DT Challenge Year 7/8 Python Sport Micro:bit

https://groklearning.com/course/aca-dt-78-py-microbit-sport/

About this activity

This course is designed to be an introduction to programming using the general programming language Python. The students will use a version of this language called micro:Python to program small computers called BBC micro:bits.

In this challenge students will learn how to program a micro:bit to respond to interactions and movement detection. They will be able to program and then play games that test their reaction times, strength and balance using sensors already built into the micro:bit. The BBC micro:bit is a small programmable computer with built in LEDs, buttons and sensors (such as the accelerometer) that will be utilised by the students.

More and more technology is helping us to stay fit and healthy by allowing us to accurately test and monitor our physical activity. This challenge has a strong focus on the use of BBC micro:bits in real life, with most of the questions and projects designed to be downloaded to a physical micro:bit and used in real life.

Throughout this challenge, students will practice three main programming concepts: loops, branching and variables. Loops are extremely useful when you are programming something that should respond to an interaction. Remotes are the most common example of loops, as they loop over checking if a button has been pressed constantly. While nothing is pressed they do nothing, they are waiting to check if a button was pressed and then perform suitable actions.

Branching and if statements make decisions by checking if the state of something is true or false. Depending on the what is evaluated, different lines of code will be executed. This will be introduced as students program functionality into the buttons on the micro:bit, with their first if statement being “if button A is pressed do something”.

Variables store data, text or numeric, for use or analysis later. Students will learn that variables can be used to save pieces of data, and then updated as many times as necessary, which is why they are an extremely powerful tool in the programmers kit.

Age

This challenge targets students in years 7-8, though it can also be used students in earlier years who have worked through block based programming and seek additional challenge.

Funded by the Australian Government Department of Education and Training.

Language

Python, a general programming language that is widely used in industry. Using English-like commands, Python is highly suitable and popular as a language in which to learn fundamental programming concepts.

Time

The course is designed to be completed in approximately 8-15 hours of class time — it can be extended with the suggested investigation and by implementing the games they build.

Objectives, Content Descriptions & Key Concepts

Digital Technologies

Content Descriptor Code	Content Descriptor	Key Concepts	Addressed by Sport Micro:bit through:
ACTDIP027	Define and decompose real-world problems taking into account functional requirements and economic, environmental, social, technical and usability constraints	Specifications	Breaking down a problem into smaller components
ACTDIP029	Design algorithms represented diagrammatically and in English, and trace algorithms to predict output for a given input and to identify errors	Algorithms	Representing solutions diagrammatically/in English
ACTDIP028	Design the user experience of a digital system, generating, evaluating and communicating alternative designs	Interactions	Design user interfaces on the micro: bit Unplugged activity
ACTDIP030	Implement and modify programs with user interfaces involving branching, iteration and functions in a general-purpose programming language	Implementation	Writing solutions that involve branching and iterations in Python

Health and Physical Education

Content Descriptor	Content Descriptor	Key Concepts	Addressed by Sport Micro:bit

Funded by the Australian Government Department of Education and Training.

Code			through:
ACPMP080	Use feedback to improve body control and coordination when performing specialised movement skills in a variety of situations	N/A	Use of micro:bit to gather data during physical movement
ACPMP088	Modify rules and scoring systems to allow for fair play, safety and inclusive participation	N/A	Use microbit to design and apply scoring systems
ACPMP083	Participate in physical activities that develop health-related and skill-related fitness components, and create and monitor personal fitness plans	N/A	Implementation of games which encourage physical movement.

Funded by the Australian Government Department of Education and Training.

What are we learning? (Abstract)

At the conclusion of these activities students will be able to:
Sports

- Discuss the impact physical activity has on their heart rate
- Obtain data as feedback, to improve body control
- Measure their reaction speed, strength and balance using the micro:bit and discuss strategies to improve
- Understand that fitness is a component of health

Digital Technologies

- Write programs using a general purpose programming language
- Recognise that breaking down a problem into smaller steps (decomposition) makes it easier to solve problems
- Be able to debug algorithms
- Design algorithms and represent them diagrammatically
- Trace algorithms to predict output, and identify errors
- Recognise that steps in algorithms need to be accurate and precise
- Recognise that problems can have multiple solutions
- Utilise branching (if, if-else, and if-elif-else) in programs
- Define the term *algorithm*
- Define the term *branching*

Module outline

The course consists of 5 core modules:

1. Getting started with micro:bit
   This module introduces showing images and text on the micro:bits 5x5 LED display.
   It also introduces simple sequencing of steps and timing to display multiple types of output.
2. Loops, buttons and music
   This module introduces the main micro:bit loop for repetition of steps.
   It also introduces the concept of branching by programming functionality into the buttons.
3. Simple decisions and variables
   This module expands on branching decision making by programming both buttons and introduces conditions.
   It also introduces variables as a way to store information and introduces the micro:bits inbuilt timer.
4. Accelerometer and loops
   This module introduces the accelerometer and the generalised movements it can detect.
   It also goes into conditional decision making in detail and breaks conditions down to their core true or false comparisons.
   It also introduces a new kind of loop (while loop) so that their micro:bit can ‘react’ to movement.
5. Numbers and Pixels

Funded by the Australian Government Department of Education and Training.

