

If teachers are unable to make a teacher account, students may still complete this lesson, but will be unable to save their work. Saving is not essential for this lesson but will be important in subsequent lessons.

Teacher Background

Scratch is what is known as a Novice Programming Environment (NPE). NPE’s provide powerful programming tools in a way that removes much of the complexity and possible frustrations of text-based programming languages. Scratch uses a similar block-based interface as the hour of code tutorials on code.org, whilst providing almost infinite freedom to create programs, from simple to complex.

Teachers wishing to become familiar with Scratch are advised to take the same tutorials given to students in this lesson. For a more in depth exploration of Scratch, some comprehensive and free resources are available online, such as:

“Programming in Scratch” at the edX website <https://www.edx.org/course/programming-scratch-harveymuddx-cs002x-1>

“Code Yourself: An Introduction to Programming” on the Coursera website <https://www.coursera.org/learn/intro-programming>

“Scratch” as a part of the “CS50” edx course <https://www.youtube.com/watch?v=697pD31GCZg>

For teachers wishing to extend their programming knowledge, the “CS50” course is available through edx at <https://www.edx.org/course/introduction-computer-science-harvardx-cs50x>. This course is a high-intensity course, being a free online resource from the first year Harvard university computer science course. Teachers will find completing 2-3 weeks of the course extremely useful in gaining a strong foundation in programming and computational knowledge. Teachers should be aware of the high time commitment that this course requires, and so it is only recommended for those with the time and desire to increase their programming knowledge to a high level.



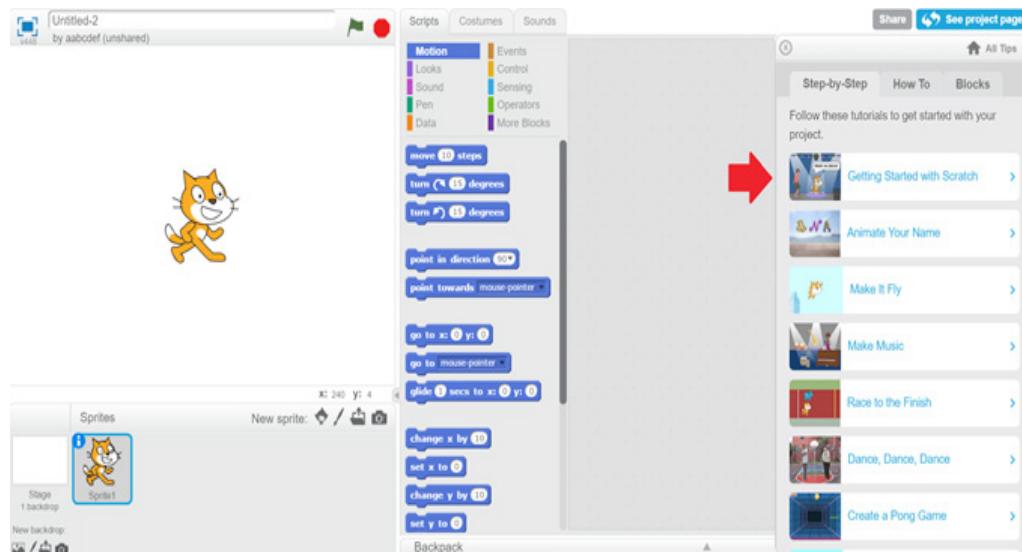
task

Student Activity 2: Getting Started - Scratching the Surface

Group the students, preferably in pairs. Walk them through on a projector how to navigate to scratch (<https://scratch.mit.edu/>) and login to the student accounts which were created earlier.

Students should:

1. Navigate to <https://scratch.mit.edu/>
2. Sign in if you have created usernames for them
3. Click on “create” in the top-left corner of the page
4. Complete the “Getting Started with Scratch” tutorial to make their first game - instructions are provided in a frame on the right-hand-side of the screen
5. Click ‘next’ after completing each instruction
6. Click the small question mark on the far right of the screen to be reinstated, if they accidentally click on the ‘x’ to exit this screen
7. Attempt one of the more advanced tutorials, if they finish early
8. Play with Scratch on the weekend if they have time and share their creations with the class the following week



Lesson 3: Dealing with Drought

Link with Year 6 Science Understanding

Sudden geological changes and extreme weather events can affect Earth's surface	(ACSSU096)
---	------------

Description

This activity provides the background context for the remaining activities in the unit. Students watch the short video segment – Drought Doco from Behind The News (BTN). They then discuss their understanding of drought.

Teacher Background

Drought refers to a period when rainfall is less than consumption of water. One of the major causes of drought is the El Niño/La Niña cycle. During an El Niño phase of the cycle, there is high atmospheric pressure over Australia, pushing atmospheric moisture towards South America. In the La Niña phase, the opposite is true, with Australia experiencing relatively low atmospheric pressures. Due to these climate cycles (along with other factors), Australia is particularly prone to drought, making water conservation and efficiency of great importance.

Drought has two major impacts that we need to consider:

1. The impact on the individual farmer; and
2. The impact on the wider community.

Three methods for dealing with drought are:

1. Storing water, such as building dams;
2. Using water more efficiently. Automatic water systems can use up to 25% less water than manually watering plants; and
3. Changing land use practices, such as growing less water intensive crops.

Setting the Scene

Show the short video – Drought Doco from Behind The News (BTN).

<http://www.abc.net.au/btn/story/s4186862.htm>

The video introduces the concept of drought and highlights some of its devastating consequences. It was filmed in March 2015 by Ellen, a 10 year old student, on her grandfather's property.

Ask students to discuss what drought is, and how it affects farmers. Below are some suggested questions:

1. What is drought?
2. What visible signs of drought were there in the video?

3. What are the immediate consequences of lack of rain for this period of time?
4. How has the family tried to deal with drought?
5. Do you think the measures used by the family were successful? What more could they have done?
6. What are the long-term effects of drought? Suggest how Australia as a nation can prepare for drought

Dealing with Drought - Water Management Challenge – Interactive Game

Teacher Background

In this interactive game, students act as farmers, deciding how much to spend on sowing new crops and how much to spend on purchasing water storage capacity to protect against the effects of drought. Water capacity can be imagined as installing tanks or building dams, and is used as a simplification in place of the complex water rights issues that farmers face.

