UNITED ARAB EMIRATES
MINISTRY OF EDUCATION

DESIGN AND TECHNOLOGY

Grade 6

TERM 2

GRADE 6
TERM 2

Authored and designed
by a specialised committee from the Ministry of Education

First edition 1438-1439 A.H. / 2017-2018 A.D.

"Extensive knowledge and modern science must be acquired. The educational process we see today is in an ongoing and escalating challenge which requires hard work.
We succeeded in entering the third millennium, while we are more confident in ourselves."

H.H. Sheikh Khalifa Bin Zayed Al Nahyan
President of the United Arab Emirates

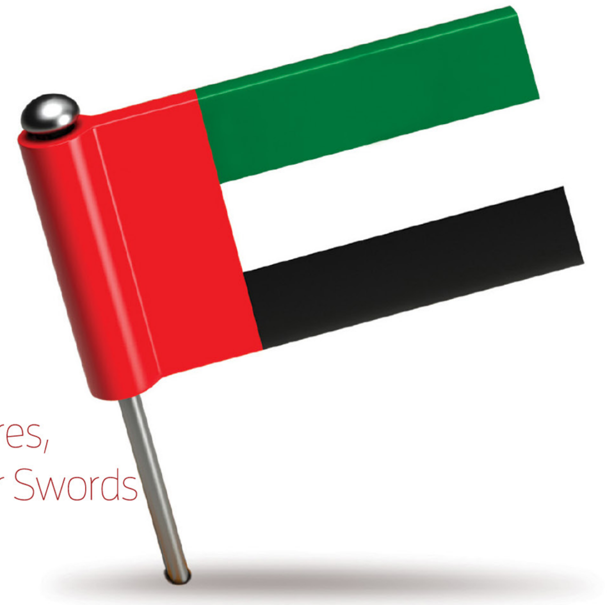
Dear Student,

Al Diwan App

To get the digital version of the book, visit the below link to download the app
www.elib.moe.gov.ae/MoElib/getting-started

Get it from Microsoft | Download on the App Store | GET IT ON Google Play

The Meaning of the United Arab Emirates Flag Colors



The colors of the United Arab Emirates (UAE) flag are inspired by the famous verse of the poet Safiyuddin Al-Hilli:

White are our deeds, Green are our pastures,
Black are our Battles, Red are our Swords

⬜	Symbolizes goodness, welfare and giving, as well as the State's approach of supporting worldwide peace and security.	🟩	Symbolizes growth, prosperity, green environment and cultural revival in the country.
⬛	Symbolizes the strength, staunchness and might of the people of the State, as well as the rejection of injustice and extremism.	🟥	Symobolizes the sacrifices of the pre-union generation, and of the nation's martyrs who sacrificed their lives to protect the homeland's achievements and gains.

The UAE Vision 2021

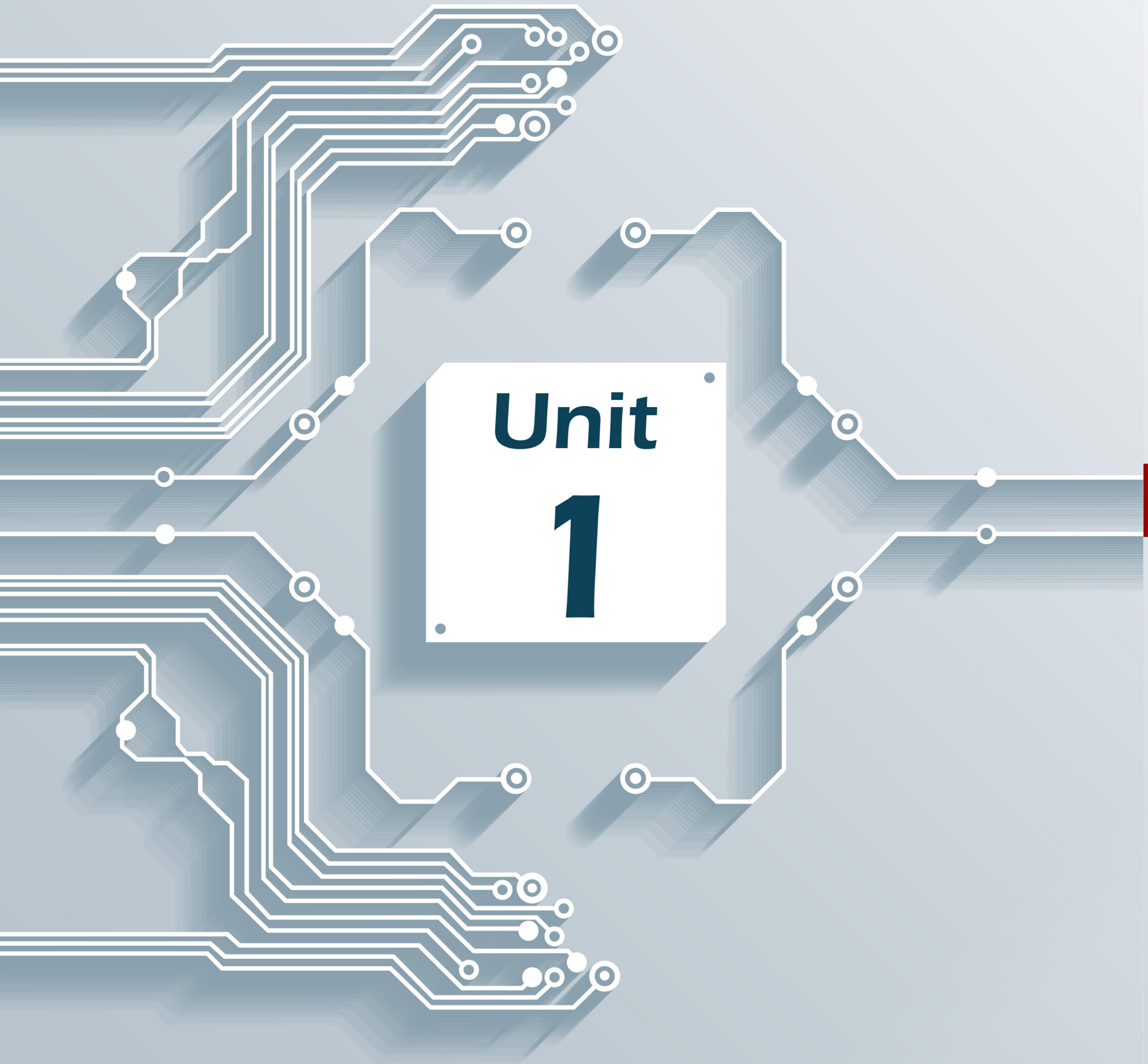
United in Responsibility
- Confident and responsible Emiratis
- Cohesive and prosperous families
- Strong and vital social relations
- Rich and vibrant culture

United in Destiny
- Following the example of the Founding Fathers
- Safety and security of the nation
- Enhancement of the UAE's status on the international arena

United in Knowledge
- Harness full potential of national human capital
- Sustainable and diversified economy
- Knowledge-based and highly productive economy

United in Prosperity
- Long and healthy life
- First-class educational system
- Well-rounded lifestyles
- Environmental protection

Tabke Of Content



Unit

1

Introduction to the Algorithms and Maker

Learning Outcomes

1.1. Exercise creativity and resourcefulness by coming up with ideas for using simple household materials to accommodate the Maker's size and weight in many different ways

1.2. Learn how to download programs and move them to the Maker file to run the Maker

1.3. Apply their understanding in a creative way by making a Maker:pet creature

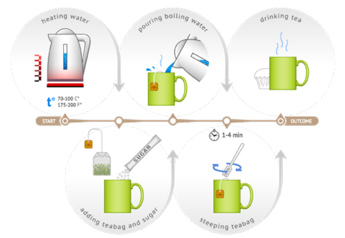
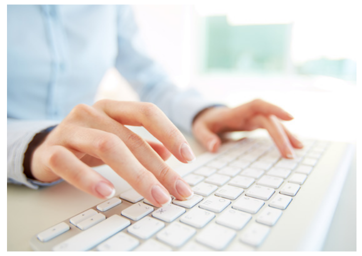
1.4. Understand that the Maker takes input, and after processing the input, produces output

1.5. Learn the variety of different types of information the Maker takes in as input

Overview

Students will get to know the Ibtikar Maker. They will learn about the features of Maker, including the buttons, the Maker code interface – the simulator, the toolbox and the workspace. They will also learn how to download code to the Maker with a USB. This unit introduces a framework. This framework is based on how a computing device works. It is something that uses code to process one or more inputs and send them to an output(s).

Keyword

Term	Definition	Image
algorithm	different options available in a software package	
unplugged	software used to make animations.	RELAX UNPLUG SLOW DOWN BREATHE TAKE IT EASY MEDITATE ENJOY LIFE HAVE FUN SMILE
input	A subject or topic	
output	A subject or topic	

Term	Definition	Image
pseudocode	A program written in plain English which is very close to actual code is called pseudocode. Pseudocode usually uses programming words such as input, output, if, if then else, repeat, etc.	
iteration	a process of repeating something over and over again	
conditional statement	allows the computer to choose the correct option based on a specific input	
variables	an empty location in the memory	

Learning Outcomes

- Show creativity by coming up with ideas for using simple household materials to accommodate the Maker's size and weight in many different ways
- Learn how to download programs and move them to the Maker file to run the Maker
- Apply their understanding in a creative way by making a Maker:pet creature
- Understand that the Maker takes input, and after processing the input, produces output
- Learn the variety of different types of information the Maker takes in as input

Introduction

The Maker is a great way to teach the basics of programming. The Ibtikar Maker block-based coding environment is a good way to make the Maker react to all sorts of inputs. You can introduce ideas such as iteration, conditional statements, and variables using Maker.

Students focus on the 5x5 LED screen for providing output. This is the easiest way to see some kind of input. There are manyways you can get your students to see the Maker as a brain that can control physical creations.

These creations don't have to be complex. It's great to have students building with common household supplies. Because the Maker is so lightweight and supports so many sensors. It can be easily used into their design (STEAM) as long as students plan ahead for its size and weight. One of the first questions you might ask your students is "Where does the Maker fit in your creation?"

In this first lesson's project, we makesomething creative that uses the Maker as its "face." We start with a lesson on making and the nature of the Maker. It is important to set the tone for the curriculum. This is about making, building, crafting, constructing and coding (STEAM).

Some common supplies you will need to gather are:

- Pizza boxes
- Scrap cardboard
- Coloured cardboard paper
- Duct tape
- Scissors
- Pipe cleaners
- Stickers
- Feathers
- String
- Coloured markers

Designing Maker:pet

Talk to each other about yourperfect pet and take notes. Pair up with each other. One is student A, and the other is student B. The goal of this activity is to gather information from your partner that will help you to design a Maker:pet for your partner.

The first step in coding by design involves knowing what is wanted . You can create a prototype that gives you the best solution

5 minutes: Student A interviews Student B. The goal is to find out what Student B wants for their ideal pet. Student A should listen and ask questions to keep Student B talking for the 5 minutes.

Here are some questions to start with:

- Do you have a pet? What is it?
- What do you like about your pet? What do you dislike?
- Is there anything you wish your pet could do? Why?
- Tell me about your ideal pet.

5 minutes: Student B interviews Student A, as above.

Try to ask open ended questions and ask “why?” As much as possible. Your partner will tell you about his or her ideal pet, but you are really finding out more about your partner’s likes and dislikes. When we design, we create real things for real life people; so, we need to start with understanding them first.

5 minutes: Student A and Student B review their notes, and circle anything that seems as if it will be important to understanding how to create the ideal pet for their partner. Circle ideas, advice, anything that could be helpful when they start building. Then, they should use what they have discovered about their partner to fill in the blanks:

“My partner needs a .... because ....

This statement should draw some ideas about your partner’s needs based on the chat you have had with your partner.

5 minutes: Student A and Student B sketch five ideas of pets that meet their partner’s needs. Stick figures and diagrams are okay. Students shouldn’t limit themselves to real animals; unicorns and mashups are totally fine!

Make sure you keep your notes and sketches! You will use these in the project for this lesson.

Installing Maker code:

Installing the code onto the Maker.You will learn how to connect the Maker and will create a simple program using a USB cable. You will need a Maker, a micro USB cable and a computer.

IMAGE OF EQUIPMENT
IMAGE OF SIMPLE CODE TO INSTALL **Repeating faces on the Maker using the red LEDs**
SCREEN GRABS OF HOW TO DO IT **Screen grabs of the software** **How to drag code to the device**
Event handlers
SCREEN SHOT of the user interface

The Maker will hold one program at a time. You do not need to delete files off the Maker before you copy another onto the Maker; a new file will just replace the old one.

For the next project, you should attach the battery pack to the Maker using the white connector. You can build it into your design without having to connect it to the computer.

Maker:pet

This project is for you to create a Maker:pet for your partner who you spoke to. You should look atyour notes and try to find what your partner wants in a pet.Then, you should use the materials that are available to create a prototype of a pet their partner would like. The purpose of the prototype is to gather more feedback to help you in the final design.

You will need to build a Maker:pet that:

- Matches your partner's need
- Supports the Maker and its battery pack
- Allows you to easily access the Maker to turn it on and off
- Your design should use whatever materials are available to support the Maker. Make sure it's face is showing. You can be as creative and decide how to mount the board and how to decorate your pet.

IMAGES OF MADE UP MODELS

TEACHER

Make sure each project has the following specifications

Program properly downloaded to the Maker

Maker supported and the face is showing

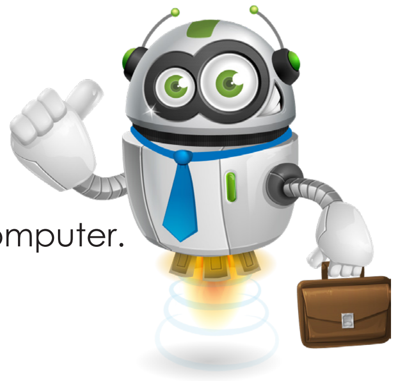
Maker can be turned on and off with taking pet apart

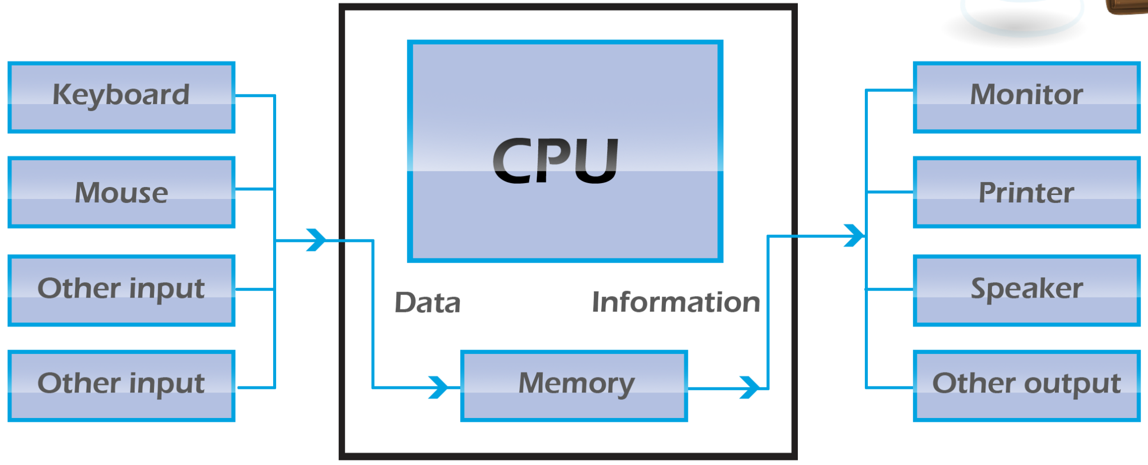
Notes about pet design included

Algorithms

What is a computer

There are 4 main components that make up any computer.





The Processor – this is usually a small chip inside the computer, and it's how the computer processes and uses information. Has anyone heard of the term "CPU"? CPU stands for Central Processing Unit. You can think of the processor as the brains of the computer - the faster the processor, the more quickly the computer can think.