This module breaks the LED display into its individual pixels for the first time and introduces mathematical operators so that students can start to manipulate individual pixels.
It also breaks the accelerometer into its different measurements axes so that they can start to detect more specific movements of the micro:bit.

New Vocabulary

From Digital Technologies:

Algorithm: A set of rules or step by step instructions to solve a problem or achieve an objective. A recipe is an example of an algorithm - it sets out what you need and the steps you follow to combine everything to create your food item(s).

Decomposition: breaking down a problem into smaller parts which can then be dealt with individually. This allows very complicated problems to be solved by first solving their individual parts separately and then working out how those individual solutions can be used together.

Branching: Changing the instructions executed by the program based on a certain condition. This allows you to specify that your program should behave one way in some cases, but a different way in others. In Python, this is achieved through the use of `if-elseif-else` statements.

Conditions: The condition being checked during branching. It will return true or false depending on whether or not the condition being checked for is true or false. “If the button A is pressed” is a condition “If the micro:bit is shaken” is a condition.

Types of component:

Discussion **Worksheet** **Plugged Activity**

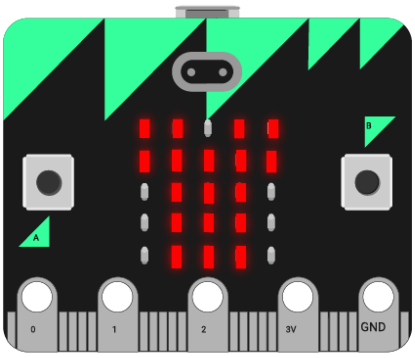
Group Activity **Unplugged Activity** **Video**

Animation **Reflection** **Game** **App**

Funded by the Australian Government Department of Education and Training.

Overarching Activity: Egg and Spoon Race

Preparation and timing



The challenge consists of 5 modules, with an additional extension project for students who have had some experience with programming before or who move through the material quickly.

Completing all four modules should take around 8-15 hours.

Overview

- The challenge assumes no programming knowledge
- It includes activities that incorporate programming concepts, HPE concepts and Python syntax concurrently

Suggested Implementation

Plugged Activity

Challenge modules

To get the most out of the modules, students should always:

- Read the interactive notes, including running any example code provided
- Attempt all problems and review questions

Clicking the **Run** button (▸) allows students to check their code by running it and **observing the output**. When they believe they have the solution correct, pressing the **Mark** button (★) will check to see if the code passes the test cases, and will provide feedback if it does not.

You can interleave the unplugged activities below with completion of the online modules, especially if you find students are struggling with a concept explored in one of the activities.

Many slides and all problems include “Teacher Notes” that verified teachers can access. These provide additional information, suggestions and activities for teaching each concept and exploring ideas further with students in class.

Funded by the Australian Government Department of Education and Training.

Activity 1 — Unplugged: Abstract Drawing

Preparation and timing

Before starting this activity, the students will need four pieces of plain paper (post-its work well) and a print out of the 5 X 5 grid (linked below).

This activity can be done prior to or during Module 1, which covers how images are shown on the micro:bit display. It could also be done during Module 3, where custom images are introduced.

Overview

- How much information is really needed to convey an idea?
- Abstraction requires the removal of unnecessary information.
- Can an animal be drawn with 25 dots?

Suggested Implementation

 **Unplugged Activity**

Abstract Drawing

Instruct students to draw a pig on one piece of paper and a dog on another one, keeping the drawings to themselves. They should have no more than 1 minute to do this, and should be done individually.

> *The short time frame is to try to stop them from trying to draw something too detailed.*

Ask the students to show only one picture to the student next to them. Can they guess which animal it is? (encourage a little discussion about funny drawings).

 **Discussion**

Did anyone fail to identify the drawn animal? What were the things that helped you to identify the difference between a pig and a dog?

> *Suggestions should include curly tail, snout, perhaps long waggy tail, tongue, pointy or floppy dog ears.*

Take 2:

Try the activity again, this time have the students see if they can represent the pig and the dog on the two remaining pieces of paper with the fewest lines possible.

Funded by the Australian Government Department of Education and Training.