The activity can be accessed as either print outs or a digital resource at <https://scratch.mit.edu/projects/119579846/>.

The digital resource is easier, as it calculates the difficult arithmetic automatically, allowing students of all mathematical levels to participate. It should be noted that despite the difficulty, the printed resource gives students richer educational insight. Another option is instructing students to fill out the paper sheet whilst playing the digital resource, allowing the computer to make all the necessary calculations.

Playing the digital resource ahead of time is a simple way for teachers to understand the game, regardless of the intended method of delivery.

The digital resource has two game modes available, 'Scratch rolls' and 'player rolls'. The 'player rolls' mode allows the students to manually input the dice rolls. The teacher may have students use this game mode, while rolling a large die in the centre of the room. The second mode is 'Scratch rolls', which rolls the die on the computer, allowing students to play by themselves.

The suggested delivery method for this interactive game is that the teacher or one student rolls the die and everyone uses the same number. Students will see, that depending on which state they are assigned, they would be either lucky or unlucky with rainfall.

After completing the activity, survey students to find out who was able to prevent the greatest number of crops from dying over the entire run, who had the most water left over and who had the most credits at the end.

Start by demonstrating a whole class example on the white board to clarify how the calculations work, as they may appear confusing and difficult to follow for students. Teaching resource 1 shows explicit calculations for the first year.

The game is best played in pairs. This enables students to cross-check answers with each other. However, students should complete their own worksheet.

Instructions for Interactive Game: Water Management Challenge

1. Distribute resource sheets 1 and 2 to students, or alternatively direct them to the Scratch version of the game at <https://scratch.mit.edu/projects/119579846/>. Allocate each student pair a state from Resource sheet 3.
2. Explain how to play. Assume it is a normal year.
3. Players start with 30 units of water stored, 20 credits, and 30 units of water capacity.
4. Water stored is students' usable water supply. It cannot exceed the water capacity. This can be visualised as pouring a 2 litre bottle of water into a small water cup. The water stored inside the cup cannot exceed the capacity of the cup.
5. At the beginning of each year, players purchase crops for \$2 credits each, or increase their water capacity for \$1 credits per unit.
6. For the paper version, students should record the number of crops purchased in the "Summer" row of the corresponding year. They note the water capacity units bought, add them to their existing water capacity units and record their new total under water capacity. This will be done automatically in the digital version.

Example: If a player purchases 7 crops and 6 litres of water capacity, it is recorded as follows

	Dice	Rain	Water Stored	Crops	Credits	Water Capacity
Start	-	-	30	0	\$20	30
Year 1 Summer				7	0	30+6=36

7. Roll a die, or have individual table groups do so. Write the die result in the "Dice" column. Students record the corresponding number from their state card under "Rain". On the digital version, players will simply enter the number when prompted.
8. For the paper version, conduct the calculation: Rain + Water Stored – Crops = X
There are three possible solutions:
 - a. If Rain + Water Stored – Crops < 0. Reduce "crops" by the result.
 - i. For example: If Rain + Water Stored – Crops = -5, then reduce Crops by 5. This means you ran out of water and 5 Crops died of dehydration. Write this new number in the next Crops box, ready for next year, your new water stored should be zero.
 - b. Rain + Water Stored – Crops > 0 or = 0. Write the same number of crops in the box below (as they all survived!), but the result of the equation Rain + Water Stored – Crops is the new Water Stored result for the next turn.
 - c. NOTE: The Water Stored cannot be greater than the Water Capacity number, and any water from rain in excess to the capacity is not saved!
9. Repeat from step 7 for each season, until the "End of Year".

10. Give players 5 Credits for each crop that survived, and set their crops to zero.
Add an additional 10 credits. Start again from Step 4, until the fourth year. Again, this is done automatically in the digital version.

NOTE: During the fourth year, there is a Drought. Players use the “Drought” set of rolls for their state. This reduces the amount of water they can expect by about 65%. **Do not inform students that there is a drought in year four until they have already planted year 4’s crops! Only inform them that they are to expect a drought very soon.** The devastating effect of being unprepared is one of the main learning outcomes of the game.

Teaching resource 1 shows a demonstration of a single game.

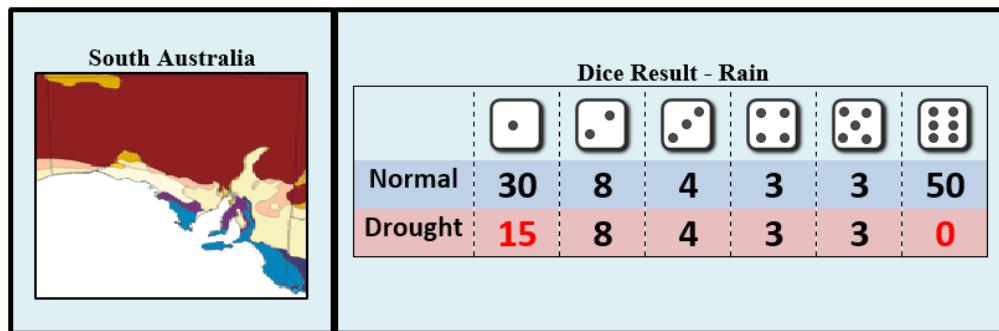
TIP: get the students to explicitly work out the calculations on a piece of paper, work in pairs to cross-check each other’s work. Provide calculators if available.

From the example game provided, students will see that the water tanks were filled up beyond capacity many times. In year 3, the player appeared to have enough water, but when the drought came in year 4, nine of the 15 crops were lost due to insufficient water storage.

Before playing the **Water Management Challenge Game**, ensure students have copies of Student Resource Sheets 1,2 and 3.

