Visual to text coding

LESSON 6: **Magic 8 Ball**

# (Top summary)

**Go to Lesson Series** |  **Go to next lesson**

This is the sixth in a series of lessons to transition from visual coding to text-based coding with a general-purpose programming language.

Included videos can be used by a teacher and/or students to see how to code each of the simple programs step-by-step in three programming languages: Scratch, Python and JavaScript.

This lesson may take two to three 45-minute periods. It builds on the previous introduction to **arrays** (also called **lists**) and brings in the **length** property.

# Curriculum links

Links with Digital Technologies Curriculum Area

| **Strand** | **Year** | **Content Description** |
| --- | --- | --- |
| Processes and Production Skills | 5–6 | Design algorithms involving multiple alternatives (branching) and iteration [(AC9TDI6P02)](https://v9.australiancurriculum.edu.au/f-10-curriculum.html/learning-areas/digital-technologies/year-5_year-6/content-description?subject-identifier=TECTDIY56&content-description-code=AC9TDI6P02&detailed-content-descriptions=0&hide-ccp=0&hide-gc=0&side-by-side=1&strands-start-index=0&subjects-start-index=0&view=quick). |
| 7–8 | Design algorithms involving nested control structures and represent them using flowcharts and pseudocode [(AC9TDI8P05)](https://v9.australiancurriculum.edu.au/f-10-curriculum.html/learning-areas/digital-technologies/year-7_year-8/content-description?subject-identifier=TECTDIY78&content-description-code=AC9TDI8P05&detailed-content-descriptions=0&hide-ccp=0&hide-gc=0&side-by-side=1&strands-start-index=0&subjects-start-index=0&view=quick).  Trace algorithms to predict output for a given input and to identify errors [(AC9TDI8P06)](https://v9.australiancurriculum.edu.au/f-10-curriculum.html/learning-areas/digital-technologies/year-7_year-8/content-description?subject-identifier=TECTDIY78&content-description-code=AC9TDI8P06&detailed-content-descriptions=0&hide-ccp=0&hide-gc=0&side-by-side=1&strands-start-index=0&subjects-start-index=0&view=quick). |

# Assessment

Students undertake a self-reflection of the programming task. The teacher can use the completed self-assessments to assist in summative assessment.

* Download the self-assessment sheet in Word or PDF format.

In assessing code in languages like Python or JavaScript, consider a rubric that brings in important skills for general-purpose programming.

* Download a sample rubric in Word or PDF format.

# Learning hook



[jennycepeda/Pixabay](https://pixabay.com/photos/doctor-team-medicine-2568481/)

Imagine you’ve been tasked with building a new online tool called *Doctor Internet*.

If someone is feeling unwell, they can answer a few simple questions about symptoms and *Doctor Internet* will identify their illness and produce a prescription to take to the chemist.

As a class, discuss:

1. What data would this system need access to?

**Sample answers:**

* + Medical database of diseases, their symptoms and prescriptions
  + Patient health background data
  + The system would need to be kept up to date with new information

1. What are the risks posed by this system? What could go wrong?

**Note:** There are numerous very high risks inherent in this system.

**Sample answers:**

* Access to individual health data is a privacy risk and the source of much controversy worldwide.
* The number of factors influencing a diagnosis is high, making the coding an enormous challenge.
* Misdiagnoses could lead to injury or death! These could happen due to lack of information, poor design, or even bugs in the programming.
* Legally, who is responsible if a misdiagnosis is given?
* Would users trust such a system? Trust in large-scale digital systems, such as social media networks, has fallen in recent years.

1. Is it possible to reduce or mitigate the risks?

**Sample answers:**

* The system might be used by medical professionals for the purpose of additional advice.
* Due to having access to a large database of diseases, the system might alert a medical professional to something they otherwise missed.

1. Do solutions like this exist in the real world?

**Sample answer:**

These solutions were much hyped, particularly in the 1980s, for business and finance applications. Such solutions were called **expert systems**. [This Wikipedia article](https://en.wikipedia.org/wiki/Expert_system) explains that the concept went out of favour in the 1990s, but the conditional logic then became integrated into other, broader solutions.

# Learning map and outcomes

In this lesson, students will:

1. access an online programming environment for visual code (**Scratch**) and for general-purpose programming (**Python** or **JavaScript**)
2. practise the new concept of the array **length** **property**
3. explore the design of a Magic 8 Ball decision-maker that gives random advice for a given question from the user
4. plan and code the Magic 8 Ball.

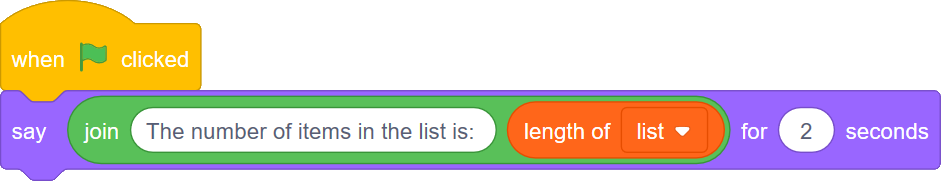
# Learning input

1. Begin by watching the video introducing the length property.

[INSERT JASON VEARING VIDEO **14. LENGTH PROPERTY**]

The video only demonstrates JavaScript.