The Memory – this is how the computer remembers things. There are two types of memory:

RAM – (random access memory) - you can think of this as the computer's short-term memory.

Storage – (also referred to as the "hard drive") - this is the computer's long-term memory, where it can store information even when power is turned off.

Inputs – this is how a computer takes in information from the world. In humans, our input comes in through our senses, such as our ears and eyes. What are some computer inputs? (keyboard, Mouse, Touchscreen, Camera, Microphone, Game Controller, Scanner)

Outputs – this is how a computer displays or communicates information. In humans, we communicate information by using our mouths when we talk. What are some examples of communication that don't involve talking? (blushing, sign language) What are some examples of computer outputs? (monitor/screen, headphones/speakers, printer)

Now, let's look at our Maker

IMAGE OF MAKER – front and back with labels

- Use the diagram here as a visual aid.
- Can you find the Processor?
- How much memory does the Maker have? (16K, which is smaller than many files on your computer!)
- Can you locate the following Inputs? (buttons (on board), pins (at base), accelerometer / compass)
- Though not pictured, the light sensor is located on the LED lights.
- Where are the Outputs? (LED lights, pins)
- All computers need electricity to power them. There are three ways to power your Maker:
- Through the USB port at the top
- By connecting a battery pack to the battery connector
- Through the 3V Pin at the bottom (not the recommended way to power your Maker)

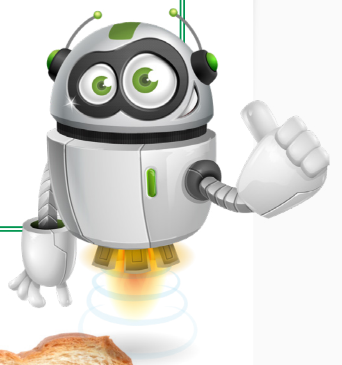
On the top left corner, you may notice that your Maker has a Bluetooth antenna. This means your Maker can talk and send information to another Maker. We will learn more about this feature in the Radio Lesson.

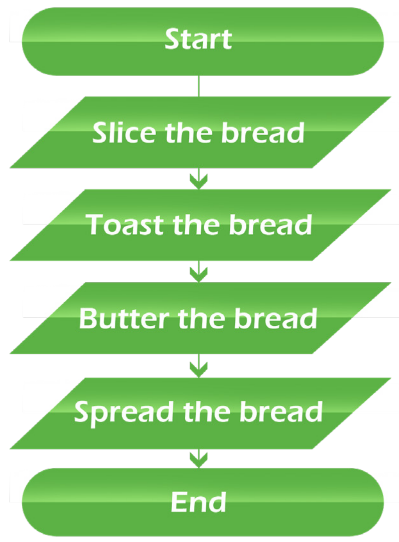
Inprogramming, algorithms are a set of instructions.Algorithms "tell" the computer how to process input and what, if any, output to produce. An example of an algorithm you might have seen in maths class is the "function machine" A function machine takes an input, processes the input, and then delivers an output. The inputs and the outputs are usually recorded in an input and output table, where the value of "x" represents the input and the value of "y" represents the output.

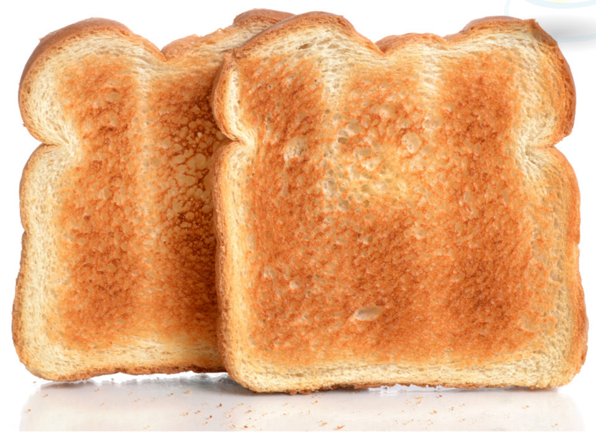
Input (x)	Output (y)
1	2
2	4
3	6
4	8

The Maker itself is hardware. It is a piece of technology. In order to make use of hardware we need to write software, (otherwise known as "code" or computer programs); the software "tells" the hardware what to do, and in what order to do it using algorithms. Algorithms are sets of computer instructions.

Ask students to write a simple algorithm to make toast in the answer box below







Explain to the student the simple flow chart for the algorithm in order to make toast.

Scan the QR code to watch you must get your instructions correct, or the computer cannot complete the task.

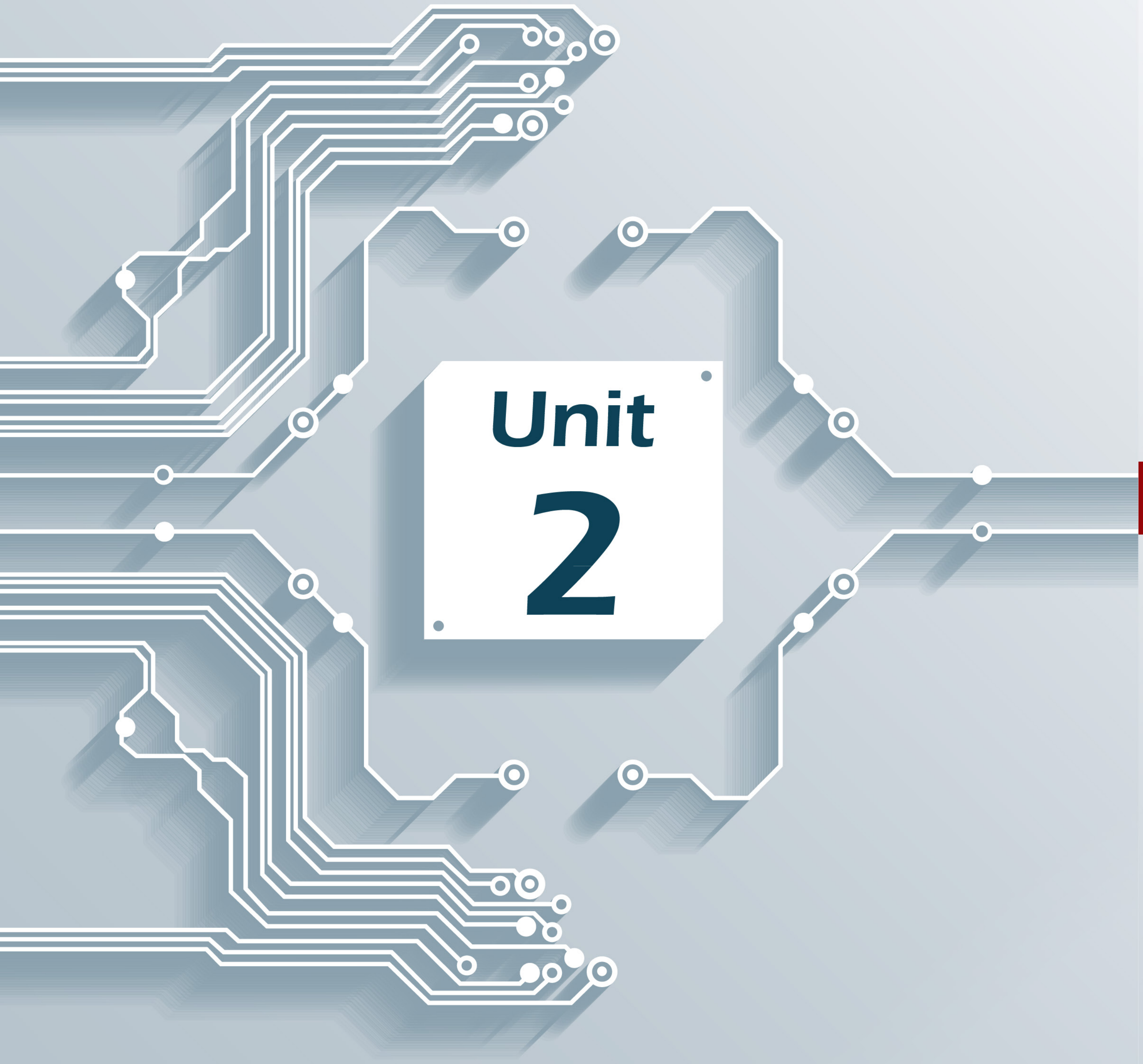
youtube link

Using the guide below, try and write an algorithm, (sequence of instructions), so that the sandwich bot will make a jam sandwich.

right hand	spread	butter	fast
left hand	scoop	tub	repeat
pick up	packet	bread	hard
press down	knife	slice	soft
cut	blade	plate	forward
put down	handle	turn	back
hold	jam	top	put
unscrew	jar	bottom	table
remove	lid	slow	surface

Always start from either right or left hand. Join the words you want to use with a line. Once you have completed the set of instructions, students to work in pairs and take turns to sit on chair in front of the teacher and give instructions – if student fails ask them to go back and try again

Write your set of instructions here:



More on maker

Learning Outcomes

2.1.Explain how Maker can show output by using LED’s

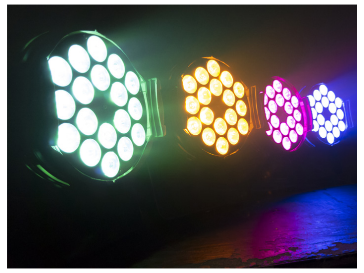
2.2.Demonstrate skills learnt by writing small programs

2.3.Apply their knowledge by creating a Maker program that takes input and produces an output

Overview

Students will be able to use conceptual framework for thinking of a computing device as something that uses code to process one or more inputs and send them to an output(s) In this activity, we will discover how to use the Maker buttons as input devices, and write code that will make something happen on the screen as output. We will also learn about pseudocode, the MakeCode tool, event handlers, and commenting code.

Keyword

Term	Definition	Image
LED	Light Emitting Diode	
program	A computer program is a collection of instructions that performs a specific task when executed by a computer.	
compile	convert the program so that it can be understood by computer	

Learning Outcomes

- Explain how Maker can show output by using LEDs.
- Demonstrate skills learnt by writing small programs.
- Apply their knowledge by creating a Maker program that takes input and produces an output.

Happy face, Sad face - using LEDs

When you start a new project, there will be start blocks. This block is an event handler.

In programming, an evert is an action done by the user, such as pressing a key or clicking a mouse button.

An event handler is a routine that responds to an event. A programmer can write code telling the computer what to do when an event occurs.

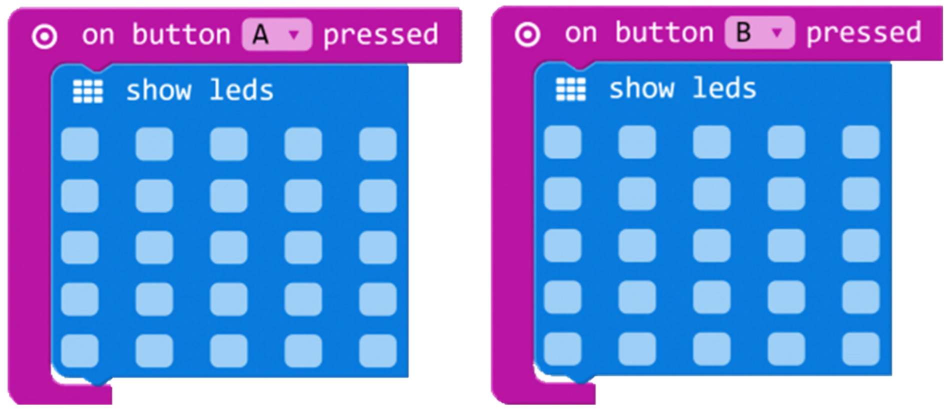
Go ahead and drag the 'clear screen' block onto the 'start' block.
As you write your code in the program, the software will automatically compile and run your code on the simulator. Before doing anything else, name the program and save it 'Happy face, sad face' press the save button to save it.

Whenever you addnew piece of code remember to save it. Giving code a meaningful name will help you find it faster from a list of programs and will let others know what your program does.

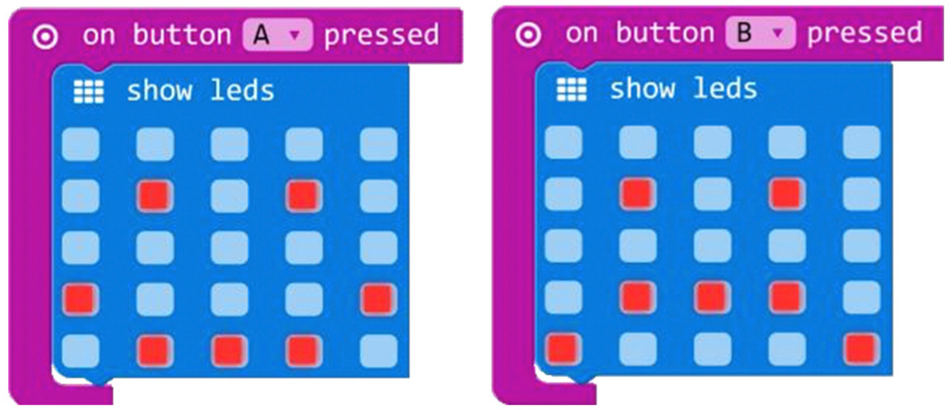
To make your program more interesting, we need to add more than twoevent handlers.

From the input menu, drag two 'on button A pressed' blocks to the coding window. Click the drop-down menu and change 'A' to 'B'

Now we can use our LED lights to display different images, depending on what button the user presses.



Now create a happy and a sad face.



Now test your code. Press, both button A and B to see the output produced by your code.

Feel free to play around with turning LEDs on and off in the 'show LEDs' blocks until you get the images you want.

Remember to save your code!

Commenting on your code

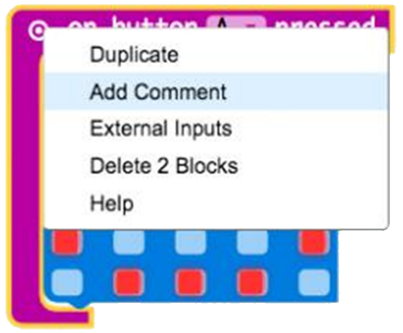
It is good practice to add comments to your code.

Comments can help you remember what a certain block of code does.

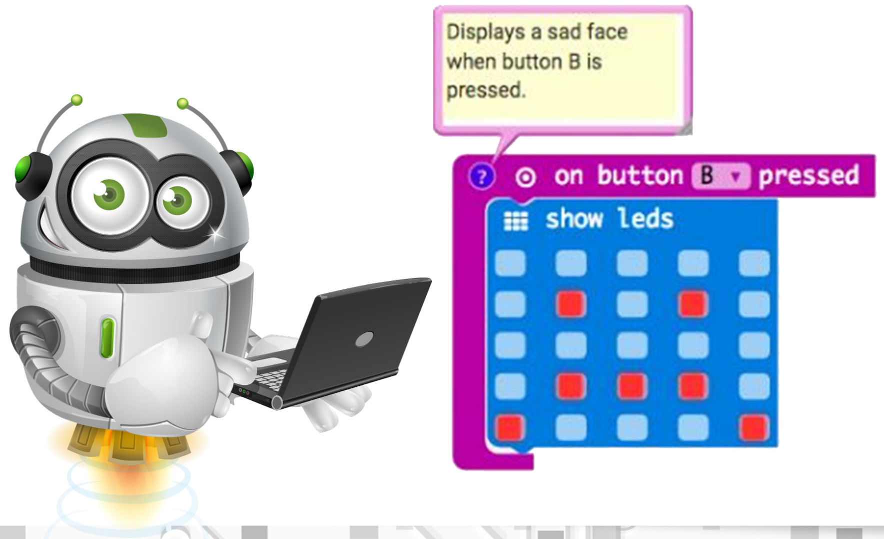
Comments also help others reading your code to understand the same things.