Water Management Challenge

Teacher Resource 1



	Dice	Rain	Water Stored	Crops	Credits	Water Capacity
Start	-	-	30	0	\$20	30
Year 1	3	4	30	10	0	30
	3	4	$4+30-10=24$	10	0	30
	6	50	$4+24-10=18$	10	0	30
	6	50	$50+18-10=60 > 30$	10	0	30
	-	-	$50+30-10=70 > 30$	0	$10 \times 5 + 10 = 60$	30
Year 2	6	50	30	15	0	$30 + 30 = 60$
	1	30	$50+30-15=65 > 60$	15	0	60
	6	50	$30+60-15=75 > 60$	15	0	60
	1	30	$50+60-15=95 > 60$	15	0	60
	-	-	$30+60-15=75 > 60$	0	$15 \times 5 + 10 = 85$	60
Year 3	1	30	60	15	0	115
	4	3	75	15	0	115
	5	3	63	15	0	115
	4	3	51	15	0	115
	-	-	39	0	85	115
Year 4	4	3	39	15	0	170
	3	4	27	15	0	170
	2	8	16	15	0	170
	6	0	9	15 9	0	170
	-	-	0	0	40	170



Student Activity 3: Dealing with Drought

Student Resource Sheet 1

Water Management Challenge

Farmers across Australia have to decide how many crops they can plant every year.

- What is the risk to farmers if they plant too many crops?
-

- What is the risk to farmers if they plant not enough crops?
-

- What type of climate or weather might unexpectedly damage a farmer's crops?
-

The amount of crops farmers plant in a year depends on many things. Farmers need to make sure they have enough water, or the crops will die. Farmers rely on the income they receive from selling their crops to buy new materials for the next year, like seeds, fertiliser, fuel and water.

In this activity, you are a farmer for four years, in one of four different Australian states. The rules are simple!

Every year, you need to decide how many credits to spend on buying crops, and how much to spend on buying water tanks to store excess water.

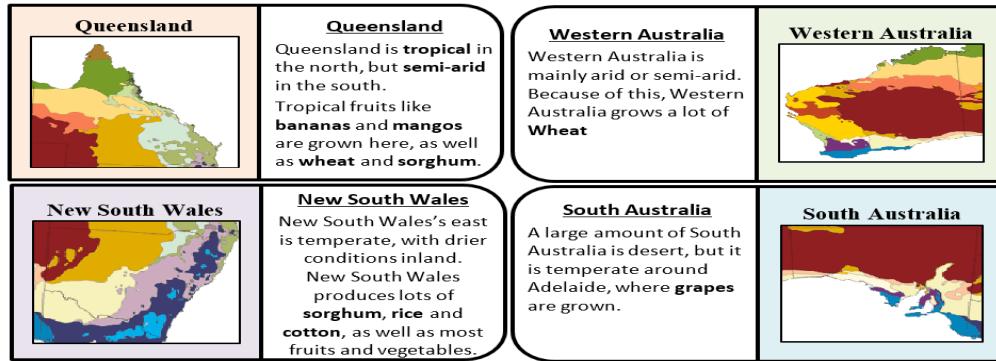
Each year has four "turns", Summer, Autumn, Winter, and Spring.

Each turn, you roll a dice to see how much it rained. All plants need one unit of water, and if you can't water the plant, then it dies! If you have 10 crops, but it only rains seven units of water, then three crops die, leaving only seven. The key to avoiding this is to have water stored for when it doesn't rain enough.

The amount of water you get from rolling the die depends on which state your farm is located in. While some die results will be great for some farmers, they may not be good for others.

Assume that scientists are predicting a drought in the next few years, so make sure you have enough water stored!

Below are the four states your farm might be in.





Student Activity 3: Dealing with Drought

Student Resource Sheet 2

Water Management Challenge

Name:

Glue the card for your given state in here

Shop	
Item	Credits
Purchase Crops	Spend \$2
Sell Crops	Gain \$5
Buy one litre of Water Capacity	Spend \$1

Hints:

- Buy plenty of water capacity for when drought hits!
- If low on stored water, buy more capacity and hope it rains!

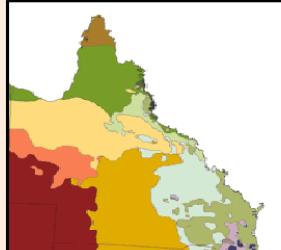
	Dice	Rain	Water Stored	Crops	Credits	Water Capacity
Start	-	-	30	0	\$20	30
Year 1						
Summer						
Autumn						
Winter						
Spring						
End of Year	-	-				
Year 2						
Summer						
Autumn						
Winter						
Spring						
End of Year	-	-				
Year 3						
Summer						
Autumn						
Winter						
Spring						
End of Year	-	-				
Year 4						
Summer						
Autumn						
Winter						
Spring						
Final Result	-	-				



Student Activity 3: Dealing with Drought

Student Resource Sheet 3

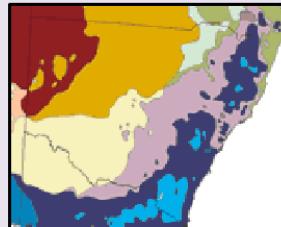
Queensland



Dice Result - Rain

	•	••	•••	••••	•••••	••••••
Normal	2	3	5	8	30	50
Drought	2	3	5	8	15	0

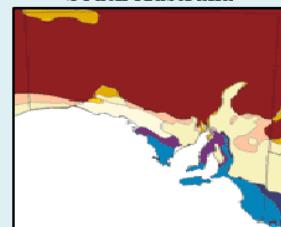
New South Wales



Dice Result - Rain

	•	••	•••	••••	•••••	••••••
Normal	8	1	30	6	3	50
Drought	8	1	15	6	3	0

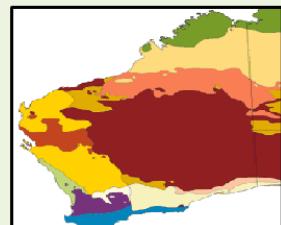
South Australia



Dice Result - Rain

	•	••	•••	••••	•••••	••••••
Normal	30	8	4	3	3	50
Drought	15	8	4	3	3	0

Western Australia



Dice Result - Rain

	•	••	•••	••••	•••••	••••••
Normal	4	30	2	3	9	50
Drought	4	15	2	3	9	0

Lesson 4: Water Management - Scratch Program Basics



learn

Learning Outcome:

