# DT Challenge Year 7/8 Python Smart Garden

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

## About this activity

In this challenge, students will learn how to use the micro:bit to measure temperature, and how to connect a simple soil moisture sensor. They can then use this “Smart Garden” device to monitor the health and growth of plants.

Scientists use computer-based sensors to collect all sorts of data about the world, from weather and climate data like temperature and air pressure, to the innermost workings of atoms and particles in particle accelerators. Sensors allow scientists to carry out careful experiments, by giving them the ability to measure accurately different variables. They are also embedded into devices and instruments in all areas of modern life, from farm equipment to aeroplanes, from supermarket checkouts to baby monitors.

The BBC micro:bit is a simple but powerful programmable computer that has several built-in sensors, such as a temperature sensor, accelerometer and compass. It also can easily be connected to external equipment to read data from a wider range of sensors.

One of the critical decision-making constructs in computer programming is that of branching — often implemented in programming languages as an “if-statement”. If-statements determine whether a condition being tested is either True or False, and will execute different statements on that basis. In this challenge, students program the micro:bit in *micropython*, a version of the Python programming language specific to the micro:bit, constructing simple but powerful if-statements to allow this simple device to be controlled through its two buttons.

Through this challenge, students will learn about if-statements and branching, as well as the basics of displaying images and text. They will apply scientific understanding and inquiry skills to conduct a scientific investigation, collect data from sensors and analyse the data collected.

### Age

This challenge targets students in years 7/8, though it can be used as an introductory course for students in later years who have not yet been exposed to basic programming concepts.

### Language

Micropython, a micro:bit-specific version of the widely-used and easy to learn Python programming language.

## Time

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

## Objectives, Content Descriptions & Key Concepts

### Digital Technologies

| **Content Descriptor Code** | **Content Descriptor** | **Key Concepts** | **Addressed by Smart Garden through:** |
|---|---|---|---|
| ACTDIK024 | Investigate how digital systems represent text, image and audio data in binary | Representations<br>Types of data | Learn that lights are turned on with an integer 1 and off with an integer 0 |
| ACTDIP025 | Acquire data from a range of sources and evaluate authenticity, accuracy and timeliness | Acquiring data<br>Checking authenticity of data | E.g. comparing micro:bit temperature data to thermometer data |
| ACTDIP026 | Analyse and visualise data using a range of software to create information, and use structured data to model objects or events | Defining (Specification)<br>Decompose problem<br>Functional Requirements<br>Constraints<br>Visualise data to create information | Decomposition of problem into smaller components<br>Monitoring moisture and temperature over time period<br>Graphing data and relating to plant health |
| ACTDIP028 | Design the user experience of a digital system, generating, evaluating and communicating alternative designs | Interactions | Displaying symbols as patterns of LEDs to communicate with the user |
| ACTDIP029 | Design algorithms represented diagrammatically and in English, and trace algorithms to predict output for a given input and to identify errors | Designing (Algorithms)<br>Flowcharts<br>Tracing | Loops to manage code repetition, decision to run when button is pressed |
| ACTDIP030 | Implement and modify programs with user interfaces involving branching, iteration and functions in a general-purpose programming language | Branching<br>Iteration<br>Tracing | Algorithms include branching, iteration, user input |

## Science

| Content Descriptor Code | Content Descriptor | Key Concepts | Addressed by Smart Garden through: |
|---|---|---|---|
| ACSIS126 & ACSIS141 | Measure and control variables, select equipment appropriate to the task and collect data with accuracy | Characteristics of fair tests<br>Recording data | Discussion of fair test examples<br>Using Smart Garden to measure environmental factors<br>Investigation activity into plant growth — using Smart Garden to take measurements |
| ACSIS129 & ACSIS144 | Construct and use a range of representations, including tables and graphs, to represent and describe observations, patterns or relationships in data using digital technologies as appropriate | Presenting data in tables and other forms | Representing data on micro:bit in different forms<br>Investigation activity — present data on plant growth in different forms |
| ACSIS130 & ACSIS145 | Summarise data, from students' own investigations and secondary sources, and use scientific understanding to identify relationships and draw conclusions based on evidence | Analysing and summarising data<br>Identifying relationships<br>Drawing conclusions | Investigation activity into plant growth using Smart Garden — interpret data to produce plant care guide |

## What are we learning? (Abstract)

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

- Understand how living things adapt to their environments
- Identify changes in the environment that can affect the growth and wellbeing of a plant
- Understand the importance of a fair test when carrying out a scientific investigation
- Write programs using the Python programming language
- Recognise that breaking down a problem into smaller steps (decomposition) makes it easier to solve problems
- Debug algorithms
- Recognise that steps in algorithms need to accurate and precise
- Recognise that problems can have multiple solutions
- Utilise branching (if, if-else, and if-elif-else) in programs
- Utilise user input
- Define the term *algorithm*
- Define the term *decomposing*
- Define the term *branching*