To comment a block of code:

- Right-click on the icon that appears before the words on a block.
- A menu will pop up. Select 'Add Comment'



- This will cause a question mark to appear to the left of the previous icon.
- Click on the question mark and a small yellow box will appear where you can write your comment.



- Click on the question make icon again to close the comment box when you are done.
- Click on the question mark icon whenever you want to see your comment again or to edit it.

In JavaScript, you can add a comment by using two forward slashes, then typing your comment. The two forward slashes tell JavaScript that the following text (on the same line) is a comment

```
// Display a happy face when button A is pressed.
```

Save and Download

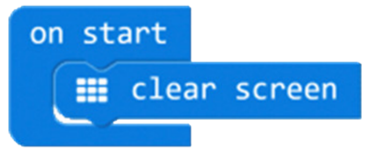
Now that your code is running and showing the happy and sad face in program, you now need to download it to Maker.

Complete program

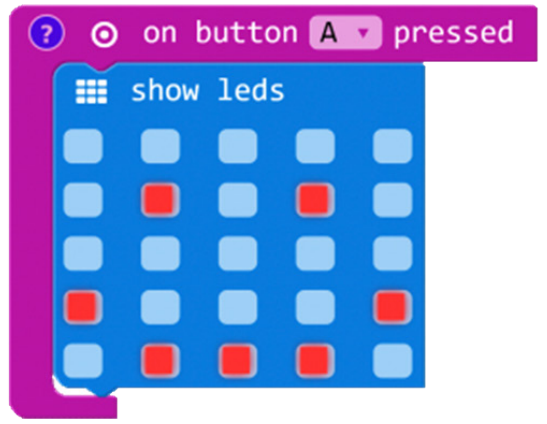
```
// Display a happy face when button A is pressed.
input.onButtonPressed(Button.A, () => {
basic.showLeds(`
 .....
.#.#.
.....
#...#
.###.
`)
})
```

```
// Display a sad face when button B is pressed.
input.onButtonPressed(Button.B, () => {
basic.showLeds(`
.....
.#.#.
.....
.###.
#...#
`) })
basic.clearScreen()
```

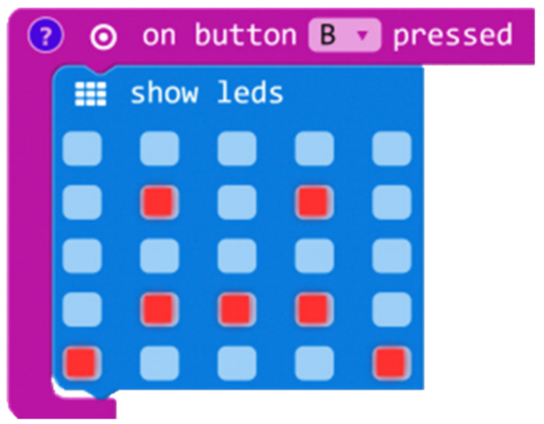
HappySadFace



on start
clear screen



on button A pressed
show leds



on button B pressed
show leds

Fidget cubes

A fidget cubes is a little cube with something different on each side.

There are buttons, switches, dials and people who like to "fidget" find it relaxing to push, pull, press and play with it.

In this project, you are going to turn the Maker into your own "fidget cube."

Remind students that a computing device has a number of inputs, and a number of outputs. The code that we write processes input by telling the Maker what to do when various events occur.

Task: Make a fidget cube out of the Maker. Create a unique output for each of the following inputs: +

- on button A pressed +
- on button B pressed +
- on button A+B pressed +
- on shake+

See if you can combine the Maker into something that can hold the cube securely when you press one of the buttons.

Challenge

Add more inputs and more outputs – use more than four different types of input. Try to use

other types of outputs (other than the LEDs) such as sound!

	4	3	2	1
Inputs	At least four inputs are successfully implemented	At least three inputs are successfully implemented	At least two inputs are successfully implemented	Fewer than two inputs are successfully implemented
Outputs	At least four outputs are successfully implemented	At least three outputs are successfully implemented	At least two outputs are successfully implemented	Fewer than two outputs are successfully implemented
Maker program	Maker program usesEvent handlers in a way that is integral to the program. Compiles and runs as intended	Maker program lacks one of the required elements	Maker program lacks two of the required elements	Maker program lacks all of the required elements

End of Unit Quiz

Statement	True or False
LED stands for Liquid Crystal Display	*False*
A computer program is a collection of instructions that performs a specific task when performed by a computer.	*True*
When you add new code, you do not need to save it.	*False*
Comments help you remember what a certain block of code does	*True*
Commenting code is not good practice.	*False*

Unit 3

Variables and Conditional Statements

Learning Outcomes

3.1. Understand what variables are and why and when to use them in a program

3.2. Learn how to create a variable, set the variable to an initial value, and change the value of the variable within the Maker program

3.3. Understand what conditional statements are, and why and when to use them in a program

3.4. Practice using the Logic blocks so different conditions yield specific outcomes

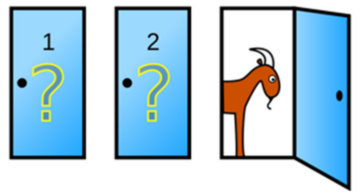
3.5. Demonstrate understanding and apply skill by collaborating with classmates to create a game that uses Maker and a program that correctly and effectively uses conditionals

Overview

Students will be able to learn how to use variables, to store data and the results of mathematical operations. Students will practice giving variables original and meaningful names. Students will be introduced to the basic mathematical operations for adding subtracting, multiplying, and dividing variables.

This lesson introduces the logic blocks such as 'If...then' and 'If...then... else'. Students will practice skills of creativity, problem-solving, and teamwork. This lesson introduces looping and iteration and presents the 'While' block as a combination of an iteration and a conditional statement.

Keyword

Term	Definition	Image
variable	temporary storage in a program to store different data types	
conditional statement	a decision-making process in programming is known as conditional statement	
logic blocks	a block which uses decision making	
programmers	people who write programs	

Learning Outcomes

- Understand what variables are and why and when to use them in a program.
- Learn how to create a variable, set the variable to an initial value, and change the value of the variable within the Maker program.
- Understand what conditional statements are, and why, and when, to use them in a program.
- Practice using the logic blocks so different conditions produce specific outcomes.
- Demonstrate understanding and applying skills by working with classmates to create a game that uses Maker and a program that correctly and effectively uses conditionals.

Introduction to Variables

Computer programs process information. It stores some information in a temporary location called variables.

Programmers createvariables to hold the value of information that may change. In a game program, a variable may be created to hold the player's current score, since that value should change during the game.

Ask the students to think of some pieces of information in their daily life that are constants and other that are variables.

What pieces of information have values that don't change during a single day (constants)?

What pieces of information have values that don't change?

....

....

The days of the week, the year, the students' names and the school address

What pieces of information have values that do change during a single day (variables)?

What pieces of information have values that do change?

....

....

The temperature/weather, the current time, whether they are standing or sitting

Keeping score: Scorekeeper

You are going to use and work with variables, pair up and play Rock, Paper and Scissors.

Ask the students to keep a score on paper, have a third of the students act asreferees and scorekeepers. After one minute ask the students to add up their scores and the number of 'rounds' they played? How have the students recorded the results? How have they recorded the ties? Ask the students what part of the score sheet represents the constants (players' names) and what part represents the variables (the players' number of wins)

This Maker activity shows you howto create a program with three variables. This will keep the score for your Rock, Paper, Scissors game.

Tell the students that they will be creating a program that will act as a scorekeeper for their next Rock, Paper, Scissors game. They will need to create variables for the parts of the scorekeeping that change over the course of a gaming session.

Q. What variables would these be?

What variables would these be?

1. ....

2. ....

3. ....

4. ....

The number of times player 1 wins

The number of times player 2 wins

The number of times there is a tie

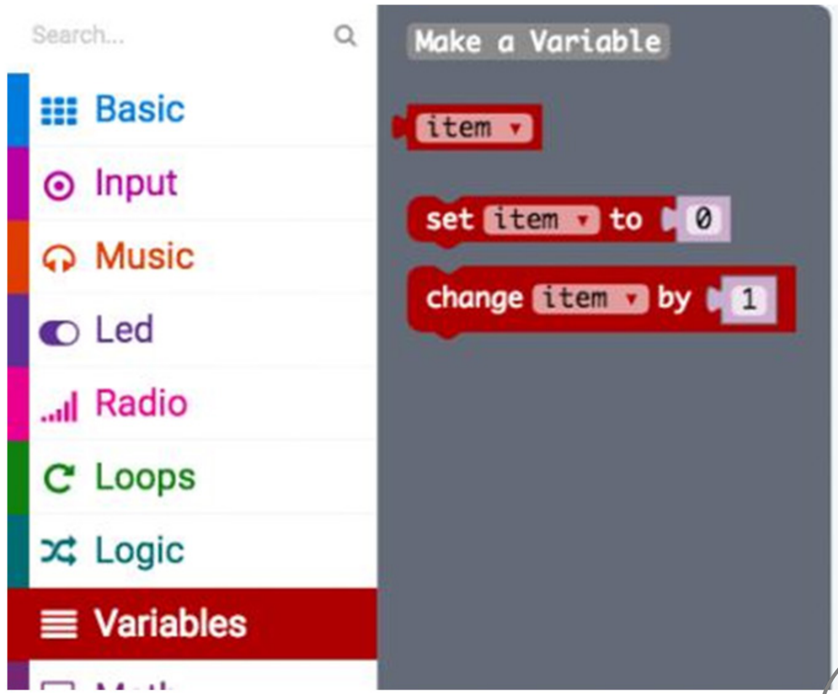
Creating and naming variables

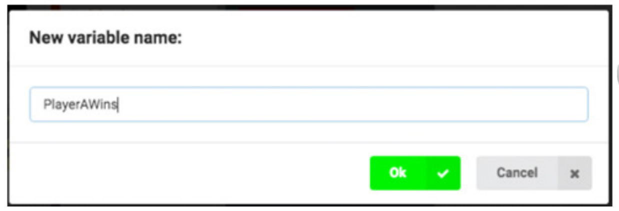
Lead the students to create meaningful names for their variables e.g. Player A wins = PAW or PlayerAwins.

Explain to the children that the variable names should clearly describe and what type of information they hold.

In Maker, from the variables menu, make and names the three variables: PlayerAwins, PlayerBwins, PlayersTie.

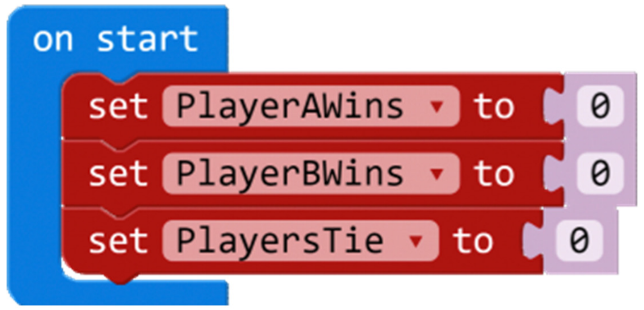
Screen shot explaining how to add variable and how to rename it





Initialising the variable menu

It is important to give your variables astarting value. The starting value is the value the variable will hold each time the program starts. Four your counter program, you will give each variable the value 0 (zero) at the start of the program.



```
on start
    set PlayerAWins to 0
    set PlayerBWins to 0
    set PlayersTie to 0
```

Updating the variable value

In your program, you want to keep track of the number of times each player wins and the number of times they tie. You can use the buttons A and B to do this.

- Press button A to record a win for player A.
- Press button B to record a win for player B.
- Press both button A and button B together to record a tie.

We have already set these variables and now need to code and to update the values at each round of the game.

- Each time the scorekeeper presses button A to record a win for Player A, we want to add 1 to the current value of the variable 'PlayerAWins.'
- Each time the scorekeeper presses button B, to record a win for Player B, we want to add 1 to the current value of the variable 'PlayerBWins.'
- Each time the scorekeeper presses both button A and button B at the same time to record a tie, we want to add 1 to the current value of the variable 'PlayersTie.'

From the input menu, drag 3 of the 'on button A pressed' event handlers to your coding window. Leave on block with 'A'. Use the drop-down menu in the block to choose 'B' for the second block and 'A+B' for the third block.

IMAGE OF THE 3 EVENT HANDLERS

From the variables menu, drag 3 of the 'change item by 1' blocks to your coding window

IMAGE OF THE 3 CHANGE ITEM BY 1 blocks

Place one change block into each of the button pressed blocks.

Choose and appropriate variable from the pull-down menus in the change blocks

IMAGES

User feedback

When the scorekeeper presses button A, button B, or both buttons together, we will give the user feedback showing that the user pressed a button. We can do this by coding our program to display:

- An 'A' each time the user presses button A to record a win for Player A

- A 'B' for each time the user presses button B to record a win for Player B
- A 'T' for each time the user presses both button A and button B together to record a tie

We can display an 'A', 'B' and 'T' using either the 'show LEDs' block or 'show string' block

An 'A' each time the user presses button A to record a win for Player A

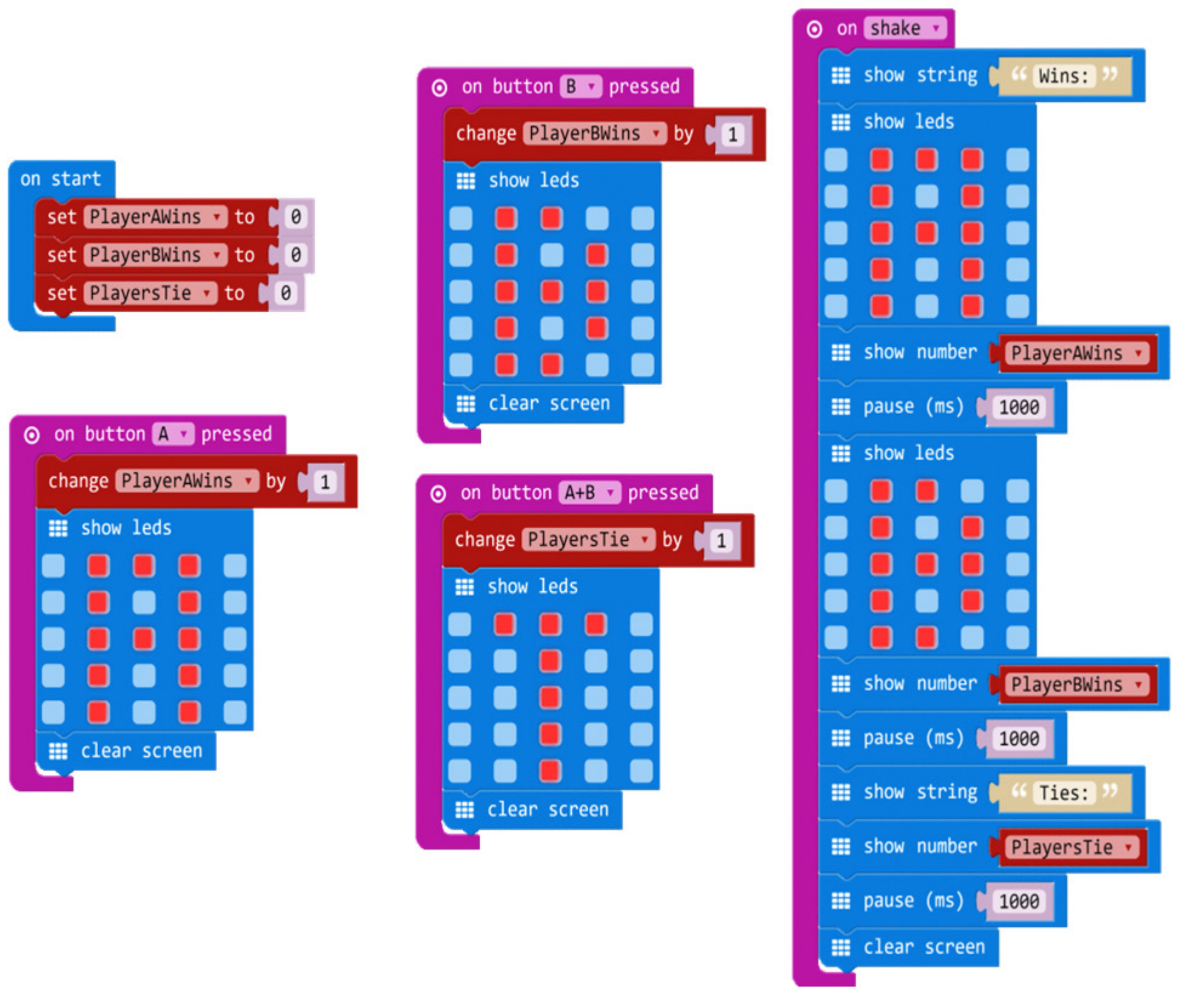
A 'B' for each time the user presses button B to record a win for Player B