Have them show the pictures to their partner to guess. Was it as easy? Did anyone do a particularly good job of representing a pig or a dog in a tiny number of lines?

Discussion

This is an activity based on abstraction, the students are abstracting away many details about pigs and dogs so that they key identifiers are all that remain. Computer science is all about abstraction — making sure we ignore the unimportant and distracting detail to focus on key details to solve a problem. Abstraction allows us to create a version of the problem being solved that can result in a generalised solution. E.g. if someone were to ask us to calculate the area of a rectangle that is 5m by 3m, a generalised solution would be a way to calculate the area of any rectangle, not just the one with the dimensions of 5m x 3m.

As humans, we abstract details all the time: if someone asks us how we got to school that morning, we generally only give them a short response (e.g. “I drove”). We don’t go into the steps we performed like what time we woke up, what side of the bed we got out of, if we got to the car from our front door or from the garage, which roads we went down, how many lights we stopped at etc etc. Similarly, for computers to solve problems effectively, they need the solution to be general (or abstract) enough that it can solve all the problems of a certain type.

Imagine if a calculator could only add 2+2 and nothing else! It would be an overly detailed solution to the addition problem. Instead, it knows what abstract process is involved in addition and executes those processes for specific numbers it is given.

Faces in 25 dots

Ask the students to suggest some facial expressions that have very identifiable characteristics.

Suggestions should include things that are represented by emoji like happy, sad, surprised, angry etc.

Get the students to colour squares on the 5 x 5 grid to represent an expression that their partner can guess.

- Link to the 5 x 5 grid printable (Google).
- Link to the 5 x 5 grid printable (PDF).

Rules:

1. They may only use one colour
2. They must colour the whole square of any square they colour

Show the pictures and ask for guesses from the class. (Encourage enthusiastic showing around the room.) Compare how many used a nose to show an expression, did anyone use ears? How many people had eyebrows? How important are these features to represent the expression? For example, showing a happy face might not need eyebrows, but a quizzical expression would need eyebrows.

Funded by the Australian Government Department of Education and Training.

Discussion questions:

- Was it hard to abstract away the least important information?
- Was it easier or harder when there was a very specific limit for how much you could draw?
- What are some other examples of abstraction (either in how humans communicate or how computers work)?
  Ask students:
- What are some other examples of abstraction that we use in daily communication?
  - (possible answers include responses to “how was your day” which may depend on who the audience is; people learning to drive a car only learn what they need to drive it and not the internal mechanics of the car; slang/words that mean certain things etc)
- What are some ways in which commonly used computer applications have abstraction?
  - (possible answers include “Web browsers enable a user to go to many websites, not just the same website every time”; “Calculators allow many different types of numeric calculations”; “I can type any sequence of characters into a word processing document”)

Extension - Define a sport with one piece of equipment
Ask students to pick their favorite sport. They will then have 2 minutes to draw one piece of equipment or uniform used in the sport that they think **abstractly represents** the **whole sport**.

For example, a football represents football, a tennis racquet represents tennis.

Encourage some harder examples like Running, Hurdles, High jump, Long Jump, Discus, AFL vs International Rugby.

Can they represent the sport using a symbol made up on the 5 X 5 grid?

Activity 2 — Introducing the micro:bit

Plugged Activity

Challenge Module 1 - Displaying images and text

Module 1 introduces the students to the micro:bit and displaying images and text on the 5 x 5 grid of LEDs. It starts the students thinking about the correct sequence of programming instructions. This is sequencing.

A *sequence* is a set of instructions in a computer program which the computer performs in a set order. Each instruction is executed immediately after the one before is complete.

Funded by the Australian Government Department of Education and Training.

Focus point: Draw the students attention to the “Wake up time” slide. In embedded systems the students need to add delays into the program to make sure there is enough time for humans to perceive the changes of instructions.This is represented in the micro:bit as the “sleep” time defined between images so that they have time to see the first image before it is replaced with the second.

This module also gives instructions on how to transfer the problem solutions they build onto micro:bits, which will be important to learn in order to implement the games later. We have a blog post that explains in detail how to get you and your student’s programs onto the physical micro:bit. You can read it here:
https://blog.aca.edu.au/uploading-a-program-from-grok-onto-the-bbc-micro-bit-b89fbbac2552

Game
Duck Duck Ghost

Students will have needed to complete the question in Module 1 titled Duck Duck Ghost. They will also need a micro:bit, however a micro:bit example displayed on a projector screen for the class to see will suffice.

The problem called ‘Duck Duck Ghost” is a variation on the classic Duck, Duck, Goose game. Rather than being able to choose who they tag as the goose (ghost), the micro:bit will tell them to tag someone as the ghost after a certain amount of time.