Generate, develop and communicate design ideas and processes for audiences using appropriate technical terms and graphical representation techniques	(ACTDEP025)
Negotiate criteria for success that include sustainability to evaluate design ideas, processes and solutions	(ACTDEP027)
Develop project plans that include consideration of resources when making designed solutions individually and collaboratively	(ACTDEP028)
Examine how whole numbers are used to represent all data in digital systems	(ACTDIK015)
Define problems in terms of data and functional requirements drawing on previously solved problems	(ACTDIP017)
Design, modify and follow simple algorithms involving sequences of steps, branching, and iteration (repetition)	(ACTDIP019)
Plan, create and communicate ideas and information, including collaboratively online, applying agreed ethical, social and technical protocols	(ACTDIP022)

Description

This lesson has two components.

1. Students plant their “crop” of alfalfa seeds.
2. Students create a flow chart outlining the steps involved in developing a water management computer program. The flow chart will form the basis for programming their watering system.

Preparation & Equipment

Research the average maximum temperature and humidity for the current month in your area.

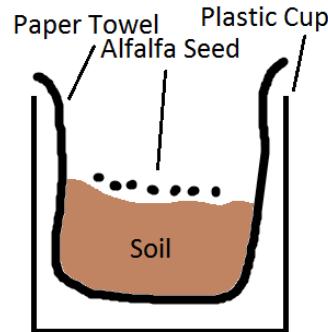
Have students work in their programming pairs. Each pair requires:

- one plastic cup
- four squares of paper towel
- a teaspoon of alfalfa seeds (soaked overnight)
- one clothes peg
- several spray bottles filled with water
- Soil or potting mix

NOTE: Store the alfalfa in the classroom at night. Ensure plants receive some sunlight each day.

Have students:

- Position the 4 paper squares on top of each other, before pushing them into the cup.
- Peg the paper towel to the edge of the cup.
- Partially fill the cup with soil or potting mix to allow root systems to develop.
- Sprinkle the pre-soaked alfalfa seeds on the soil.



Explain that the seeds will take several weeks to grow, and will need to be watered every day. The alfalfa should be kept moist at all times. If it is drying out during the day, more water must be added.

If necessary, apply some fertiliser 2-3 weeks into the activity. If your school has a local garden, consult with the garden manager.

The remainder of this lesson will be deciding how much water is needed, using flow-charts.

Teacher Background Information – Scratch Program Basics

Solving problems with computer programs usually involves breaking the problem into a series of smaller steps. Each step can then be programmed individually in a more manageable way than tacking the whole problem at once.

Flow charts are an excellent visual tool for showing these programming steps, as the flowchart format closely matches the flow of computer code. Most students will readily understand the function of flow charts when presented with simple, real-world examples.

There are many conventions for flow charts, although for the purposes of this lesson it is acceptable to teach the simplest.

- Rounded rectangles - for “start” and “finish” boxes.
- Rectangular boxes for simple events that happen.
- Diamond boxes for choices (branches).

Getting Started.

Draw this flow chart on the whiteboard or display it on a projector. Discuss how it works with the class.

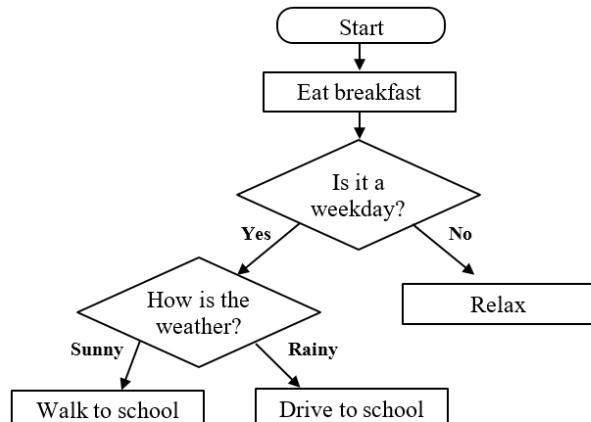
Give students 5-10 minutes to develop a simple flow chart for a topic such as:
Eating breakfast.

Brainstorm on the board possible steps such as:

Pouring milk

Looking in the cupboard

Toasting bread.



Allow a few minutes for students to share their work with a partner and discuss similarities and differences between their created flowcharts.



Student Activity 4: Scratch Program Basics – Constructing a flowchart

In this activity, students construct a flowchart on butcher's paper for calculating the watering requirements of their alfalfa crop. Explain to students that you will provide rough guidelines on how to calculate the correct amount of water for their plants each week. Emphasise that their assessment will not rely on the success of their alfalfa crop, but rather their reflections on the unit.