A 'T' for each time the user presses both button A and button B together to record a tie

We can display an 'A', 'B' and 'T' using either the 'show LEDs' block or 'show string' block

Showing the final values of the variables

To finish our program, we can add code that tell the Maker to display the final version for the variables. Since we have already used buttons A and B we can use the 'on shake'event handler block to trigger this event. We can use the 'show string', 'show LEDs', 'pause' and 'show number' blocks to display these final versions in a clear way



```
on start
    set PlayerAWins to 0
    set PlayerBWins to 0
    set PlayersTie to 0

on button A pressed
    change PlayerAWins by 1
    show leds
        . # # # .
        . # . # .
        . # # # .
        . # . # .
        . # . # .
    clear screen

on button B pressed
    change PlayerBWins by 1
    show leds
        . # # . .
        . # . # .
        . # # # .
        . # . # .
        . # # . .
    clear screen

on button A+B pressed
    change PlayersTie by 1
    show leds
        . # # # .
        . . # . .
        . . # . .
        . . # . .
        . . # . .
    clear screen

on shake
    show string "Wins:"
    show leds
        . # # # .
        . # . # .
        . # # # .
        . # . # .
        . # . # .
    show number PlayerAWins
    pause (ms) 1000
    show leds
        . # # . .
        . # . # .
        . # # # .
        . # . # .
        . # # . .
    show number PlayerBWins
    pause (ms) 1000
    show string "Ties:"
    show number PlayersTie
    pause (ms) 1000
    clear screen
```

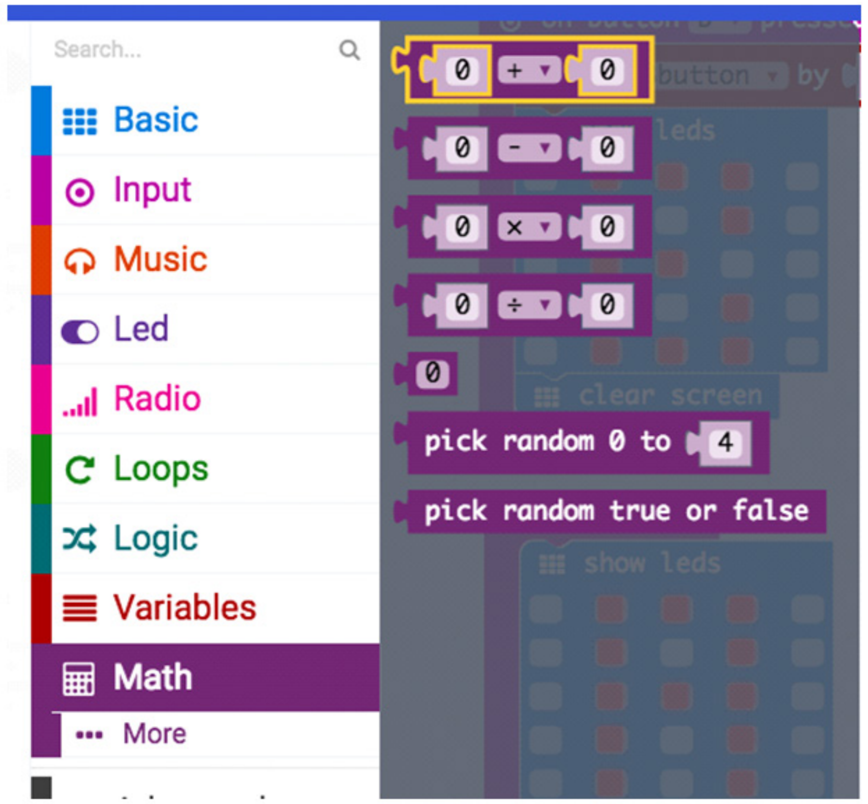
Try it out!

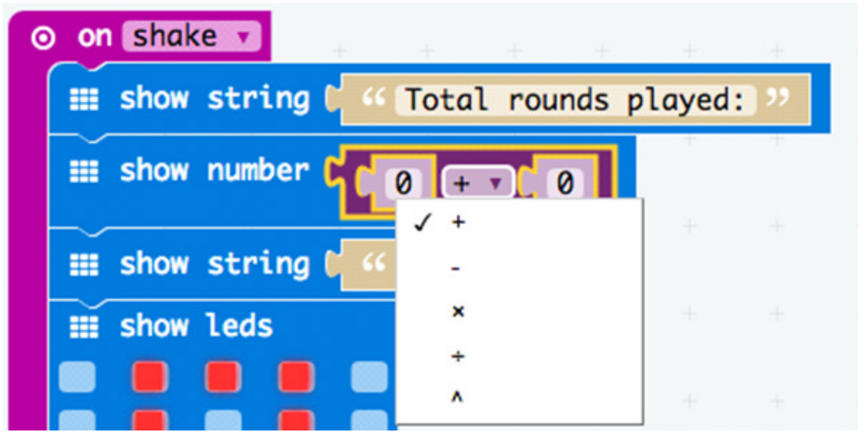
Download the Scorekeeper program to the Maker.Play one last round of Rock, Paper, Scissors using their Maker to act as the scorekeeper!

Adding on with calculations

Adding, subtracting, division and multiplication is very important in programming. When playing the game, players want to keep track of scores and this can be done by using a variable and by adding score in the variable.

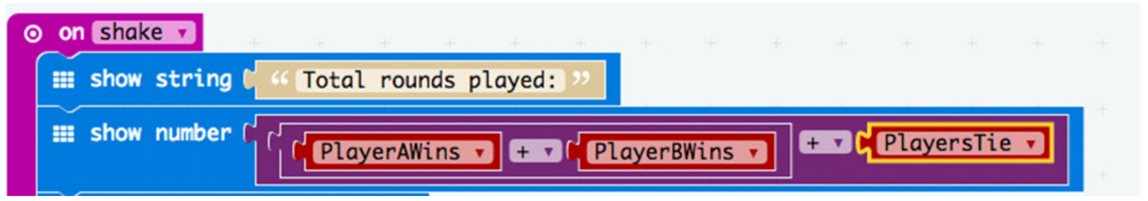
Now add the following code to the code you have already written. You can find adding, multiplying, subtracting and dividing block in Maths section of the toolbox.





```
on shake
  show string "Total rounds played:"
  show number (0 + 0)
  show string "
  show leds
```

Once you add the block, replace the zero with the name of the variable and you can add more than one variable at the same time.



```
on shake
  show string "Total rounds played:"
  show number (PlayerAWins + PlayerBWins + PlayersTie)
```

Save and download the program on Maker and test it.

What other things can you calculate apart from adding scores?

Calculate and display if player wins or lose?

Display the score as a percentage.

Calculate and display number of tied game as a percentage of all rounds.

Can you write out one of your answers using code below?

Conditional statements

Computer programs are instructions telling the computer how to process input and deliver output. An important part of programming is telling the computer "WHEN" to perform a certain task.

For this we us something called conditional statements. In real life, we make a lot of decisions every day such as when parentssay, 'if you complete your work then you can play a video game.'

Think of a situation where you had to make a decision and share your decision with other students in class.

Students to write a program to demonstrate conditional statements

Step by step guide to conditional statement program

End of Unit Quiz

Fill in the Blanks

What other things can you calculate apart from adding scores?

The decision making process in programming is known as?

What other maths calculations you have learned in this unit?

Unit 4

Iteration and looping + mini Project

Learning Outcomes

4.1. Understand the value of iteration in programming

4.2. Understand looping as a form of iteration

4.3. Learn how and when to use the looping block

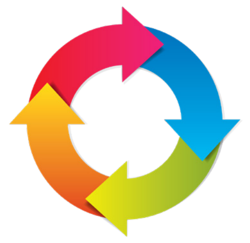
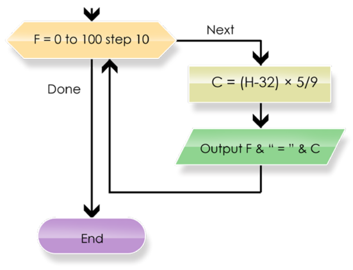
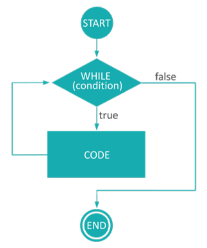
4.4. Apply the above knowledge and skills to create a unique program that uses iteration and

4.5. Understand looping as an integral part of the program

Overview

Students will be able to learn concept of looping and iteration. Presents the 'While' block as a combination of an iteration and a conditional statement. Students will be able to write programs to repeat sequence of lights. Students will be able to work through plugged and unplugged activities in order to understand the concept of iteration in programming.

Keyword

Term	Definition	Image
Iteration	process of repeating a piece of code over and over again	
For loop	a type of loop used in programming	 F = 0 to 100 step 10 Next C = (H-32) × 5/9 Done Output F & " = " & C End
While loop	also known as a conditional loop to repeat certain instruction until a condition is met	

Learning Outcomes

- Understand the value of iteration in programming.
- Understand looping as a form of iteration.
- Learn how and when to use the looping block.
- Apply the above knowledge and skills to create an original program that uses iteration and looping as an important part of the program.

Introduction

In computer programming, iterationis the repeating of a sequence of code. A loop is a form of iteration. A loop repeats code until a certain condition is met.

Questions for the students:

Do you use shampoo to wash your hair?

Most will say, 'yes'.

Have you ever read the instructions on a bottle of shampoo?

Most will say, 'no'.

Most of us have never read the instructions on a bottle of shampoo because we already know how to use shampoo. What algorithm could you write for shampooing your hair?

=Example:

1) Wet hair *2) Apply shampoo to wet hair*

3) Scrub shampoo into hair *4) Rinse shampoo out of hair*

Unplugged: walk a square

Students will give the teacher instructions to do a simple activity, then look for places where using iteration could shorten their code and make it more efficient.

Process

Place a chair in the front of the room.

Stand at the back-right side of the chair facing the students.

Ask the students what instructions they could give you, that when followed, would lead you to walk around the chair, ending up just as you started. You may want to demonstrate what this would look like by walking around the chair.

Tell the students you can only process one instruction at a time, so their algorithm needs to be step-by-step.

As students suggest instructions write them on the board or wherever everyone can see them.

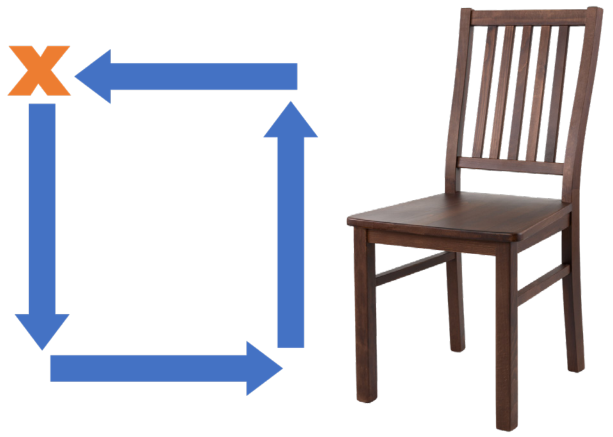
Their pseudocode will probably end up looking something like this:

1) step forward 2) turn left

3) step forward 4) turn left

5) step forward 6) turn left

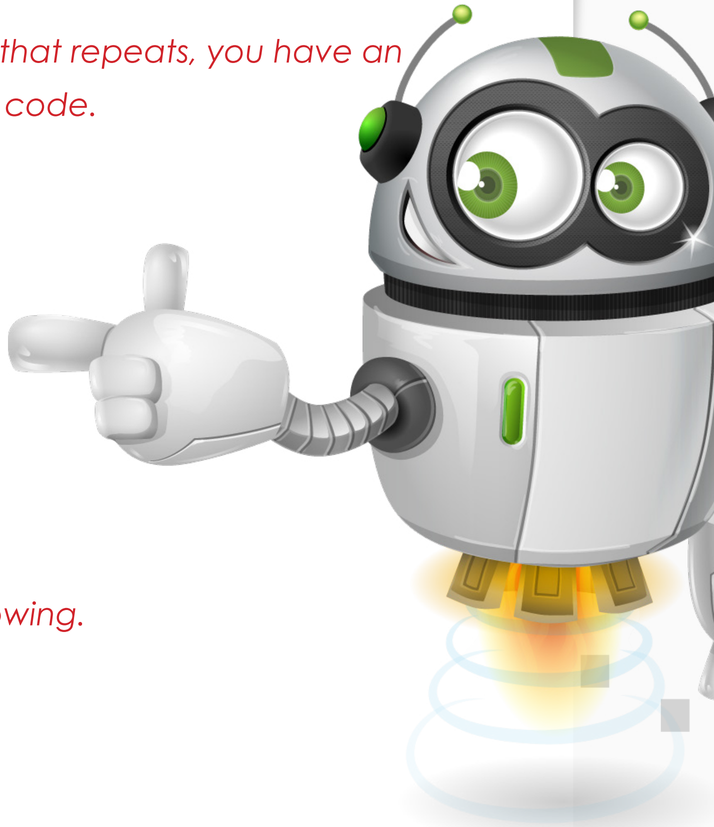
7) step forward 8) turn left



Go ahead and follow their algorithm to prove that it works.

That is just eight lines of code. Tell the students that the same lines of instructions can be written using just three lines of code. If they have not noticed already, have students look for places where the code repeats.

Tell them that whenever you have code that repeats, you have an opportunity to use a loop to simplify your code.



Prompts:

What lines are repeated?

1) Step forward. 2) Turn left.

How many times are they repeated?

Four

So how could we rewrite this code?

Students will suggest a version of the following.

Repeat 4 times:

1) Step forward

2) Turn left

Go ahead and follow the revised algorithm to prove that it works.

There! You have just rewritten eight lines of code as three lines of code by using a loop. The repeated commands create a loop. The code within a loop gets repeated a certain number of times until a condition is met.

Activity loops demo

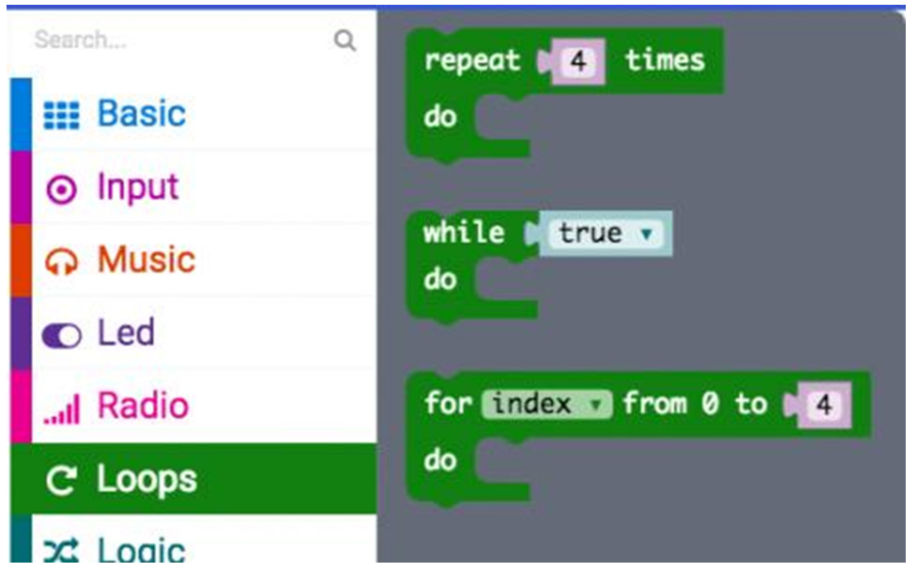
Maker has three different loop blocks:

- Repeat block
- While block
- For block

Repeat' block

Step by Step Guide

Code a sprite to walk a square. Ask students click on the loops category in the toolbox, and look at the three choices available.



```
repeat 4 times
do

while true
do

for index from 0 to 4
do
```

The very first one is the repeat block. When you drag the repeat block to the work space, you will notice that this block takes parameter.