As they repeat the same steps while playing the game (duck, duck until ghost, then the chase, resume the duck, duck until ghost, then chase), start to introduce them to the idea of a loop of steps. It is the next concept introduced and this is a great stepping stone opportunity.

Funded by the Australian Government Department of Education and Training.

Activity 3 — Loops buttons and music

Computers are used in so many things we don't think of as computers. Things like cars, airplanes and traffic lights, even toasters, TV remotes and smart light bulbs.

Programming is needed in all of those devices that have computers. By learning how to program embedded systems you can learn how some of the technology around you works.

The examples in this challenge abstract away the details of some real world embedded systems so students can explore the basic concepts of input, output, branching and iteration in real world systems without being given control of a traffic sign or a complex navigation system.

Preparation and timing

No prior knowledge is required for this activity.

This activity would be best delivered either prior to or concurrently with Module 2, where the micro:bit's branching and loops are introduced.

Overview

- An introduction to branching as a simple yes or no check
- Loops keep programs checking on repeat

Game , Discussion:

Are we there yet?

On the board, draw a house on one side called "home" and another house on the opposite side of the board called "school". Tell the students that they must face the back of the class and you are going to draw a line connecting home to school. When students think you have finished the line 'journey' they should turn and say "Are we there yet?".

If they turn around and you are **not** there yet, they are out of the game and they have to face the back again with their hands on their heads. They can only check once. First student to turn around and correctly check if you have finished wins. Play a few rounds (the line can go all over the board and in loops and squiggles to make the 'journey' take longer and longer).

Would students have a better chance at winning if they could check over and over and over? Of course they would! Play some rounds letting the students turn around as many times as they want. If they turn and ask "Are we there yet?" and you are not there, they must turn back before they can check again.

Funded by the Australian Government Department of Education and Training.

During that game the students were acting like little “Are we there yet?” programs. In their first round they were checking “Are we there yet?” one time, then their program finished and they were out. In the second round, their “Are we there yet?” program had a **loop** that let them check again and again.

Another example of this kind of program is a remote control. It would use useless if the remote checked if you pressed a button only once and then stopped working. Instead, the remote checks on repeat if you pressed any buttons so that it can catch when you press something and it can respond accordingly. These are loops in programming that will be introduced to in the next online module.

This game also introduces branching with 2 options. When the students turn to check they branch into two potential outcomes. If the line ‘journey’ is not finished, they must turn around and check again. If the line ‘journey’ is finished, they win.

Plugged Activity:

[Challenge Module 2 - Loops, buttons and music]

The second challenge module teaches branching and the micro:bit infinite loop. The micro:bit loop is what keeps the micro:bit instructions running and allows it to constantly check for and respond to button presses.

Activity 4 — More branching and Variables

Overview

- An extension of branching
- What are conditions?
- What is a variable?
- How does it relate to programming?

Group Activity, Game
Decisions with 2 options

The introduction of the **else** can be tricky as it is usually addressing the part of a decision that is implied in real life, but must be explicitly defined in computers.

Play a game similar to Simon Says to help students to understand. One student will stand out the front of the class. Simon says has an implied else to it. If Simon says “Simon says” the students do what they say. **Else, do nothing**. Have the class play Simon Says in the classic manner for a few rounds.

To make the else more explicit, have the class choose a move or stance to take if the person does not say “Simon Says”. For example, putting their hands on their heads. Write it on the board in the format “Else: (the move to do)” so that they can see it. Now, as they play, when simon does not say “Simon says” the students must make the Else move, they will be explicitly be acting out the else statement.

The important thing to remember about else statements is that they don’t check for any condition. They don’t have to.

For example;
If the answer is yes, do this. Else, if the answer is no, do that.
If they say “Simon says”, copy them. Else, they don’t say “Simon says” so don't copy them.
Can be shortened to;
If the answer is yes, do this. Else, do that.
If they say “Simon says”, copy them. Else, don’t copy them.

Game

Dance Dance Revolution

Dance Dance Revolution explores branching using the if/ elif/ else. The else should come last as it does not check anything. Remember that else does not check for anything? Elif (else if) is a way to check multiple things in an if statement. Make sure that all options that should be checked are checked first, followed by else which covers all other options.

Discussion

What is a variable?

Variables are something that exist in the world around us but the concept proves difficult for students to grasp. At its core, a variable has a name and contents. You use the name of the variable to refer to the contents in programming. Variables can have their contents changed and updated.

A real world examples:

- The score in a sports game is a variable. It is called the ‘Score’ (name of variable) and the contents is the number of points scored. It is updated when a team scores more points, when someone asks for the score they are asking”

Funded by the Australian Government Department of Education and Training.