## Module outline

The course consists of four 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. Sensors and Pins

   This module introduces the micro:bit's built-in sensors, focussing on using the temperature sensor
   It then shows how to construct a simple external sensor and connect it to the micro:bit, and use it to obtain data about soil moisture.
3. Buttons and Custom Images

   This module introduces the term algorithm, decisions, and allows students to write programs that change their behaviour on the basis of information from the user, through input from the micro:bit's two buttons.
   It also discusses the importance of designing a scientific experiment around a fair test, where one aspect is changed and all others are kept the same.
4. Putting it all together

   The final project brings all aspects of the first three modules together, to complete the full Smart Garden program that measures temperature and soil moisture, through interactions with the micro:bit's buttons. Possible investigations are suggested using the Smart Garden as a measurement device, and a sample investigation is provided in more detail.

## 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 blockly, this is achieved through the use of `if-elseif-else` blocks.

### From Science:

*Adaptation*: Describes how a species evolves over many generations to better survive in its natural habitat. Includes both structural adaptations (physical features that assist with things such as hunting prey, avoiding predators or surviving harsh environmental conditions) and behavioural adaptations (changes to the species' activities, such as being nocturnal to avoid daytime predators, or to take advantage of keen eyesight in dark conditions).

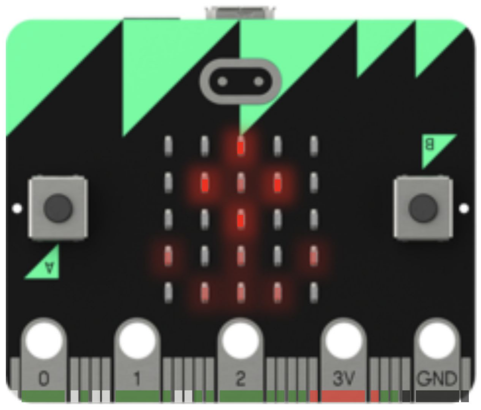
*Habitat*: The natural environment where a species lives and thrives. Even in the harshest of environments, species survive due to adaptations in physical features and behaviour.

**Types of component:**

| | | | |
|---|---|---|---|
| **Discussion** | **Worksheet** | **Plugged Activity** | |
| **Group Activity** | **Unplugged Activity** | **Video** | |
| **Animation** | **Reflection** | **Game** | **App** |

# Overarching Activity: The Smart Garden Challenge

## Preparation and timing



The challenge consists of 4 modules, with an additional extension investigation for students who have had some experience with programming before or who move through the material quickly.

Completing all four modules should take around 8-10 hours.

## Overview

- The challenge assumes no programming knowledge
- It teachers programming concepts, scientific 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 above 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.**

# 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***
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, only 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.

As humans, we abstract details all the time: if someone asks us how we are in the morning, we generally only give them a short response (e.g. "good, thank you"). We don't go into the details of the toe we may have stubbed, or the bus to school that braked suddenly causing you to stumble on the way out.  Similarly, for computers to solve problems effectively, they need the solution to be general (or abstract) enough that it can solve for all problems of a certain type. Imagine if a calculator could only add 2+2 and nothing else!

**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 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 ask for guesses from the group. (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?
- Extension: what are some other examples of abstraction (either in how humans communicate or how computers work)?

# 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 order. Each instruction is executed immediately following the one before is complete.

Focus point: Draw the students attention to the “Giraffe Duck” 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 — Senses and Sensors

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 needed 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

## Preparation and timing

No prior knowledge is required for this activity. This challenge is all about using the micro:bit as a scientific sensor to measure aspects of the world. Students can think about and discuss examples of this idea in the world around them.

This activity would be best delivered either prior to Module 1, to introduce the overall scope of the challenge, or concurrently with Module 2, where the micro:bit's sensors are introduced.

## Overview

- What senses do we have in our bodies, and what do we "measure" with them?
- What electronic sensors are in our homes, or at school, or other places, and what do they "sense"?
- How do scientists use sensors?

**Discussion:**
**Sensing with our Senses**
What senses do our bodies have? Students should be able to name most or all of the five senses — sight, hearing, taste, touch, smell. You can ask them to think about what each of those senses is actually doing, for example, what do you "see" when you see something? What do you "smell" when you smell something?