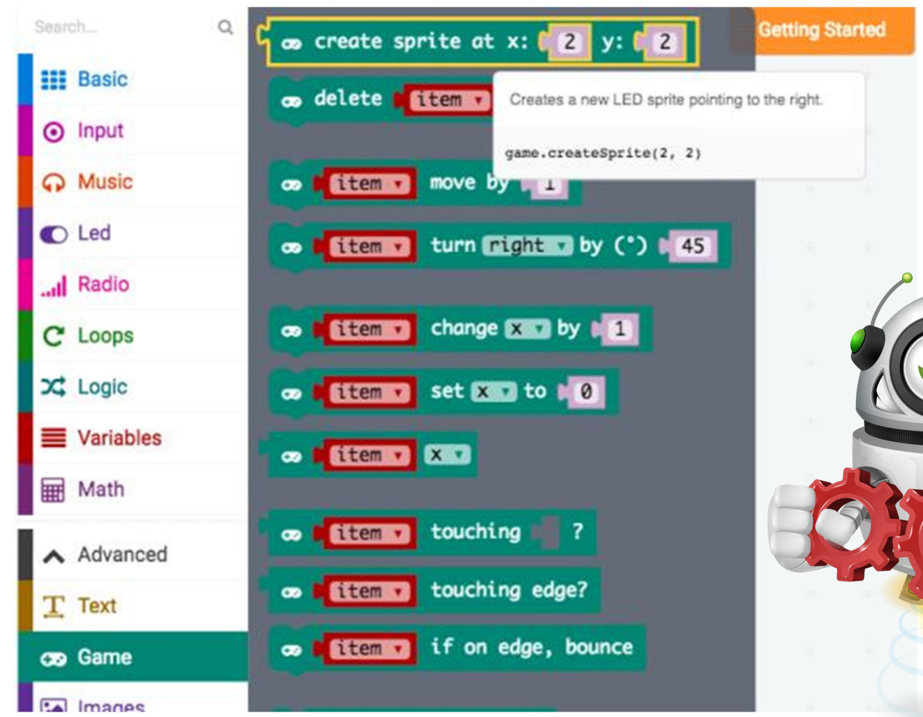
A parameteris a type of variable used as input to a function or routine. In this case, the parameter tells the repeat block how many times we want the code within the block to repeat.

For now, we will leave the parameter at 4.

To create a sprite that will walk a square:

Click on the advanced category in the toolbox. This will open up more advanced menu of blocks.

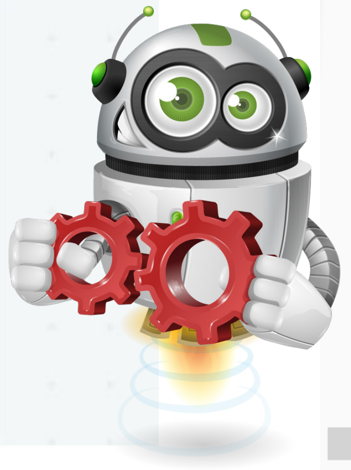
Click on the game category and drag a create a sprite block to the coding worksurface.



```
create sprite at x: 2 y: 2
delete item
item move by 1
item turn right by (°) 45
item change x by 1
item set x to 0
item x
item touching ?
item touching edge?
item if on edge, bounce
```

Creates a new LED sprite pointing to the right.

`game.createSprite(2, 2)`



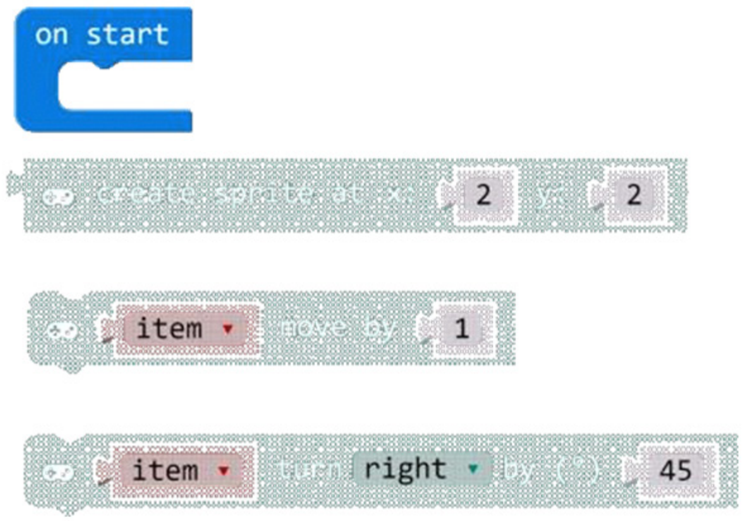
You will need to move blocks from the game menu.

Referring to the walk a square pseudocode, can you find the blocks you need for moving the sprite and turning the sprites.

Drag out the move by block and turn the right by block.

You now have these blocks in their work surface.

For this project, you can delete the for ever block.



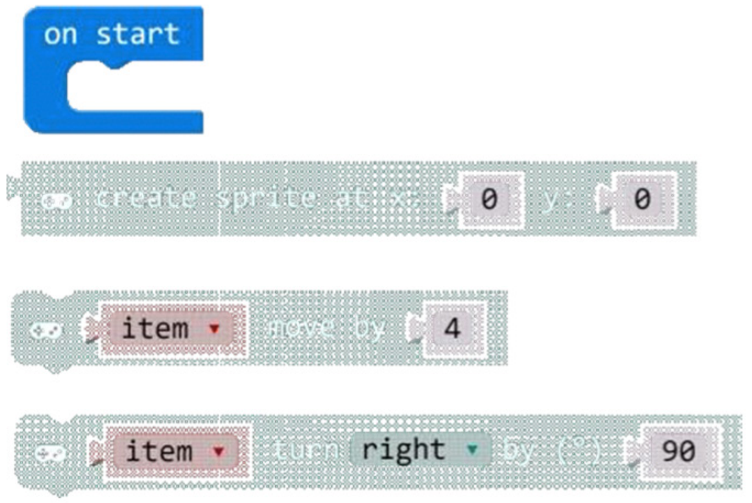
```
on start

create sprite at x: 2 y: 2

item move by 1

item turn right by (°) 45
```

Time to fix those default parameter values!

- We want your sprite to start in the top left corner of the Maker screen, so change the
  parameters for both x and y to zero.
- To make your sprite move from one side of the screen to the other (as though walking
  around a chair), change the move by parameter to 4.

To make a sprite turn to walk a square, change the turn right by degree to 90for now. It's ok to leave the sprite turning right instead of left, as we did in our pseudocode.

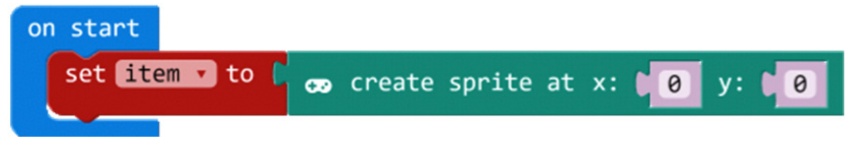
Your blocks now look like this



```
on start

create sprite at x: 0 y: 0

item move by 4

item turn right by (°) 90
```

Notice that all the blocks are greyed out. This is because we have not attached them to any event handlers.

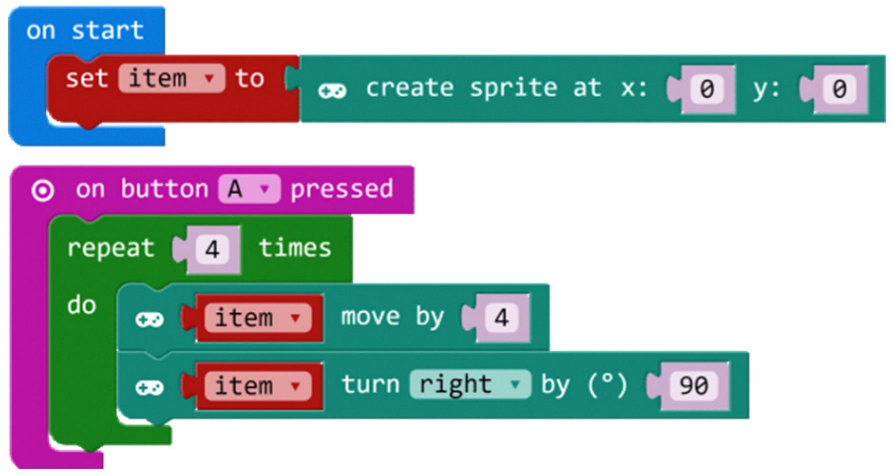
- On start we want our sprite to appear. To make this happen go to the variable menu and drag the set of item block to the coding window.
- Place the set item block into the start item block.
- Attach the create sprite to the set item block.



```
on start
    set item to create sprite at x: 0 y: 0
```

You should now see the sprite appear in the top left of the Maker simulator.

- To add more control when sprite moves, drag the on button A pressed block from the input menu
- Place the repeat block into the on button A pressed block
- Place the move by block into the repeat block
- Place the turn right by block into the repeat block just under the move by block



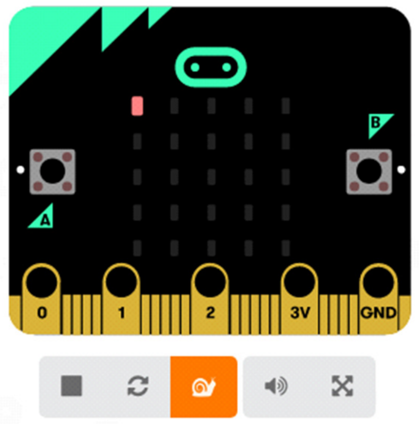
```
on start
    set item to create sprite at x: 0 y: 0

on button A pressed
    repeat 4 times
    do
        item move by 4
        item turn right by (°) 90
```

Go ahead and run the program. Make the sprite move by pressing button A.

What happened? Did you see the sprite move? No?

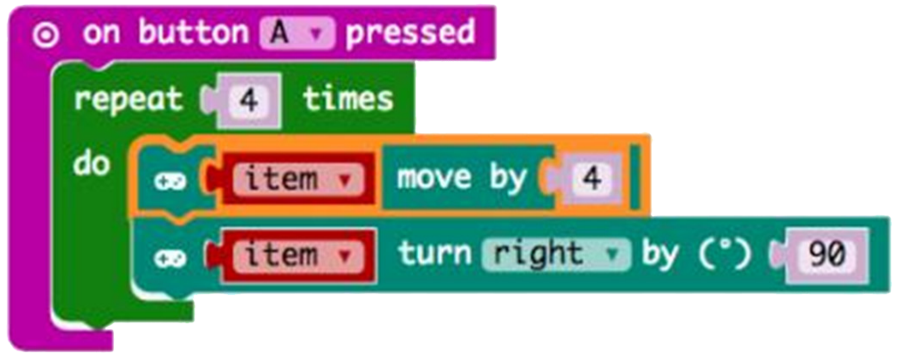
Slow-Mo

A helpful feature of Microsoft Make Code is 'Slow-Mo', or slow-motion mode. • Click on the snail icon under the Maker simulator.



This will slow down the execution (running) of the program, and highlights parts of your code so you can see step-by-step, which line of code is being processed.

Now run your program several more times. Do you see the different lines of your code highlighted as the program runs? Do you see the sprite move?



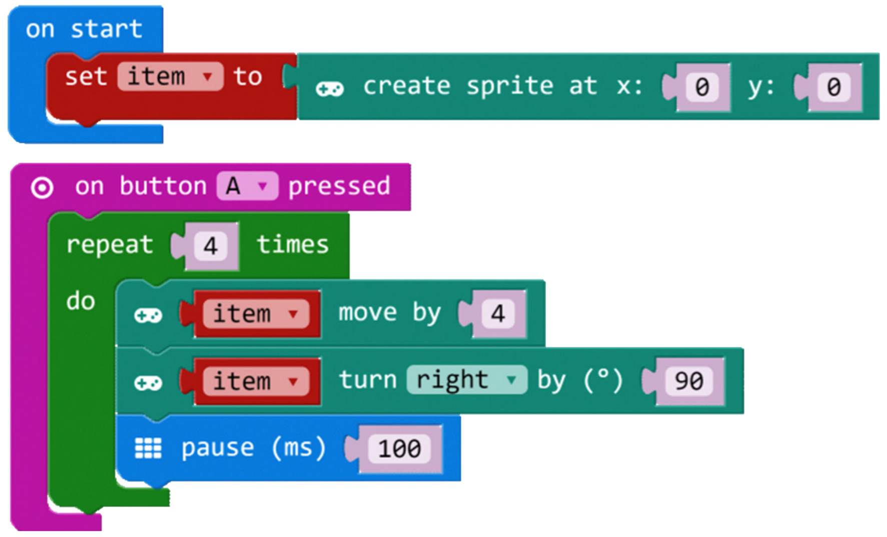
```
on button A pressed
  repeat 4 times
  do
    item move by 4
    item turn right by (°) 90
```

So, the code is running and the sprite is moving! Sometimes we forget just how fast computers are, so that we can see the sprite move even in regular mode. Let's add a pause to our program, right after each time the sprite moves. This will give the human eye a chance to see it move.

Step by Step Guide

- Click the snail icon again to turn off Slow-Mo.
- From a basic toolbar category, drag a pause block of coding window and add to our repeat block right after the turn right by block.

The final block of code should look like:



```
on start
  set item to create sprite at x: 0 y: 0

on button A pressed
  repeat 4 times
  do
    item move by 4
    item turn right by (°) 90
    pause (ms) 100
```

Run your program again. Now we can see the sprite move. It still moves pretty quickly, but at least we can see it move.

Now try changing the parameters to see how these changes effect their program. Write the new parameter you tested and what it does?

'For' block: Traveling Light

The for block is useful when you have a variable in your loop that you want to change by a fixed amount. Let's look at the example.

Let's look at the LED light move across the display from left to right, top row to bottom row.