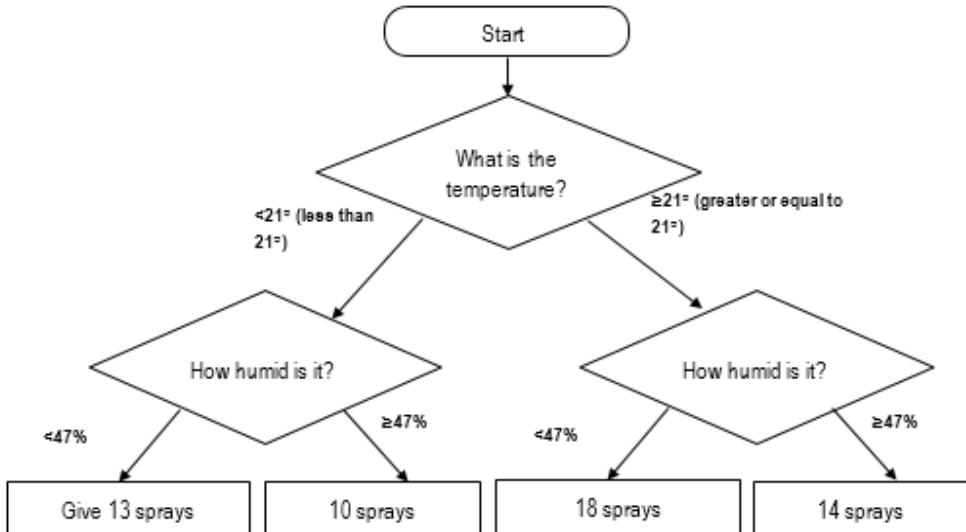
Teachers need to consider local circumstances and climate. Explain to students that in some extreme climatic conditions, their alfalfa seeds may need as many as 40 sprays per day. However, students should aim to use as little water as possible to conserve water. The goal should be to keep their plants damp but not waterlogged.

Write the word "temperature" on the board and ask students to discuss with their partner how this would affect the amount of water needed each day. Do the same with the word "humidity".

Teachers may choose to have students investigate the average high temperature and humidity for the local area for the current month. Record these next to the two words on the board.

Discuss with the students how watering may be altered if a day is hotter, cooler, drier or wetter than the figures below.

Finally, have students construct their own a flow chart to determine the watering amount for each day. For example, if the average temperature for this time of year is 21 degrees and the average humidity is 47%, the flow chart may look like this:



Whilst student flowcharts will vary, they should follow this basic design to make their subsequent programming task manageable. Once teachers have checked the appropriateness of the numbers the students may use the flowchart to start watering their crop.

Explain that after some further investigation into coding, they will use the flowcharts to create a computer program to do the hard work for them.

Student Activity 5: All about Algorithms

<https://scratch.mit.edu/>

Lesson Time: 60 minutes.



learn

Learning Outcomes

Examine how whole numbers are used to represent all data in digital systems	(ACTDIK015)
Define problems in terms of data and functional requirements drawing on previously solved problems	(ACTDIP017)
Design, modify and follow simple algorithms involving sequences of steps, branching, and iteration (repetition)	(ACTDIP019)
Plan, create and communicate ideas and information, including collaboratively online, applying agreed ethical, social and technical protocols	(ACTDIP022)

Description

This lesson focuses on understanding computational ideas relating to algorithms in order to win a “guess the number” game. Students will extend their knowledge of programming, loops, if-statements and the use of variables.

Teacher Background Information

Computational thinking is a key skill for programming, and helps programmers design efficient programs. An example of using computational thinking in our daily lives is when we search through a phone book, or search for a name on a roster. If we want to find “Sam”, then we will generally not start at “A” and look for Sam, instead we will open roughly to the middle of the phone book. If the page we are on says “N”, then we know “Sam” won’t be in the previous section, as S is not between A-N. Through this method of dividing the book up into smaller and smaller sections, we can quickly find Sam, even if the book contains thousands of pages.

Preparation

Set up computers for use.

Setting the Scene

Ask the students what they think it means to “Think like a computer”

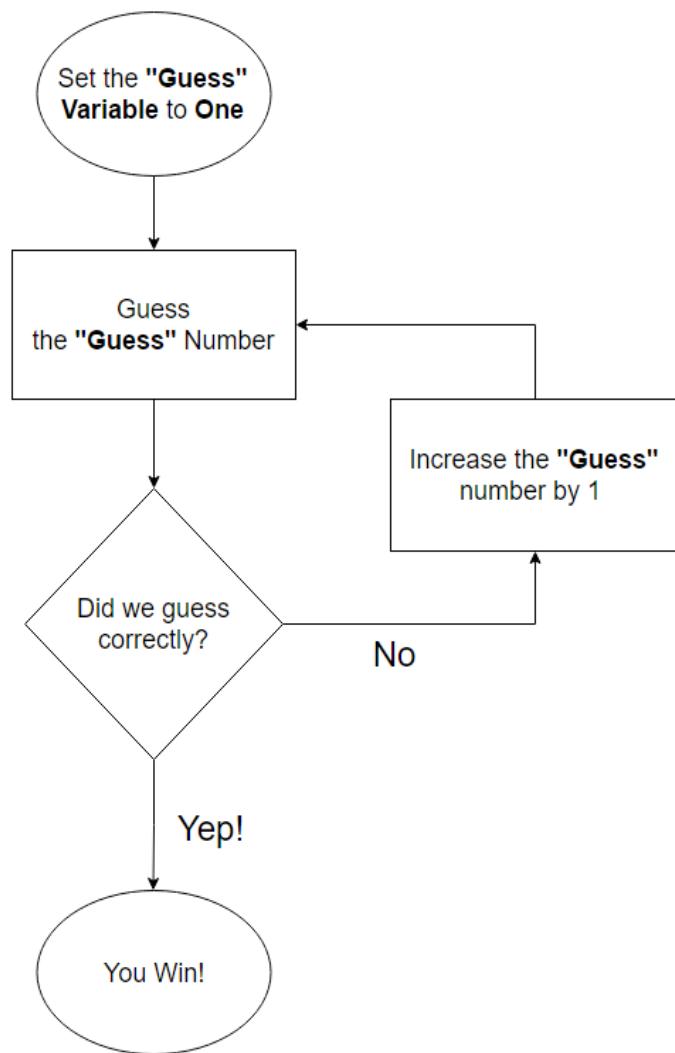
Ask: What is the most efficient way to count the number of people in the class? Students can discuss with a partner or in small groups. Count the number of people in the class using one of their proposed methods. Tell students that at the end of the lesson, they will return to this task with a quicker counting method.