**What “smart devices” and sensors are in your house?**

What devices do you have in your house that have tiny computers in them?
Sample answers:

- Air Conditioning
- Laundry machines (washer / dryer)
- Refrigerator
- Security System
- Television

Do any of those devices have sensors in them to measure or “sense” the world around them?
Sample answers:

- Remote control
- Oven (temperature)
- Heater/Air conditioner (temperature)
- Smoke alarm
- Phone (light sensor, accelerometer, camera, …)
- Lift doors

**Discussion:**

What information does a farmer need to care properly for a crop of plants? Ask the students to come up with as many things as they can that could affect how well the crop grows, for example:

*Amount of rain, amount of sunlight, temperature, winds and storms, kind and quality of soil, type of pests, …*

Then they can discuss which of those things the farmer would be able to measure, and how would they do that? What devices would the farmer use to measure, for example, the temperature? What about nutrients or moisture in the soil? What about predicting the weather next week, or next month?

Farmers, and many people who need detailed information — especially scientists — use computers and sensors to gather information about the world. Sometimes it’s to help grow plants efficiently, sometimes it’s to understand the world better.

# Activity 4 — Branching: “Simon says …”

## Preparation and timing

Most students will have played this game before.

This activity could be delivered prior to or concurrently with Module 3.

## 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. Olders 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 includes the word “you”* or *does the instruction have exactly five words in it*. You are demonstrating that the condition can change, but regardless of 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.

Blockly provides a few different ways of performing branching, and some of these are covered in modules 3 and 4 of the challenge.

## Plugged Activity

### *Challenge modules 3 & 4: branching/decision questions*

In modules 3 and 4, you may find it useful to walk through the examples as a group, demonstrating the use of branching for students and explaining the thinking process. You can then have students complete the problems on their own or with a peer, giving them a chance to apply the knowledge and what they've seen demonstrated.

## Discussion questions:

**Reflection** **Group Activity**

### *Branching is everywhere!*

Students share some examples where they think branching occurs in the programs they use regularly, or other situations in life. Things might include:

- If the phone rings for 30 seconds and isn't answered, it automatically goes to voicemail
- If you choose a particular song on a CD or streaming app, it plays that chosen song
- 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 5 — “How To Care For Your Plant!”

## Preparation and timing

This longer investigation activity can be done once the full Smart Garden project has been completed, so that the micro:bit can be used to monitor some of the variables. It can also be done independently of the Smart Garden challenge (though it wouldn’t be as fun!).

This investigation gives the students a plant to look after — these can be prepared ahead of time, or the class could germinate them from seeds as a prior activity. Tomatoes, peas, and herbs like mint and parsley are quite easy to grow and look after (and could be taken home, given away or sold at a school fair!).

They can do this investigation individually, or in small groups. With a whole class involved, enough data could be collected for discussions about what has the most effect on this type of plant — the sunshine, the water, the temperature, or all of these.

Give each student a copy of the investigation worksheet, and if they are using the Smart Garden, make sure their program is working and they know how to use it.

Students can discuss all the different things their plant needs to grow well, and what they can do to measure and keep track of what they are doing to care for their plant.

## Overview

- What different variables affect how well a plant grows and thrives?
- How can we measure and monitor those variables?
- How do we provide information for someone else to follow to take good care of the plant?

## Suggested Implementation

**Unplugged Activity** **Discussion**

***What does your plant need?***

Discuss with the students what their plants need to grow and flourish — for example, the right amount of water, sunshine, the correct temperature.

They can then discuss how they could measure some or all of those things, and make a record of their measurements. How can they monitor changes in the plant’s environment over time, and bring that data together to present the information in a clear way?

## Discussion questions:

- What does my plant need to survive? What does it need to thrive?
- How can I measure those things? Are any of them things I can measure with numbers (e.g. temperature)? Are any of them things I can record with words or images/photos?

**Plugged Activity OR Unplugged Activity**
**Worksheet Group Activity**

***The Investigation***
The details of the investigation itself are on the worksheet — download here (PDF).

Students can use the Smart Garden device to measure and monitor the temperature and soil moisture levels for their plants. The investigation can be done with or without the micro:bit, though, by finding other ways to measure the temperature and amount of water given to the plants.

With enough information from the whole class, the students should be able to prepare a "How To Care For This Plant" note to be attached to their plant — which can then be taken home, given away as gifts, or sold at a school fair.

## Extension:
## Add Radio Communications to your Smart Garden

Any students who are interested in taking their micro:bit Smart Garden to the next level can explore the radio communications built-in to the micro:bit device. By adding code that uses the radio communications, students could send the temperature and soil moisture readings from one micro:bit (located in the garden, for example) to a "master" micro:bit located in their home or in the classroom.

We have not included specific instructions in this Challenge to include the radio communications, however we do have a Mini DT Challenge on Python Networking to show how it's done: find out more at https://aca.edu.au/resources/python-microbit-radio/

## Next Steps

For further resources that might extend, enrich or deepen the classroom experience or support assessment, teachers can visit the Digital Technologies Hub at https://www.digitaltechnologieshub.edu.au.