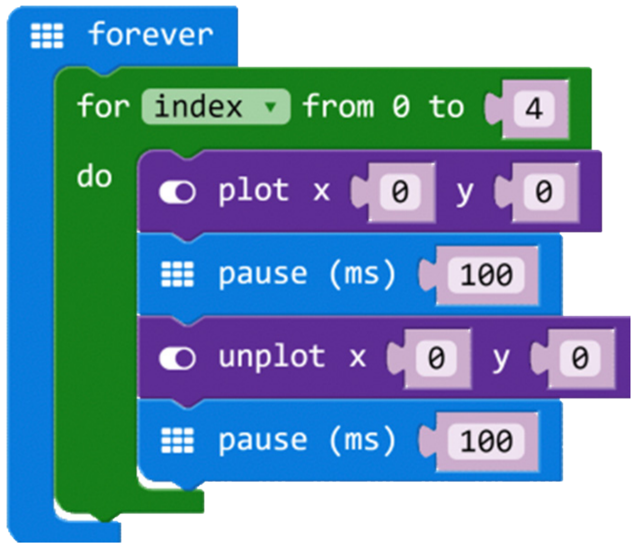
Our pseudocode for the first row might look like this:

```
Turn led x:0, y:0 on
Pause
Turn led x:0, y:0 off
Pause
Turn led x:1, y:0 on
Pause
Turn led x:1, y:0 off
Pause
Turn led x:2, y:0 on
```

```
Pause
Turn led x:2, y:0 off
Pause
Turn led x:3, y:0 on
Pause
Turn led x:3, y:0 off
Pause
Turn led x:4, y:0 on
Pause
Turn led x:4, y:0 off
```

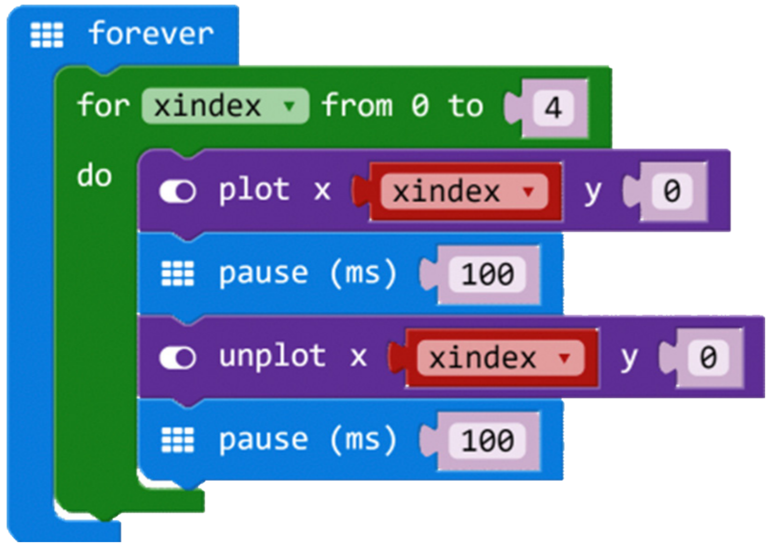
Step by Step Guide

- From the loop toolbar drawer drag a for loop block to the coding surface.
- Since we will be changing the value of x coordinates make a new variable name it as xindex.
- We will plot and then unplot LEDs to turn them on and off.
- From the LED toolbar drawer, drag a plot block and unplot block to the coding surface.
- From the basic toolbar. Drag two pause blocks to the coding workspace.
- Place the following block in to the for block, the plot block, a pause block, the unplot block, the second pause block.
- Place the for block inside the forever block.



```
forever
    for index from 0 to 4
    do
        plot x 0 y 0
        pause (ms) 100
        unplot x 0 y 0
        pause (ms) 100
```

- Let's look at the parameter
- Change the index variable to xindex we created
- Change the value of the x coordinates in the plot and unplot blocks to the same variable.

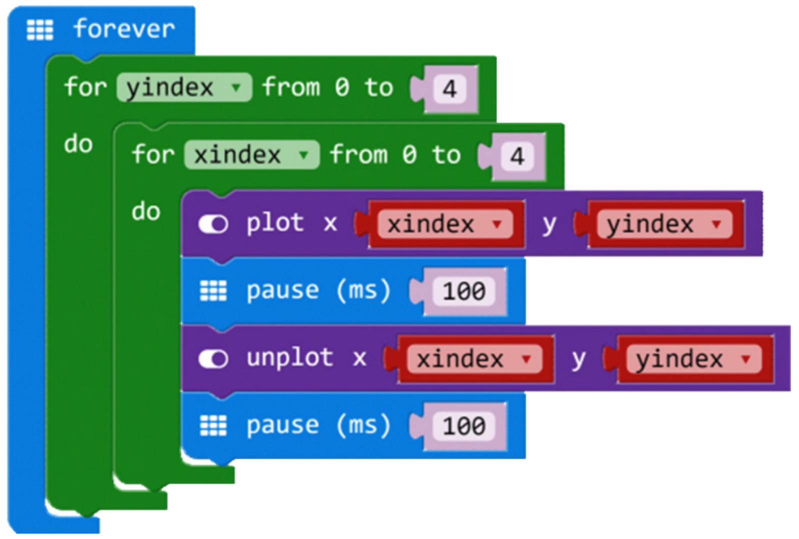


```
forever
    for xindex from 0 to 4
    do
        plot x xindex y 0
        pause (ms) 100
        unplot x xindex y 0
        pause (ms) 100
```

You should now see a light moving from left to right along the top row of the Maker simulator.

Now to make the pattern continue, we need to change the y coordinates as well. To do this follow the steps below.

- So that we can change the value of the y coordinate, make a new variable, named yindex.
- Drag another for block from the loops toolbox.
- Place this new for block around the previous for block. All within the forever block.
- Change the index in the outer for block to a yindex variable we made.
- Change the value of the y coordinates in the plot and unplot blocks to this same variable.



```
forever
    for yindex from 0 to 4
    do
        for xindex from 0 to 4
        do
            plot x xindex y yindex
            pause (ms) 100
            unplot x xindex y yindex
            pause (ms) 100
```

Now experiment with changing the parameters to see how these changes effectof program. Write your changes below.

What happens if you switch the positions of the nested loops, so the outer loop loops through the xindex values and the inner loop loops through the yindex values?

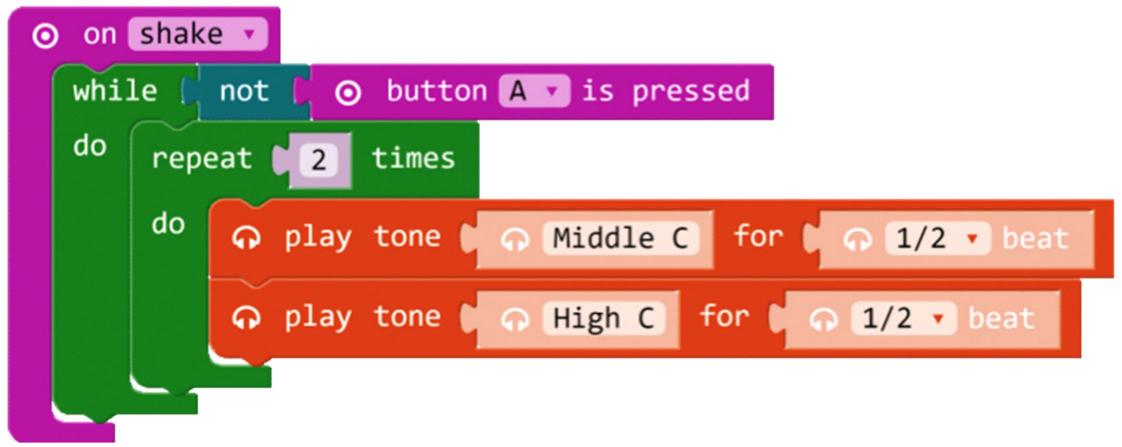
What happens if you remove the unplot block and the pause block below it?

While block

The while block is useful when you want your program to loop until a certain event happens or a different condition is met.

For example, maybe you want an alarm to sound if someone shakes your Maker!
In order to turn the alarm off, you press the button A. Until you press the button, the alarm should continue to sound!

You can use the while block with a nested repeat block like this,



```
on shake
  while not button A is pressed
  do
    repeat 2 times
    do
      play tone Middle C for 1/2 beat
      play tone High C for 1/2 beat
```

Can you read what this code does?

Can you write out pseudocode that describes what this code does?

Example Pseudocode:
When someone shakes the maker, while button A is not pressed, play the two-tone alarm twice. Keep playing the alarm tones until the user presses the A button.

Code the above program and test it using speaker.

Project to get the loop to be added. Some example project ideas:

- Create an animated gif (looping image that changes) and add music that matches.
- Create animation that repeats for one of the melodies included in Make Code (like Happy Birthday).
- Create different animations that run when different buttons are pressed.
- Create an alarm that includes sound and images. What will set the alarm off? What will make the alarm stop sounding?
- Use servo motors to create a creature that dances and changes its expression while a song plays.

End of unit Assessment

What are the three types of loops learned in this unit?

1. ....
2. ....
3. ....

Match the word with the correct definition

Word	Definition
Iteration	a type of variable used as input to a function or routine
For Loop	iteration that is the repetition of a sequence of code
While Loop	This block is useful when you have a variable in your loop that you want to change by a fixed amount.
Parameter	This block is useful when you want your program to loop until a certain event happens or a different condition is met.

Word	*Definition*
Parameter	*a type of variable used as input to a function or routine*
Iteration	*iteration that is the repetition of a sequence of code*
For loop	*This block is useful when you have a variable in your loop that you want to change by a fixed amount.*
While Loop	*This block is useful when you want your program to loop until a certain event happens or a different condition is met.*

Boolean, Bits, Bytes and Binary (coordinate Grid system)

Learning Outcomes

5.1. Understand what Booleans and Boolean operators are, and why and when to use them in a program

5.2. Learn how to create a Boolean, set the Boolean to an initial value, and change the value of the Boolean within a Maker program

5.3. Learn how to use the random true or false block

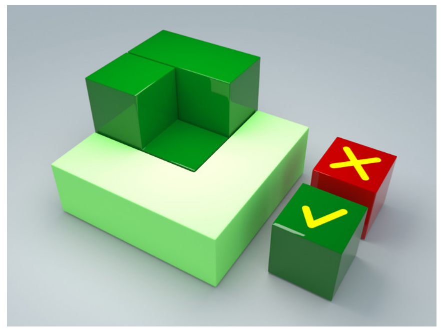
5.4. Apply the above knowledge and skills to create a unique program that uses Booleans and Boolean operators as an integral part of the program

Overview

This lesson introduces the use of the Boolean data type to control the flow of a program, keep track of a state, and to include or exclude certain conditions.

This lesson presents the idea of binary digits and base-2 notation. Students will learn how data is stored digitally and how it can be read and accessed.

Keyword

Term	Definition	Images
Boolean	a data type which has two values; true or false	
False Block	A block when a certain condition is not met	
True block	A block when a condition is met	

Learning Outcomes

- Understand what Booleans and Boolean operators are, and why and when to use them in a program.
- Learn how to create a Boolean, set the Boolean to an initial value, and change the value of the Boolean within a Maker program.
- Learn how to use the random true or false block.
- Understand what a bit and byte are and how they relate to computers and the way information is processed and stored.
- Learn to count in Base-2 (binary) and translate numbers from binary to Base-10 (decimal).

Introduction to Boolean

There are several different data types used in computer programming. We have already used two types.

- Integer
- String

Boolean is another type of data. A Boolean data type has only two values: true or false.

Booleans are useful in programming for making decisions. Whenever you play a computer game you get to choose different options; for example, one player, two players, different modes, etc. these are all programmed using Boolean logic.

Think of a game you like the most. Write down some choices you get in that game.

Write down what you usually use?

Example options

Different modes available in games

Different players available in games

Different backgrounds

Playing online

Different themes

Boolean Operators AND, OR and NOT

In programming, if you have worked with conditionals or loops, you have already worked with this type of logic:

- If a certain condition is true, do this, otherwise (if a condition is false), do something else.
- While a certain condition is true, do this:

There are different types of logics used in computer programming: AND, OR and NOT. All the circuit board are designed using these logics.

AND

For this logic both the conditions have to be true. For example, you cannot move onto level 2 if you have not collected 10 pieces of fruit AND jumped over all the obstacles.

Can you give an example of AND logic from the game you play?

OR

For this type of logic only one condition has to be true; for example, you can go to level 2 if you have collected 10 pieces of fruit OR jumped over all the obstacles.

Can you give an example of OR logic from the game you play?

NOT

NOT can be used when checking that a condition is false (or not true).

NOT is also useful when using a loop; for example, while the correct name is NOT entered, continue to ask for the name.

Activity 1

Working in pairs, make a table or list of the possible outcomes if each student flipped a coin at the same time. Also write down what type of Logic will apply.

Coin A	Type of Logic	Coin B	Totals
Head		Head	
Head		Tail	
Tails		Head	
Tail		Tail	

Coin A	Type of Logic	Coin B	Totals
Head	*AND*	Head	*1*
Head	*OR*	Tail	*2*
Tails	*IR*	Head	*2*
Tail	*AND*	Tail	*1*

Double coin flip program

In this program students will write a code to flip two coins.

Step by Step Guide

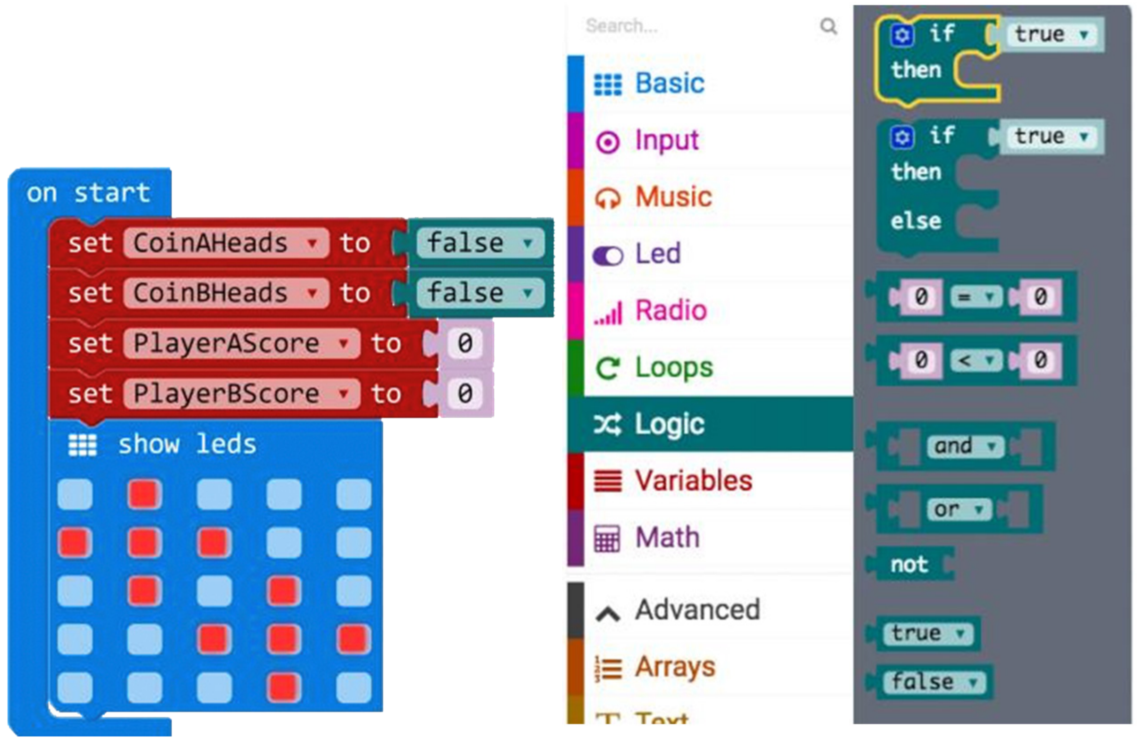
Let's start by creating following variables.

- CoinAHeads
- CoinBHeads
- PlayerAScore
- PlayerBScore

Now we need to initialise the variable values. Put a 'set' variable block for each of these four variables inside the 'on start' block.