    - What is the contents of the variable named score?
- The Prime Minister of Australia is a variable. The variable name is 'The Prime Minister of Australia'. The contents of the variable is the name of the person who is currently in office. The name of the person changes with elections. If you talk about 'The Prime Minister' other people know who you are talking about without you needing to use their name.
    - What is the contents of the variable named 'Prime Minister'?

Have the students try and come up with everyday variables of their own. If you wanted to include branching you could ask the students to define what would result in a change of the variable content. For the Prime Minister example, your branching example would be: "If a new Prime Minister is elected, update the Prime Minister variable to contain the new name."

Plugged Activity:

[Challenge Module 3: Simple decisions and variables]

The third challenge module covers variables at the end so this activity could be delivered after the completion of the module or concurrently.

Game

Reaction Time tester

This game requires the students to use variables and arithmetic. Arithmetic using the micro:bit is not explicitly covered in the early slides but the example on the slides before is very similar to what is required of them.

It might be beneficial to go through the calculations that are explained with the class a few times. Using the variable names from the problem will also help their understanding of variables and how they can contain different values.

Unplugged Activity

Algorithms

Algorithms are a key concept in digital technologies, and an important step when designing the logic of a computational solution. Students sometimes confuse algorithms with programs - and one way to distinguish this is to think of algorithms as the blueprint/plans; and the program as the construction which follows the design on the blueprint. Just like in architecture and construction, good algorithm designs that consider all the requirements will mean a better solution.

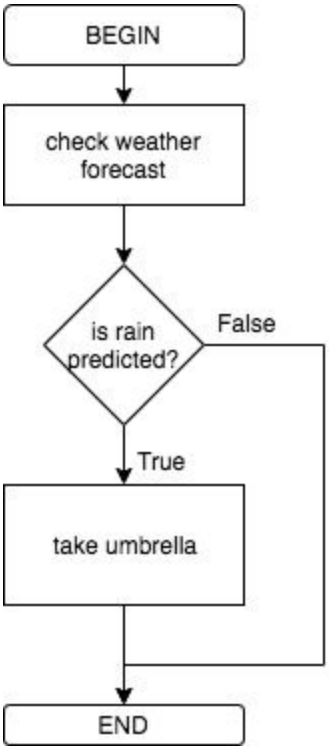
Funded by the Australian Government Department of Education and Training.

We follow algorithms all the time, even if the term itself is new to us. Humans follow instructions, design sequences of activities, make decisions and repeat steps in all tasks that we undertake on a daily basis. This is one way to link algorithms as a key concept to prior learning and understanding.

In year 7-8, students are required to:
"Design algorithms represented diagrammatically and in English, and trace algorithms to predict output for a given input and to identify errors"

Diagrammatic algorithms are more commonly known as flowcharts. Algorithms can also be described using simple, structured English, which is called pseudocode. The *design* of algorithms is an expectation of the Australian curriculum in Years 7-8, and thus incorporating this step as a **precursor** to writing code is one way to integrate algorithms and their implementation.

1. Present the following diagram to students. Ask them to follow the diagram, and identify the purpose of this algorithm (ie what problem it might solve).



(Answer: this is an algorithm to help decide whether to take an umbrella, depending on the weather prediction).

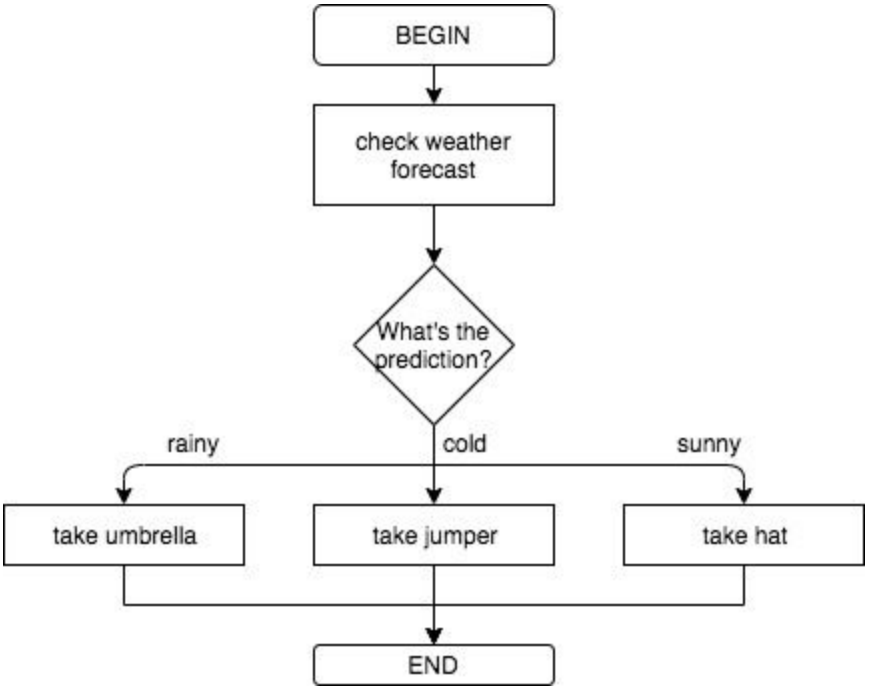
2. Discuss the symbols used in this diagram. What do the rectangles, arrows, diamond and rounded rectangles represent? Help students reach a shared understanding of the representations, and what they mean.

