

# DT Challenge Year 7/8 Intro to Micro:bit

<https://groklearning.com/course/aca-mini-dt-py-microbit-intro/>

## About this activity

One of the most interesting reasons to learn programming is to interact with the real world. For example programming robots, interactive digital art or smart appliances. These are commonly referred to as *embedded systems*.

Like most examples of programming this requires us to understand the concepts of: output in order to give feedback to users, input to get data from users and the environment, branching to make decisions based on different values of input and iteration in order to repeat sets of instructions.

In this challenge we use the BBC micro:bit as an embedded system to create different pieces of a virtual pet game. Students in the process learn about the micro:bit's features and can think about how to make their own version of a virtual pet game, or even an entirely new project.

## Age

This challenge targets students in year 7/8 and, though it can also be used as an introductory course for students in later years who have not yet been exposed to basic programming concepts or who already know how to program and would like to learn about the micro:bit. There is also a version of this challenge for year 3/4/5/6 students that uses Blockly.

## Language

Python — a general-purpose programming language that is widely used and easy to learn.

## Time

The course is designed to be completed in approximately 3 hours of class time, with another hour included in the extension module.

## Key Concepts

Key Concept	Coverage
Abstraction	Students perform some abstraction themselves in unplugged activities within this course. It is also important to notice the abstract nature of animals represented in so few leds and being able to use refer to the images by name rather than draw each image with pixels. The requirement of wiring and understanding the physics behind the electronics is also abstracted away with the use of the micro:bit.
Data: collection, representation	This course deals with many types of Data Representation. Data, such as an animal can be represented by text and by images, images are made up of dots which are individually represented by a number to denote brightness. Data is collected in a very small way by the buttons and

	sensors on the device.
Algorithms, implementation	Simple Algorithms, branching, and very simple repetition
Digital Systems	This course shows the interaction between the hardware and software within the micro:bit.

## Objectives (Content Descriptions)

### Digital Technologies

ACTDIK024	Investigate how digital systems represent text, image and audio data in binary
ACTDIP029	Design algorithms represented diagrammatically and in English, and trace algorithms to predict output for a given input and to identify errors
ACTDIP030	Implement and modify programs with user interfaces involving branching, iteration and functions in a general-purpose programming language
ACTDIP031	Evaluate how student solutions and existing information systems meet needs, are innovative, and take account of future risks and sustainability

## What are we learning? (Abstract)

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

- Write basic programs using the Python programming language
- Recognise that breaking down a problem into smaller steps (decomposition) makes it easier to solve problems
- 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
- Utilise for loops in programs
- Define the term *abstraction*
- Define the term *sequence*
- Define the term *algorithm*
- Define the term *branching*
- Define the term *iteration*

## Module outline

The course consists of three core modules:

1. Displaying images and text

This module introduces showing images and text on the micro:bit's 5x5 LED display.

It also introduces simple sequencing of steps in order to display multiple types of output.

2. Making decisions with buttons and gestures

This module introduces the idea of repeating a part of a program forever to continually display an animation or react to input. It also introduces input in the form of buttons and gestures. And it introduces using branching based on input to result in different outputs.

3. Virtual Pet extensions

This module uses the concepts from the previous modules to explore extra features of the micro:bit, such as custom images (output), animations (output/iteration), music (output), randomness (input/branching) and the temperature sensor (input/branching).

Types of component:			
 Discussion	 Worksheet	 Plugged Activity	
 Group Activity	 Unplugged Activity	 Video	
 Animation	 Reflection	 Game	 App

## Challenge Introduction — what are embedded systems?

Computers are used in so many things we don't think of as computers. From cars, airplanes and traffic lights to toasters, TVs and smart light bulbs.

Because computers are in almost every device we use programming is used in all of those devices too. So by learning how to program embedded systems you can learn about how the world 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.

We have a blog post that explains in detail how to get your 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>

## New Vocabulary

**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-elif-else`.

**Iteration:** Repeating a set of instructions a number of times as part of the program. This allows your solution to work over multiple changes in input. In this challenge, a non-terminating `while True` loop is used for iteration and we don't cover any terminating loops until an extension problem which contains a `for` loop.

**Abstraction:** Abstraction is hiding details of an idea, problem or solution that are not relevant, to focus on the important information.

As humans we abstract away details all the time. When someone asks you how you got to school today do you tell them the detail of every step you take or do you generally answer "By car" or "By bus"?

# 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)

## 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**

Give the students 2 minutes to draw a pig on one piece of paper and a dog on another one.

*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 get the students to see if they can represent the pig and the dog on the two remaining pieces of paper with the fewest lines possible.