The initial value of a variable is the value the variable will hold each time the program starts. By default:

- A string variable is initialised to an empty string “”
- A number variable is initialised to 0
- A Boolean variable is initialised to “false”



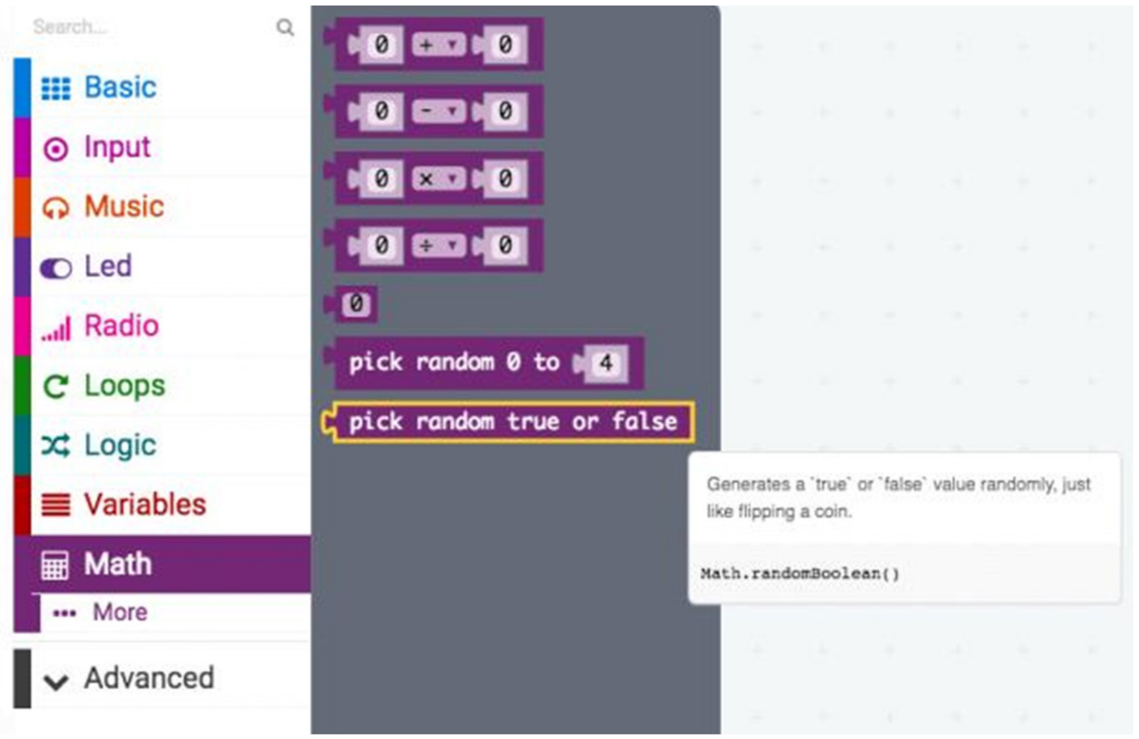
```
on start
    set CoinAHeads to false
    set CoinBHeads to false
    set PlayerAScore to 0
    set PlayerBScore to 0
    show leds
        . # . . .
        # # # . .
        . # . # .
        . . # # #
        . . . # .
```

Notice that we also added an image for the start screen, so the user knows the program has started and is ready. Does the image look like two coins?

Random coin flips

When the player shakes the Maker, we will code the Maker to give each of our Boolean variables a random true/ false value.

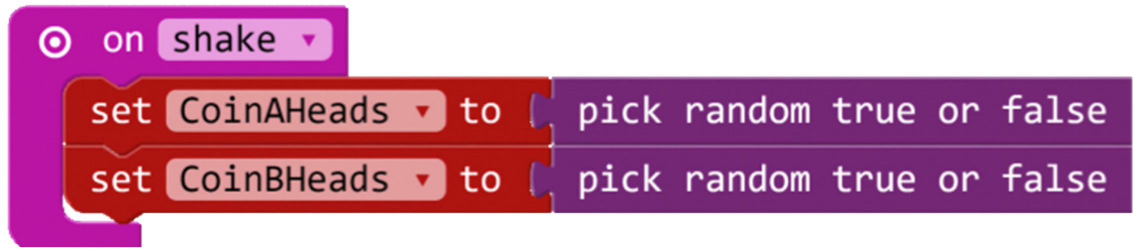
- From the input toolbox, drag the on-shake block to the coding surface
- From variable toolbox, drag 2 set variable bocks
- Drag the 2 set blocks onto on-shake block
- Change the default name to CoinAHead and CoinBHead
- From the Maths toolbox drag 2 random true or false block



Generates a 'true' or 'false' value randomly, just like flipping a coin.

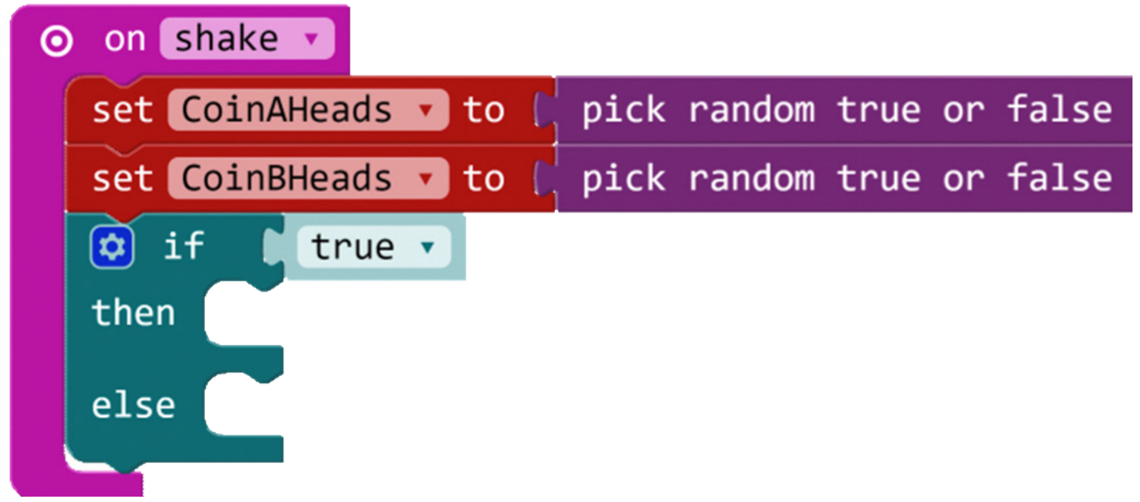
```
Math.randomBoolean()
```

- Attach these 'pick random' blocks to the 'set' variable blocks in the 'on shake' block

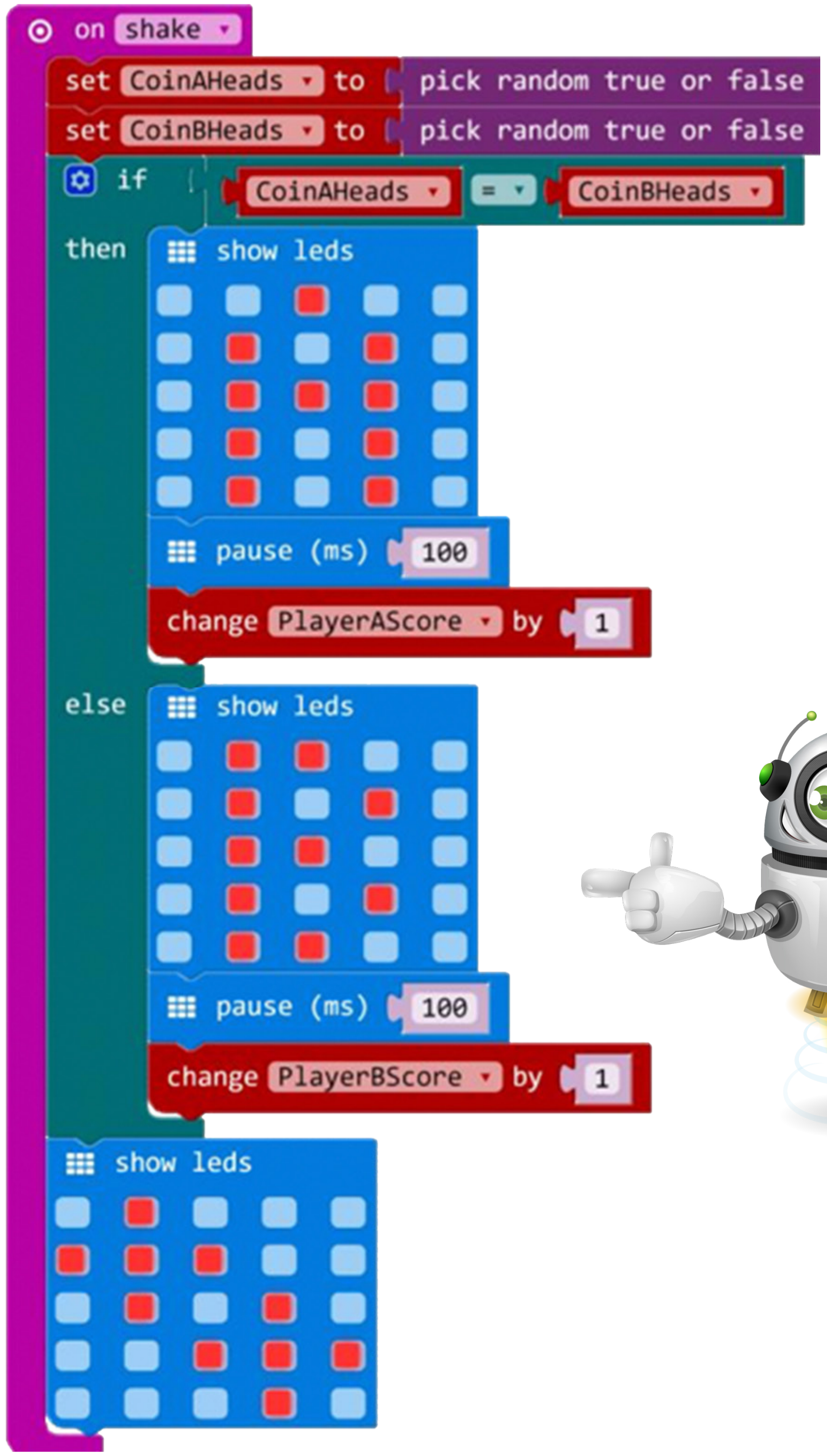

```
on shake
  set CoinAHeads to pick random true or false
  set CoinBHeads to pick random true or false
```

Now that the virtual CoinA and CoinB have been virtually flipped, we need to compare the outcomes to see if they are the same or different.

- From the Logic Toolbox drawer, drag an 'if...then...else' block to the coding workspace
- Drag the 'if...then...else' block into the 'on shake' block under the 'set' variable blocks

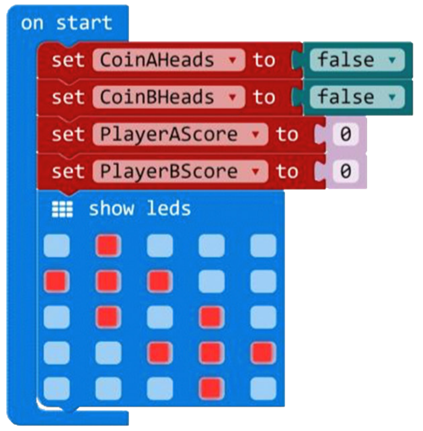

```
on shake
  set CoinAHeads to pick random true or false
  set CoinBHeads to pick random true or false
  if true
  then
  else
```

- We can now simply add this to our current code
- And provide user feedback by adding some visuals

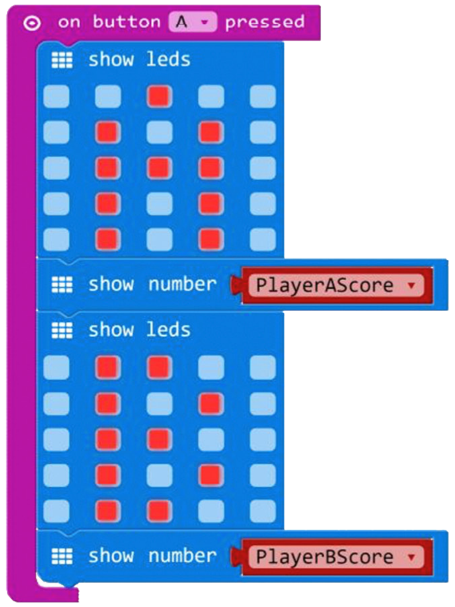


```
on shake
    set CoinAHeads to pick random true or false
    set CoinBHeads to pick random true or false
    if CoinAHeads = CoinBHeads
    then
        show leds
        pause (ms) 100
        change PlayerAScore by 1
    else
        show leds
        pause (ms) 100
        change PlayerBScore by 1
    show leds
```

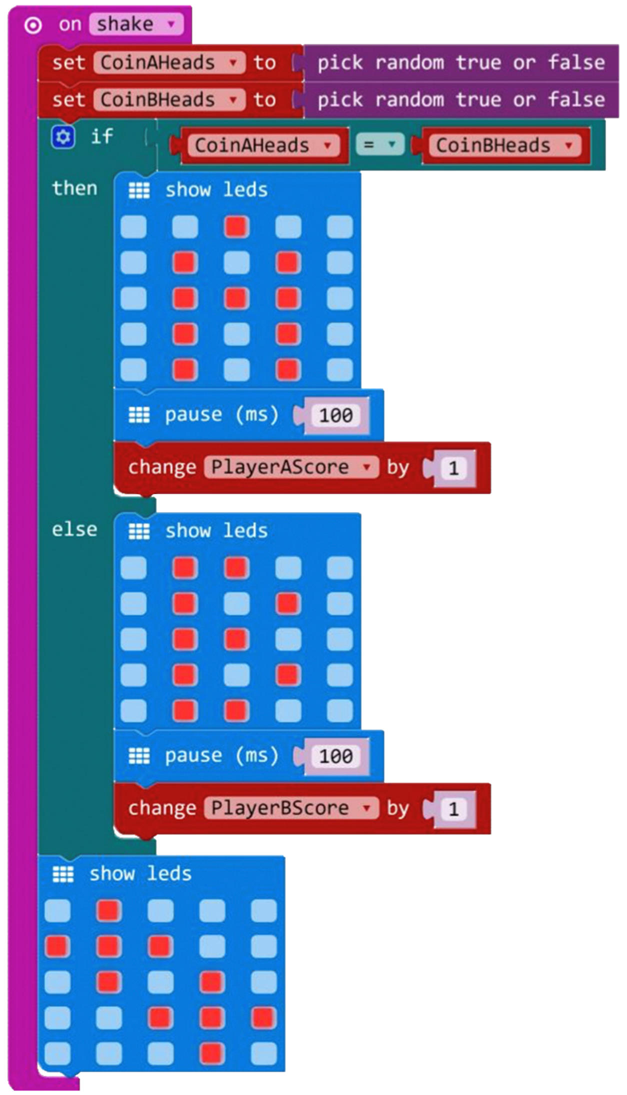
Here is the complete program.



```
on start
    set CoinAHeads to false
    set CoinBHeads to false
    set PlayerAScore to 0
    set PlayerBScore to 0
    show leds
```



```
on button A pressed
    show leds
    show number PlayerAScore
    show leds
    show number PlayerBScore
```



```
on shake
    set CoinAHeads to pick random true or false
    set CoinBHeads to pick random true or false
    if CoinAHeads = CoinBHeads
    then
        show leds
        pause (ms) 100
        change PlayerAScore by 1
    else
        show leds
        pause (ms) 100
        change PlayerBScore by 1
    show leds
```

Bits, bytes and binary

As we use different units for measuring objects and distance, we use kilobytes, megabytes, gigabytes and terabytes to measure the size of the files and hard disks. We also use these units to look at a download and upload speed, size of an image and other things related to computers.

A bit in binary has only two possible values, zero or one. A power switch is a good example as it can only be on or off.

A byte is a sequence of binary digits made up of 8 bits. A byte can represent 256 possible values.

Binary to decimal conversion

Computers can only understand binary. Computers are made up of thousands of transistors attached to a circuit board. We are going to see how a denary number can be changed into a binary number.

Example:

1100 is a binary number equitant to 12

8	4	2	1
1	1	0	0

This is converted using a formula. The numbers in row 1 have two rules:

- Starts from left
- Doubles each time

The numbers in row two are binary numbers. To convert this binary number to a denary number we do the following:

Wherever the number 1 is in row 2, we take down the number from row 1; for example:

- We will take 8 and 4 from the table above as they both have 1s in them

- We will ignore 2 and 1 as they have 0s below them.
- We will now add 8+4 = 12

Activity 2: {

Convert the following binary numbers in to denary numbers

1111

8	4	2	1

Answer is 15

1010

8	4	2	1

Answer is 10

0011

8	4	2	1

Answer is 3

1000

8	4	2	1

Answer is 8

End of unit Quiz

Statement	True or False
Boolean is a data type in programming?	True
Boolean logic can be True or False?	True
1 means False and 0 Means True	False
For AND logic only one condition has to be true?	False
Computers understand denary numbers	False
0001 is 11 in denary?	False
1011 is 11 in denary?	True

Binary Conversion Project

Learning Outcomes

6.1. Apply the skills learnt in this unit by creating project

6.2. Demonstrate a good understanding of variables, conditional statements, iteration and Boolean by applying these skills in the final project

Overview

The final project is a chance for you to use all the skills you have learned throughout the semester to create something that is original, and that solves a problem or serves a purpose.