Funded by the Australian Government Department of Education and Training.

a. Rectangle: individual steps (or instructions)
b. Arrows: represent the order of the instructions
c. Diamond: represents a decision - in this case, there are two possible outputs
d. Rounded rectangles: representing the start and end of the algorithm

Every algorithm must have a single start point and a single end point.

3. Present the following algorithm to students. What does this algorithm do, and how is it different from the last one?



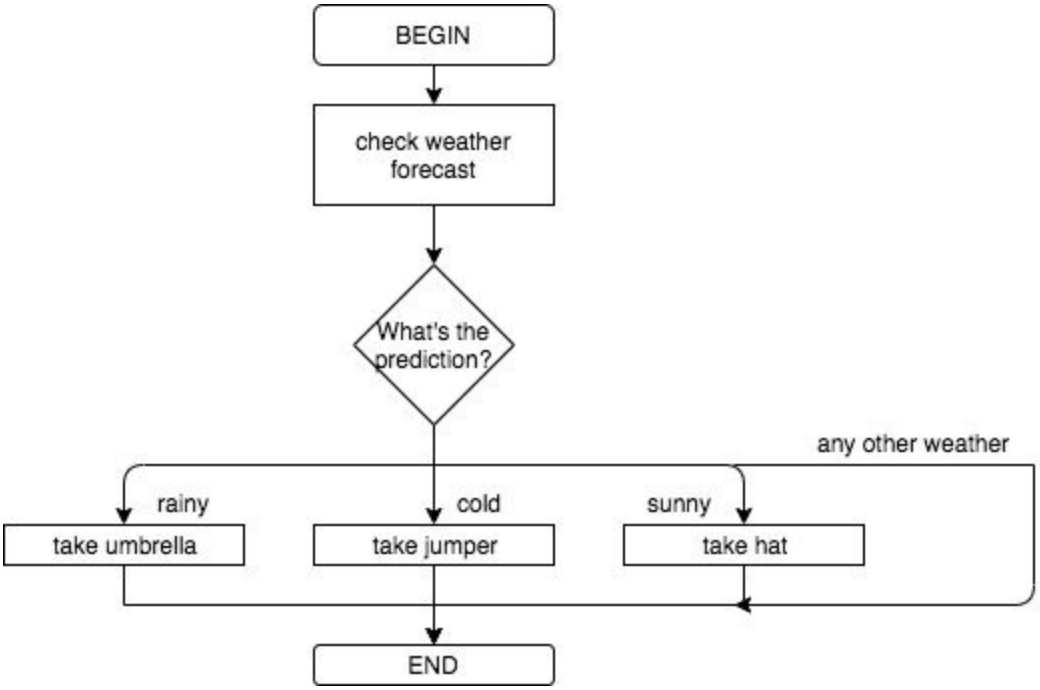
(Answer: this algorithm caters for more weather options, and better reflects every day decision making.)

4. What might be the limitations/problems with this algorithm?
   (Answer: there is no option for if the weather is something other than rainy/cold/sunny. What if it is windy, or hailing, or stormy, or some other option? If that happens, the algorithm is stuck, cannot end and the program may crash.)

5. How might we improve the algorithm so that it can incorporate *any other* weather conditions?

   Answer: have an option from the decision (diamond) so that any other weather prediction can still successfully end the algorithm.