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

Talk about the fact that the students have abstracted all but the most identifiable details away.

Computer science is all about abstraction - making sure we ignore the unimportant and distracting detail to focus on what's necessary to solve a problem.

### Animals in 25 dots

Ask the students to suggest some animals, like the pig, that have very identifiable characteristics.

*Suggestions should include elephants(trunk), giraffes(neck), camel(hump), narwhal(horn)*

Get the students to colour squares on the 5 x 5 grid to represent an animal 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 get the guesses. (Encourage enthusiastic showing around the room)

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?

## 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.

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

Focus point: Draw the students attention to the “Ducks in a Row” 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.

## Activity 3 - Branching: Simon says

### Preparation and timing

Most students will have played this game before.

### Overview

- What is branching?
- How does it relate to programming?
- How do we determine the conditions that we need to check to perform different instructions?

## Suggested Implementation



### Unplugged Activity



### Group Activity

#### **Simon says...**

Ask the student if they've all played simon says before and, if any of them haven't, explain the rules of the game. You should then play a game with the group. Older students may not make many mistakes, and you may choose to end the game early if they aren't being knocked out very quickly.

You should then say that you'll play another game, but this time make the rule a little more difficult. Instead of the rule being *does the instruction start with "Simon Says"*, try something like *does the instruction include the word "you" or does the instruction have exactly five words in it*. You are demonstrating that the condition can change, but regardless what determines if the instruction should be performed is some condition that you've set as your rule.

*Branching* in programming is what allows us to define different behaviour in the same program. It allows us to check some value or condition in our program, and respond differently based on what the result is. Without branching, all of our programs would perform exactly the same thing every time, and we would need to write brand new programs every time the tiniest little thing changed. Our programs would not be able to respond differently to new users or data.

Both Blockly and Python provide ways of performing branching, and these are covered in modules 3 and 4 of the challenges.



### Plugged Activity

#### **Challenge module 2: Making decisions with buttons and gestures**

The challenges have been designed such that there are two programming exercises for each of the concepts introduced. The questions are very similar - often the only difference will be the text being printed and/or used in the input prompts.

In module 2, you may find it useful to walk through the first question in each set as a group, demonstrating the use of branching for students and explaining the thinking process. You can then have students complete the second question in the set on their own or with a peer, giving them a chance to apply the knowledge and what they've seen demonstrated to solve a problem on their own.

Discussion questions:



**Reflection**



**Group Activity**

***Computer decisions are encountered everywhere***

Students share some examples where they think branching occurs in the programs they use regularly. Things might include:

- If the phone rings for 30 seconds and isn't answered, it automatically goes to voicemail
- If you choose a certain character in a game, then it loads that character so that you can play as that person
- If you try to use an ability in a game and it isn't charged, it doesn't let you do it
- If the user types in different answers to a question in a program, different options are presented to them for their next action

All of these actions require the program to check some value or condition. The way the computer responds is determined either by the user themselves, or due to the state of the program at the time the check is performed.

## Activity 4 - Digital Systems - Pass the pinky squeeze

### Preparation and timing

There is a combination of unplugged and plugged activities in this set of activities. There are no resources required but the unplugged activity requires some space.

### Overview

- How does hardware communicate with other pieces of hardware?
- How does animation work?
- Different kinds of data is fundamentally represented the same way in a digital system



**Unplugged Activity**



**Group Activity**

Get the students to sit within reach of each other. They should link pinky fingers. Tell them you're going to pass messages kind of like a digital system does. Everyone is going to close their eyes. When you feel one hand squeezed, you need to squeeze the hand on the other side.

Send the "message" around the circle once or twice in each direction so that the students get the idea of how to do it. Then remove yourself from the ring. You're going to muck around with the message. Ensure the students still have their eyes closed and then make a break in the circle and get the students to send a message... it will never make its way all the way around the circle. Try the next one but nominate two students to start the message - there should be a very mixed up response.



**Plugged Activity**

***Challenge module 3: Virtual Pet extensions***

The challenges have been designed such that there are two programming exercises for each of the concepts introduced. The questions are very similar - often the only difference will be the text being printed an/or used in the input prompts.

In module 3, the students will explore some of the more customisable aspects of the Micro:bit display. Encourage them to produce their animal design from Activity 1 on the micro:bit grid. These are good ancillary activities to build the students skills towards a mini project of their own.

Have the students complete Module 3.

**Reflection****Group Activity**

***Electronics requires stable connections between the hardware in the digital system***

Students suggest pieces of hardware that require stable wired connections

- A mouse needs to send messages to a computer
- Headphones are connected to a phone with a wire
- Printers are connected to a computer