Class Activity

- Open up the scratch game <https://scratch.mit.edu/projects/120979887> and press the green flag at the top right of the project screen to run the game. If unavailable, the “guess the number” game can be done offline with the teacher writing down the number between 1-100 on a card, hiding from the class until the end.
- Ask one student at a time to come to the front and guess the number; writing numbers on the board as they are guessed.
- After the student guesses the number correctly, ask the students to go to their computers to try out the game, aiming for the lowest score. Who can get the lowest score in the game?
- Next, ask the students what instructions they would need to give to someone who has never played this game before, so they could always win.
- Examples of possible slow Algorithms are shown on the following pages. Display these for the students and ask them if there are any problems with the algorithms.

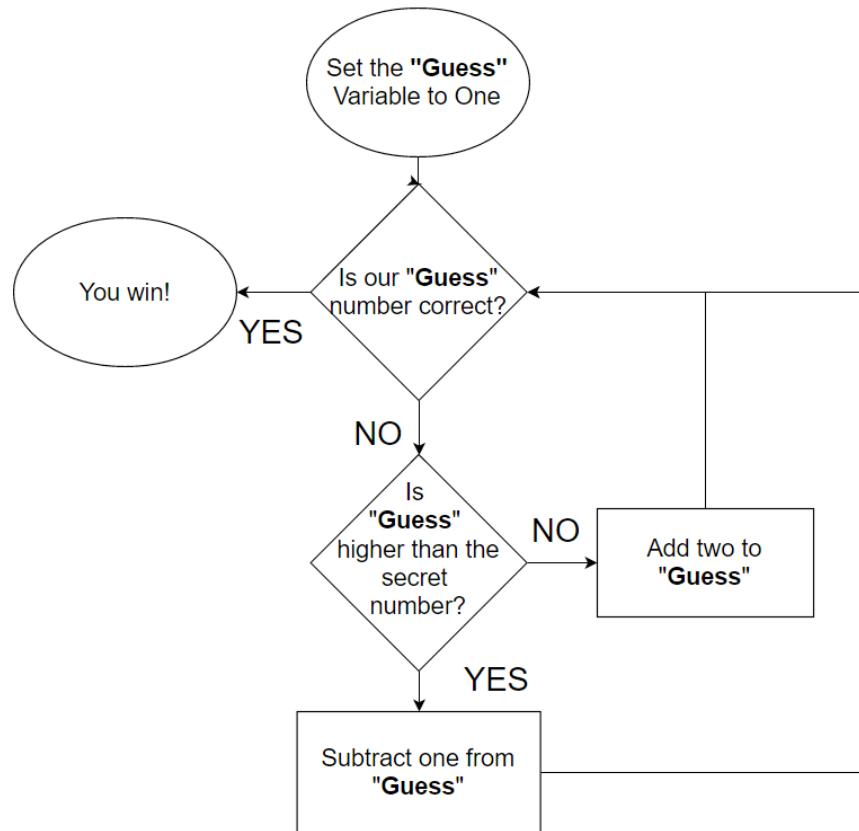
All about Algorithms - Algorithm #1



1. Set the “Guess” Variable to One.
2. Guess the “Guess” number.
3. If “Guess” is correct, go to step #4, ELSE GO TO 3a):
 - a. set Guess = Guess + 1, then go to step #2
4. You win!

All about Algorithms - Algorithm #2

Problem: If the number is 100, you need to guess 100 times! How do we fix this?



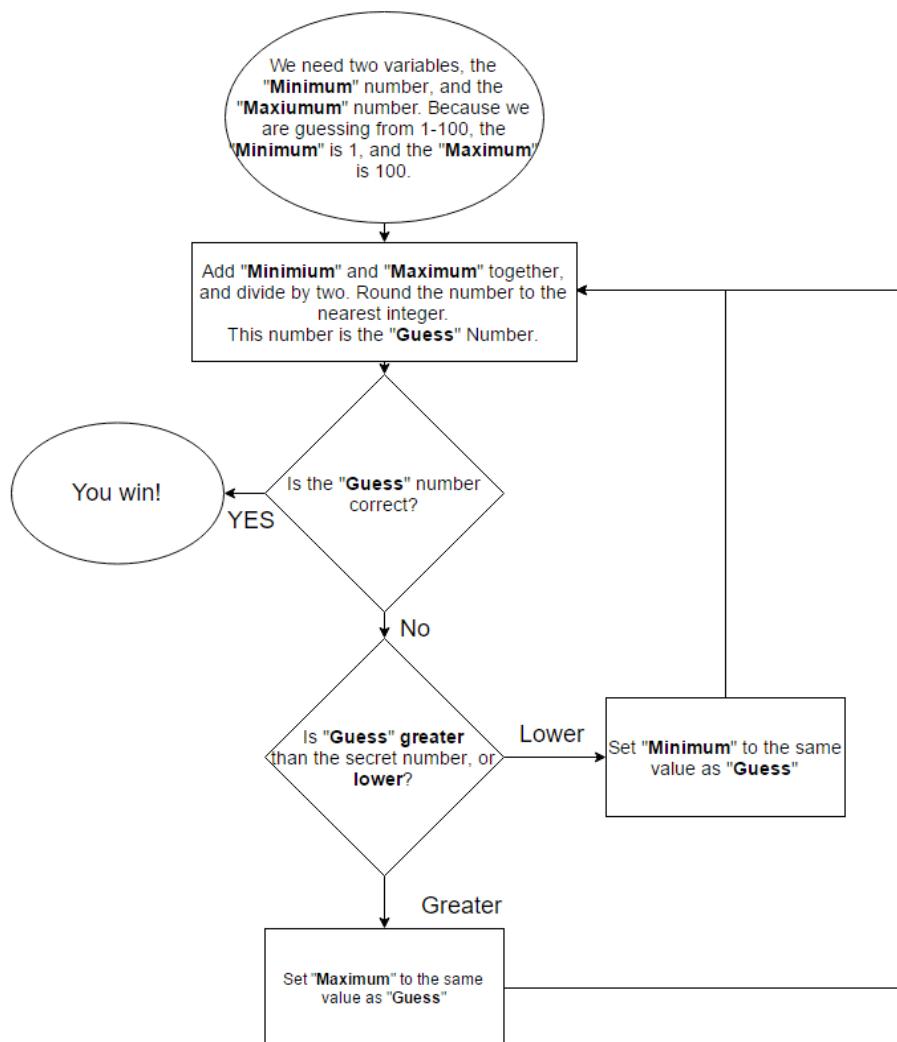
1. Set the “Guess” Variable to One.
2. Guess the “Guess”.
3. If number is correct, go to #4, ELSE GO TO 3a:
 - a. If “Guess” is greater than the secret number, set “Guess” = “Guess” – 1 and go to step #2, ELSE GO TO 3b.
 - b. set “Guess” = “Guess” + 2, then go to step #2
4. You win!