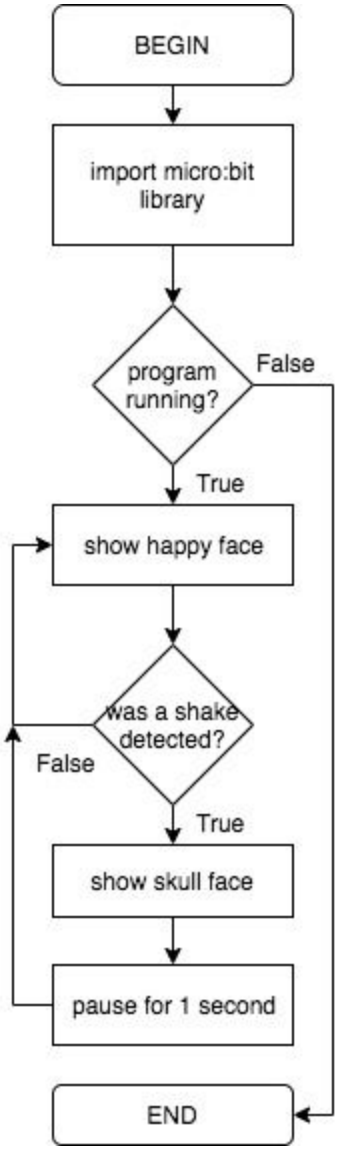
Funded by the Australian Government Department of Education and Training.



6. Designing our own algorithms: ask students to design algorithms for teacher selected/student selected problems from Module 4, **prior** to coding their solution. In pairs, students may check each other's algorithms for how well that approach may solve the problem, and identify any errors in logic. Student designed algorithms and feedback can be used as formative/summative assessments.

   E.g. The following is an algorithm for the solution to "Try not to Shake" in Module 4.

Funded by the Australian Government Department of Education and Training.



Activity 5 - The Accelerometer and conditions

Preparation and timing

Stopwatches or a single stopwatch on a screen that the students can all see.

Overview

- What is the difference between IS_Pressed and was_pressed?
- Practicing a skill makes you improve, but you need to think about how you practice
- Accelerometers detect movement
- Ways to improve muscle strength

Accelerometer?

Unplugged Activity
The is_pressed vs was_pressed.

To illustrate is_pressed, hand the students stopwatches (or find an internet stopwatch to project for the class to see) that shows milliseconds. If it shows microseconds even better. Ask students to try and stop it at exactly 5 seconds (5:00:00 seconds). It should prove extremely difficult, if not impossible.

When they use the ‘is pressed’ statement in their code, the micro:bit only checks if the button is pressed for a tiny amount of time, perhaps a few microseconds. Trying to press the button at the exact time the micro:bit reaches the check in the loop is like trying to stop the stopwatch at exactly 5 seconds.

The ‘was pressed’ is different. It doesn’t actually check the button state at all! When ‘was pressed’ is selected, the micro:bit allocates a bit of extra memory to the button. Whenever the button is pressed that bit of memory turns on. When the micro:bit goes to check the button press it just looks at the memory bit instead. If that memory bit is on, the button was pressed.

Plugged Activity:
Challenge Module 4: Accelerometer and loops

The following unplugged activities should be completed while students are completing the module. It rounds out the teaching from the slides and helps to give a real world example that the students can clearly see.

Funded by the Australian Government Department of Education and Training.

Game **Discussion**

Try not to Shake, Musical Legs

After the students have had a chance to play the balancing games, you can ask them some discussion questions about the sports aspects of the challenge.

- Did you use your arms to help you balance? Would you be as good with your arms behind your back?
- Do you think the micro:bit detected a shake more when you ran faster? Why?
- Do you think the micro:bit detected a shake more if you ran on your toes? Why?
- Could you practice balancing and get better? How?
- Do you think having stronger leg muscles will help you hold up one leg longer? What can you do to make them stronger?

Game **Discussion**

Milk Bottle Challenge

After the students have built their milk bottle challenge they can take turns using the micro:bit to see how long they can hold up the bottle. Some discussion points that you can cover would be:

- What different kinds of exercises can they do to make their time longer?
- What can they do to make the challenge easier for someone who is struggling to hold up the bottle?
- What is a safe stretch to use before you try this challenge ? What about after?

Activity 5 - The Final Module

Preparation and timing

The final module builds to creating an egg and spoon race micro:bit program. Play some rounds of the classic game with students just in case they have forgotten how or have never played it before.

Overview

- You can update a variable using itself
- How to turn individual pixels on and off
- X and Y coordinates
- How Accelerometers work and what the values mean

Funded by the Australian Government Department of Education and Training.

Plugged Activity:

[Challenge Module 5: Numbers and pixels]

The following unplugged activities should be completed while students are completing the module. Some concepts may be challenging for students and so unplugged activities can help to support student understanding.

Unplugged Activity

Updating Variables

Increasing the value of a variable by itself can be a tricky thing for students to understand. They have to stretch their abstract thinking quite far to keep in mind that the variable value can be on both sides of the equals sign.

Learning to update the variable using the existing variable is an important part of learning to use variables effectively. Working through the examples in front of the class will be extremely useful for the students, and they will need lots of examples.

A good real world example to use would be the updating of scores during a sports game. When a team scores a goal, the points are added to their already existing score. You can find examples of scores from local football games played over the weekend and use them as examples on the board. (Look for games with high score totals so that you have lots of goals to work with) It will help students to be engaged if it is teams they support.