Keywords

Term	Definition	Image
Binary	0 and 1 represents off and on	
Maker	device with LEDs	

Learning outcomes

- Analyse the given problem
- Demonstrate the skills learned to convert binary to decimal
- Display the conversion using LED lights

Binary conversion using maker

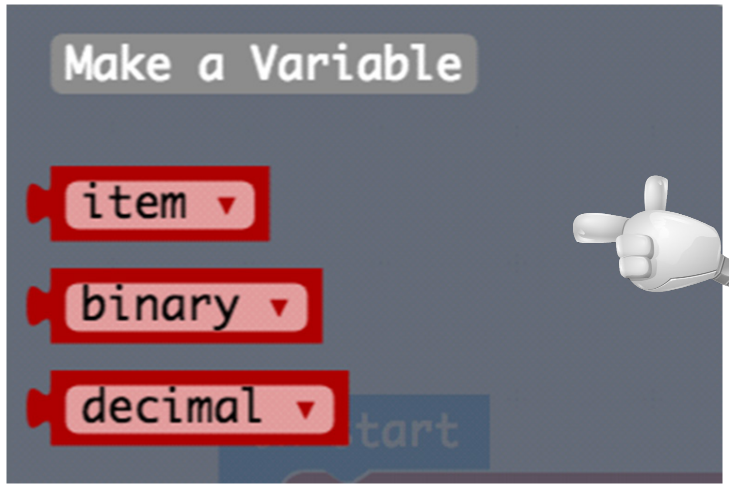
Binary converter changes between binary and decimal numbers. Binary numbers are '0' and '1.' Decimal numbers are base 10 numbers so (0 – 9).

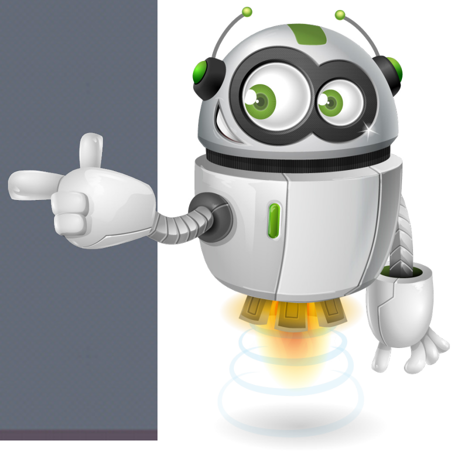
We will be using buttons on maker to get input. Button A will be used for 0 and B will be used for 1. If you press both A and B together, it will give you the answer in decimal by flashing lights.

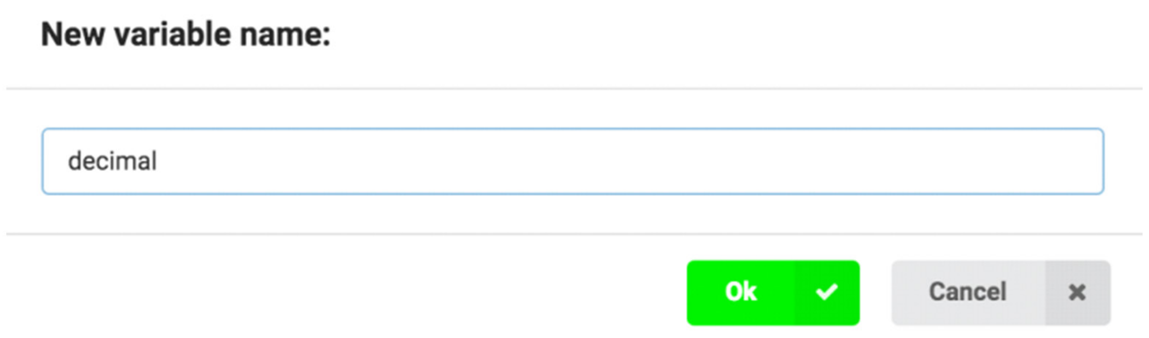
We will look at basics of binary conversion. Follow the step-by-step guide to do basic conversion.

Step -by-step guide

- We will start by creating variables to hold binary and decimal numbers.






New variable name:

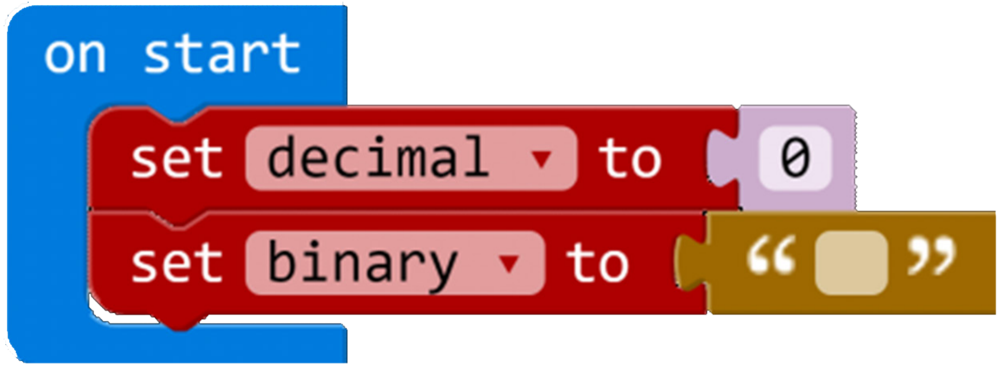
decimal

We will now initialise the variable by giving them a starting value.

decimal = 0

Binary = “” (empty string)

This also tells the micro:bit what type of variable it is. Use the empty string value found in the text toolbox drawer, under the advanced menu.



```
on start
    set decimal to 0
    set binary to “ ”
```

We are going to use a simple method of calculating the decimal number. Every time the user presses a 1 or a 0, it calculates the current decimal value of that string. You only need to deal with one 0 or 1 at a time.

Before we start coding, we are going to start with pseudocode. We have previously looked at pseudocode, which is instructions in plain English. Your pseudocode might be different than the one shown below.

Activity 1

Write the pseudocode to convert a binary number to a decimal number. Use the hints below.

When Button A is pressed

When button B is pressed

When button A+B is pressed

When button A is pressed;

Add “1” to the end of binary string

Show the current value of binary string

Update the current decimal value with the total

When button B is pressed;

Add “0” to the end of binary string

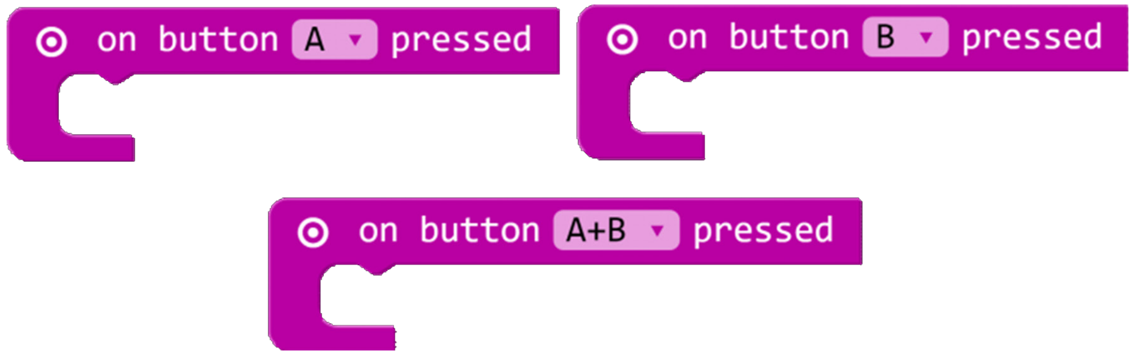
Show the current value of binary string

Update the current decimal value with the total

When button A+B is pressed;

Show the current value of the decimal string.

- Once you have planned the solution you need to then code it using Maker. Here are some blocks which can help you.

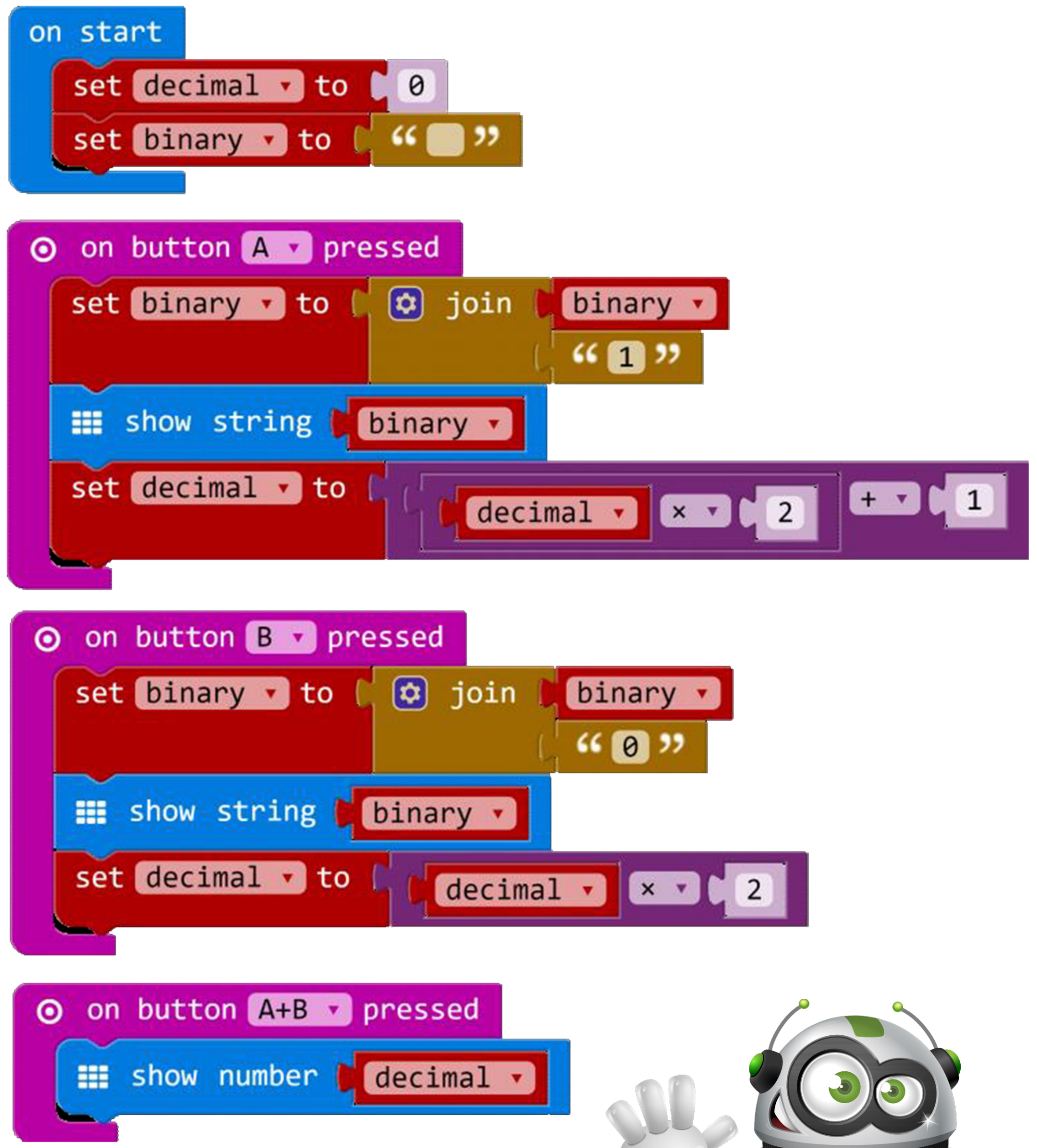


```
on button A pressed

on button B pressed

on button A+B pressed
```

Once you have completed this, try to add the following features:

- Add a way to clear the binary and decimal values so that you can start over again.
- Add a way to erase the previous value.
- Create a decimal-binary converter that allows you to enter a decimal value and see the binary equivalent when you press A+B

Teacher answers:



```
on start
    set decimal to 0
    set binary to " "

on button A pressed
    set binary to join binary "1"
    show string binary
    set decimal to decimal × 2 + 1

on button B pressed
    set binary to join binary "0"
    show string binary
    set decimal to decimal × 2

on button A+B pressed
    show number decimal
```

Self-evaluation

What were the variables that you used to keep track of information?

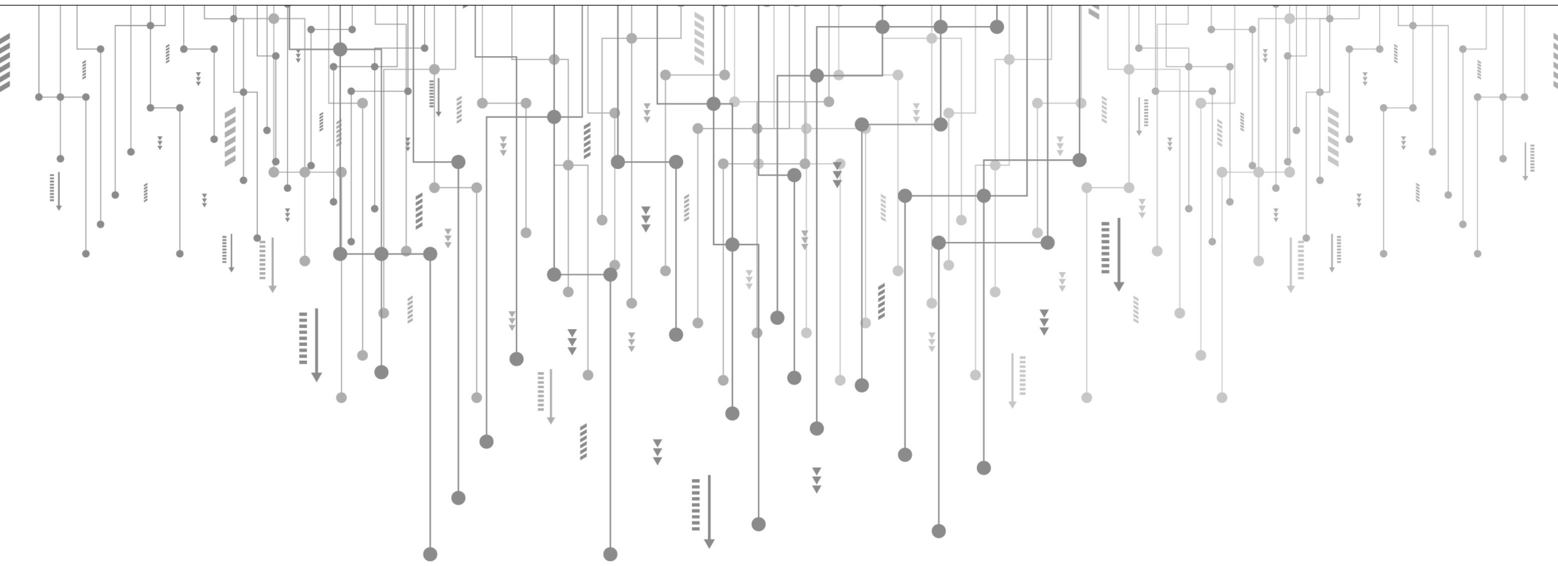
What mathematical operations did you perform on your variables?

Describe what the physical component of your Maker project was (e.g., an armband, a wallet, a holder, etc.)

How well did your prototype work? What were you happy with?

What would you change?

What do you think can be improved?



DESIGN AND TECHNOLOGY