Problem: We can get to 100 twice as fast with this method, but how could we guess a number between one and 1 million, as computers can? Taking half a million turns is not very efficient. There is an algorithm that guesses the number between one and 1 million in only 21 turns!

Open up <https://scratch.mit.edu/projects/120980614>, again clicking on the green flag to start the project, and take answers from the class to guess the number between 1 and 1 million. The program suggests the number to put in. Run this through with students, before showing them how it does this.

All about Algorithms - Algorithm #3

Run through with the students the following algorithm. The algorithm's structure is to always half the remaining range of numbers. To guess one to 100, we first guess 50. If 50 is too high, then we know it's below 50, but higher than one, so we next guess 25 etc.



Draw the following table on the board for students do the calculations as a group. The table is a worked example using 58 as the secret number.

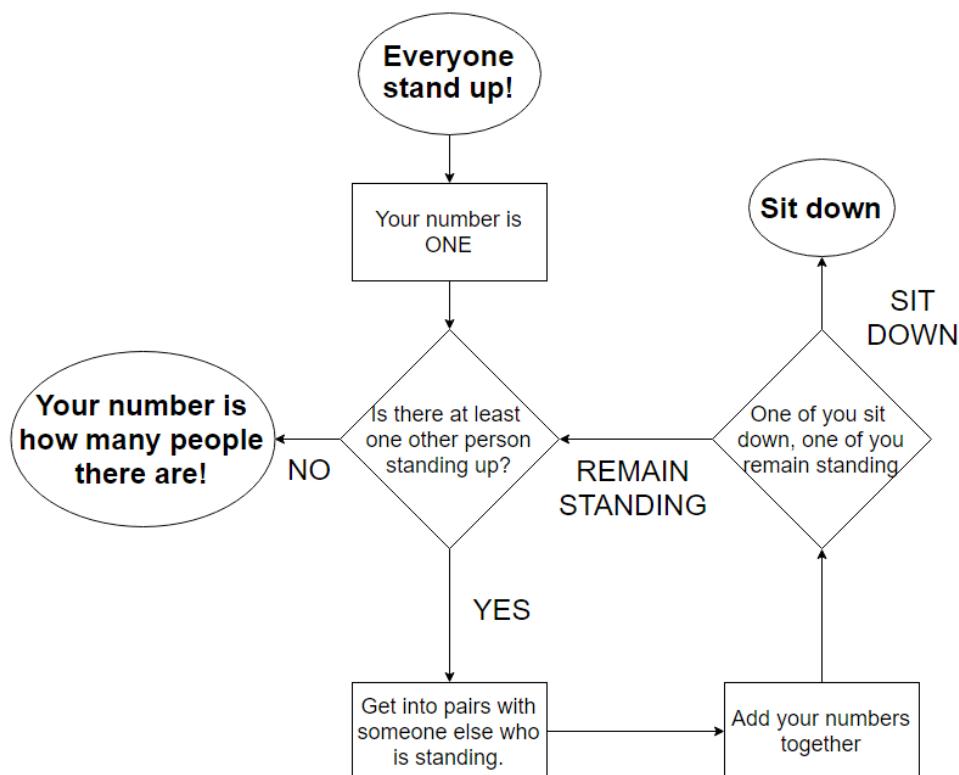
Guess	Is the secret number Higher or Lower?	New Minimum	New Maximum	$\frac{\text{Minimum} + \text{Maximum}}{2}$
-	-	0	100	50
50	Higher	50	100	75
75	Lower	50	75	63 (62.5)
63	Lower	50	63	57 (56.5)
57	Higher	57	63	60
60	Lower	57	60	59 (58.5)
59	Lower	57	59	58
58	Correct!	-	-	-

Conclusion

Finally, ask students to suggest an algorithm to count the number of people in the class (and even the entire school) that would be quicker than counting by ones.

Solution

After listening to student ideas, show the class the algorithm below. The number of students still standing halves each “cycle.” If counted one-by-one, the number of students left to count decreases only one-by-one. Computers save time by using these type of algorithms to do difficult calculations. Try it out as a class (this will require some teamwork and several attempts).



Lesson 6 – Plant Programming Prototypes

Lesson Time: 60 minutes



learn

Learning Outcomes:

Examine how whole numbers are used to represent all data in digital systems	(ACTDIK015)
Acquire, store and validate different types of data, and use a range of software to interpret and visualise data to create information	(ACTDIP016)
Define problems in terms of data and functional requirements drawing on previously solved problems	(ACTDIP017)
Design a <i>user interface</i> for a <i>digital system</i>	(ACTDIP018)
Design, modify and follow simple algorithms involving sequences of steps, <i>branching</i> , and <i>iteration</i> (repetition)	(ACTDIP019)
Implement digital solutions as simple visual programs involving <i>branching</i> , <i>iteration</i> (repetition), and user <i>input</i>	(ACTDIP020)
Plan, create and communicate ideas and information, including collaboratively online, applying agreed ethical, social and technical protocols	(ACTDIP022)

Description

Students translate their flow chart, produced in Activity 4, into a Scratch program by turning the algorithm they created on paper into a computer algorithm. They use their Scratch program to determine how much water to apply daily to their alfalfa plants.