Work through the addition of each goal using only number/s initially, then replace the starting score with a variable name such as “score”.

Score = score + (number)

If you wanted to extend it further, you could replace the number of points with the name of the type of goal, and have the students calculate what the new score will be. For example:

Score = score + Try
Score = score + converted try
Score = score + penalty

Unplugged Activity

Pixel Bingo

Before starting this activity, the students will need a few copies of the print out of the 5 X 5 grid (linked below). This activity is best done **after** students have had a chance to read the “Pixel

Funded by the Australian Government Department of Education and Training.

layout" slide in module 5 of the course which explains how to interpret the coordinates of each pixel.

Ask the students to randomly choose 10 pixels out of the grid and colour them in.

- Link to the 5 x 5 grid printable (Google).
- Link to the 5 x 5 grid printable (PDF).

You then read out pixel coordinates at random, the students must find that coordinate on their 5 x 5 sheet. If is one of their coloured pixels they mark it off. Keep calling until one student has marked off all their chosen pixels and calls "Bingo!".

Extension:
Ask the students to update their chosen pixel coordinates by adding 1 to both x and y.

x = (x +1) and y = (y + 1)

If their coordinated become a 6, they must reset it to a 0. Give them a second 5 x 5 grid and ask them to add one to each of their old square coordinates and fill out new squares. This replicates what they will be programming later, where they must stop a moving pixel from 'falling off the edge of the micro:bit' by resetting coordinate values of 6 back down to 0..

After they have updated their bingo sheets, play a few more rounds of bingo. If you want to make it trickier, you could read out a coordinate and ask the students to mark their sheets at the +1 values of the coordinate you read out.

Worksheet, Unplugged Activity
Micro:bit axis helper (Accelerometer X, Y and Z)

Access the worksheet here:
Axis Helper Worksheet

Planning: Students will need a pair of scissors. This activity should be completed during or directly after Module 5, slide titled "Accelerometer X and Y (and Z!)".

Thinking in 3D space can be difficult, giving the students something physical to give them feedback will help.

Have the students complete and build the micro:bit axis helper worksheet. The sheet is designed to help them to interpret the accelerometer x and y values relative to the position of the micro:bit. They will need a pair of scissors if they want to build their own Axis helper for use during their final project. They can also use a real micro:bit on the helper if they prefer.

Funded by the Australian Government Department of Education and Training.

This worksheet complements the beginning of Module 5 and can be completed as a class activity or individually. After they have completed the sheet, ask the students to demonstrate the micro:bit position associated with different axis measurements you provide to test their understanding. The final page is a cut out which they can fit onto a real micro:bit that will show them which axes are adjusting to which movements.

Unplugged Activity

Algorithm Design

As this is a complex problem, designing algorithms to demonstrate problem solving approaches is highly recommended. Devoting some time (ie one lesson) to break down different elements of the problem, and identifying clear steps will assist with making the project less intimidating to students.

Plugged Activity

The final project

As students attempt the module it may be important to remind them that the commands and structures they are using to build it have all been encountered before.

Building the project in stages, and testing the projects on physical micro:bits will also help them to get a sense of how things are progressing and help them to see where their program may be failing.

Discussion:

Discussion Questions:

- Is there a way that someone can cheat this game in its final form?
- How would you optimise against cheating?
- How sensitive is the micro:bit to tilting in its current state? How could you make is more/less sensitive?

Funded by the Australian Government Department of Education and Training.

Activity 6 - micro:bit custom Project

Preparation and timing

3-4 lessons, either in conjunction with Module 4 or 5, or when students have completed the Challenge.

Overview

- Identify a commonly practiced physical/sport/recreational activity, and design, build and test a solution for an aspect of that activity using the micro:bit.
- Students may use various problems in Modules 4 and 5, and adapt them further for their own project. (e.g. AFL scorer)
- Sample suggestions include:
    - micro:bit step counter that can be attached to a shoe
    - Micro:bit reaction tester
    - Score tracker for various sports
    - Push-up counter
- Key steps:
    - Brainstorm ideas
    - Identify inputs (buttons/gestures) and outputs (display)
    - Design algorithm/s
    - Develop and test code
    - Present to peers/teacher
- Extension: the micro:bit has an inbuilt transceiver that is capable of radio communications. Students can program the micro:bits to transmit and receive messages between each other. This can enhance the functionality of their projects. If students are interested in exploring how to do this, the ACA's micro:bit radio mini DT challenge is a great way to learn.

If students do build their own projects, please let us know! Send their projects to help@aca.edu.au!

Funded by the Australian Government Department of Education and Training.