Teacher Background Information

The concept of a fully automated farm might sound like something in the distant future, but there could very soon be “robot farms.” Automated agriculture is an attractive business prospect. It can:

- Reduce labour costs;
- Improve product quality control; and
- Enable more efficient use of resources such as recycling water.

Discuss examples of farms which are heading towards full automation. In Japan, there are farms which grow lettuce using robots to plant the seeds, water, and harvest the lettuce. One such farm produces 50,000 lettuces every single week!

Setting the Scene

Explain to students that they will be programming the computer to give watering instructions for their alfalfa crop.



Student Activity 6: Plants Programming Prototypes

task

Students access the computers in pairs or groups to use Scratch. Guide them through each step to create the program below.

1. Ask my name and age

Students should navigate to <https://scratch.mit.edu/projects/122089923/>. Ask the students to look at the code without running it and imagine what it will do.

Ask students to run the code by clicking the green flag on the project. Did the code do what the students expected?

Have students explain to their partner what they believe the 'join' block did in the code.

Explain that the join block function allows the user to add a variable directly into the sentence. In this case, it allowed the coder to join the variable 'name' with the phrase - 'how old are you?'



2. Follow up and Digitise it.

Now students should know enough to turn their watering flow diagrams into computer code using Scratch. They will create a program that will instruct them on how much water to add each day depending on temperature and humidity.

Ask each student pair to navigate to scratch.mit.edu (if they are not already there from the previous activity), clicking on the ‘create’ tab in Scratch to start a new project.



As in lesson 2, students will create a Scratch program by dragging blocks onto the screen.

Students may need guidance to translate their program correctly onto Scratch. When guiding students, you may find it helpful to refer to Teacher Resource 2.

Discuss as a class which variables they will need to create (Temperature and Humidity) in the “Data” section.

1. Explain that they will need to use the “Ask” block in the “Sensing” section to ask the user for the current temperature and humidity and use the “Say” block in the “Looks” section to tell the user how much water to use at the end of the program.
2. Emphasise that they will need to use the “if” or the “if else” block found in the “Controls” section to work out this answer.
3. After finishing the basic program, students may like to change the background and sprites, or even create a short animation to make their project more visually interesting.

Sample Water Management Program: Teacher Resource 2

This is an example of a student water management program. You may use it for your own reference when guiding students, or even show students who are struggling





task

Student Activity 7: Crop Watch – Data Report

Student Resource Sheet 4

Run your computer program each day, entering the temperature and humidity. Record the amount of water added daily (measured in bottle sprays). You may adjust your watering from what your program tells you if your alfalfa is getting too wet or dry. You may choose to alter your program.

Team Members:

	Week 1	Week 2	Week 3	Week 4	Week 5
Monday					
Tuesday					
Wednesday					
Thursday					
Friday					
Total					

Time to harvest! What were your results?

<i>Crop Size (from soil to top of highest sprout)</i>	
<i>Crop Yield</i>	
Total Amount of Water Used	



Review – What did I Learn?

Lesson Time: 30 minutes



learn

Learning Outcomes:

Explain how student solutions and existing information systems are sustainable and meet current and future local community needs	(ACTDIP021)
Examine how people in design and technologies occupations address competing considerations, including sustainability in the design of products, services, and environments for current and future use	(ACTDEK019)

Description

This reflection activity reinforces the learning outcomes and also provides an opportunity for students to practise reflective writing.

This reflection activity enables students to demonstrate creativity in considering alternative ways for farmers to save water, while also thinking about how humans differ from computers and how they could have improved their own program.

Teachers can assess students' understanding of the activity's key concepts. Students will be asked to reflect on the following concepts:

- Creative methods for farmers to save water
- Human-computer interaction
- Improving their own program
- Key definitions

Setting the Scene

Ask students to complete the What Did I Learn reflection worksheet.

Depending on how you wish to use the assessment data, you may choose to prompt students with some suggested hints from below. (Note: This is not a comprehensive list of solutions.)

1. Using crops that need less water; Recycling water; Preventing water from being evaporated
2. Programs follow instructions much more literally than humans; Computers don't have 'common sense'
3. This response depends on their own code. For instance, they may have been using too much water for both warmer and cooler conditions, in which case they might say that they would reduce the amount of water the code recommended they use
4. Definitions can be found in the 'Key Words' section or online.



Student Activity 8: Review - What did I Learn?

Student Resource Sheet 5

Congratulations!!!

You have completed the unit. Hopefully, your plants grew successfully and you saved water by making sure you didn't give them more water than they required.

Questions:

- 1) Suggestion some other ways for farmers to save water?
 - 2) You wrote a program that told you what to do.
Did you always agree with what your program told you to do?
Why do you think this is?
 - 3) What would you change in your program to improve it and why?
 - 4) What do the following words mean when we talk about programming?

Algorithm: _____

www.nature.com/scientificreports/

Loop: _____

Variable: _____

- 5) Suggest another task that robots could do? Who would benefit?

Write down a basic set of instructions, like you did for your plant, to give to the robot to complete the task.

Produced by**AgriFood Skills Australia**

General inquiries:

Phone: 02 6163 7200

Fax: 02 6162 0610

Email: reception@agrifoodskills.net.au

Web: www.agrifoodskills.net.au

Location

Level 3, 10-12 Brisbane Avenue

Barton

ACT 2600

Postal address

PO Box 5450

Kingston

ACT 2604

Developed by

Steven Landman

James Liu - ANU Robogals

Matthew Mc Dowell - ANU Robogals

© AgriFood Skills Australia 2016

This work is licensed under a Creative Commons Attribution 4.0 International licence.



Images 16, 18-22: derived from http://www.bom.gov.au/iwk/climate_zones/map_2.shtml

Images 1-12, 15, 30-31: derived from software available at <https://scratch.mit.edu/>

Image 13: logo of <https://scratch.mit.edu/>