* Scratch has an orange block to do the job ([try code](https://scratch.mit.edu/projects/309362654/)):



* Python has a simple command, len() ([try code](https://repl.it/@digitechhub/Show-array-length)):

words = ['dark', 'ghastly', 'light', 'bones', 'monkey']

no\_of\_items = len(words)

print('The number of items in the array is', no\_of\_items)

1. Carefully read the pseudocode below, then type up and test the program in Scratch as well as in JavaScript or Python.

BEGIN

**colours** ← [‘red’, ‘yellow’, ‘green’, ‘purple’, ‘orange’, ‘blue’]

Display “I know this many colours: ”, length of **colours**

Display “The first colour I know is ”, **colours**[0]

Display “The second colour is ”, **colours**[1]

Display “The last colour is ”, **colours**[length of **colours** – 1]

END

1. Finally, add a couple of extra colours to the array, then run your program again. The last colour in the array should still be displayed correctly.

SIDEBAR – Positions in arrays

|  |  |  |
| --- | --- | --- |
| Remember, in Scratch, items in a list are numbered **starting from 1**. |  | But in Python, list elements are numbered **starting from 0**. The first entry ‘Canberra’ has position 0. The last entry ‘Rome’ has position 4.  capitals = ['Canberra', 'Paris',  'London', 'Kinchasa',  'Rome']  print(capitals[4])  This program displays “Rome”.  JavaScript array elements are also numbered **starting from 0**.  var capitals = ['Canberra', 'Paris',  'London', 'Kinchasa',  'Rome'];  alert(capitals[4]); |

# Learning construction

**Step 1: Setup**

For more on setting up and choosing a language, see [Setting Up](https://www.digitaltechnologieshub.edu.au/teachers/lesson-ideas/visual-to-text-coding-setting-up/).

**Step 2: Solution development**

[This video](https://youtu.be/V4WCpVg73d0?list=PL0dMM9mg1u6NKQNT1odqe-5dI-lEovDxm) demonstrates coding a simple decision maker program, then expands it to a Magic 8 Ball program.

Choose two of the links below (eg Scratch and Python) to find the simple decision-maker programs. As in the video, build on those programs so that they now include all of the [standard 20 Magic 8 Ball responses](https://en.wikipedia.org/wiki/Magic_8-Ball).

* [Scratch](https://scratch.mit.edu/projects/290110374/fullscreen/)
* [Python](https://repl.it/@digitechhub/decisionmaker)
* [JavaScript](https://jsfiddle.net/nathanesa/duw9cL6o/)

*Solution code:* [*Scratch*](https://scratch.mit.edu/projects/309386845/)*,* [*Python*](https://repl.it/@digitechhub/Magic-8-Ball)*,* [*JavaScript*](https://jsfiddle.net/nathanesa/ek2d4mxw/)

**Step 3: Tinker task**

Edit the program to make your Magic 8 Balloptimistic or pessimistic. After asking the first question, the program should ask the user if they want an optimistic or pessimistic response (‘o’ or ‘p’).

If the user chooses ‘o’, the program chooses randomly from among the first 10 responses. If the user chooses ‘p’, the program chooses randomly from among the last five responses.

*Solution code:* [*Scratch*](https://scratch.mit.edu/projects/309389763/)*,* [*Python*](https://repl.it/@digitechhub/Lesson-6-Tinker-Task)*,* [*JavaScript*](https://jsfiddle.net/nathanesa/eso26bv8/)

# Challenge

These challenges use the skills covered so far. By writing or modifying their own programs, students have an opportunity to demonstrate Application and Creation.

1. Create a program help a teacher pick a random student in the class. The program will choose and display a student name from an array of 20 or more names.

Back in [Lesson 3: Challenge 1](https://www.digitaltechnologieshub.edu.au/teachers/lesson-ideas/visual-to-text-coding-lesson-3-heads-or-tails/#challenge), we used ‘if-else’ code to make a decision based on our random number, but that was useful only for a small number of names. Now, you can start with an array that includes all the names, so there’s no need for if-else code.

Remember, your program must still work perfectly if the array is different, for example, if you add or remove an extra student name.

*Solution code:* [*Scratch*](https://scratch.mit.edu/projects/311404587/)*,* [*Python*](https://repl.it/@digitechhub/Lesson-6-Challenge-1)*,* [*JavaScript*](https://jsfiddle.net/nathanesa/2qyjmbfr/)

1. Create a program to tell you the country where you can find a famous landmark.

The program should contain two arrays, one with landmarks and another with their countries. The order of each array must match. For example:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Landmarks** | |  | **Countries** | |
| 0 | Eiffel Tower |  | 0 | France |
| 1 | Big Ben |  | 1 | UK |
| 2 | Borobudur |  | 2 | Indonesia |
| 3 | Sydney Opera House |  | 3 | Australia |

Start by displaying the complete array of landmarks. The user then selects one by entering a number. Finally, the correct country is displayed.

An example of how the program should appear:

Here are the landmarks you can choose from...

Eiffel Tower, Big Ben, Borobudur, Sydney Opera House

Enter number to learn the country (first is 0): **2**

Borobudur is in Indonesia.

Challenge early finishers to make the program more user friendly. User can now enter ‘1’ for the first landmark, instead of ‘0’.

*Solution code:* [*Scratch*](https://scratch.mit.edu/projects/311430300/)*,* [*Python*](https://repl.it/@digitechhub/Lesson-6-Challenge-2)*,* [*JavaScript*](https://jsfiddle.net/nathanesa/axsm1309/)

1. (Optional) Create a program to sort a list of TV shows into alphabetical order.

Start with an array of 10 or more TV show names, not in alphabetical order. Display the array as it is.

Both Python and JavaScript come with simple **sort** commands to sort an array.

* In Python:

capitals = ['Canberra', 'Paris', 'London', 'Kinchasa', 'Rome']

capitals.sort()

* In JavaScript:

var capitals = ['Canberra', 'Paris', 'London', 'Kinchasa', 'Rome'];

capitals.sort();

Once sorted, display the array again to see the result.

# Resources

* Online environments for coding in each language
  + [Scratch](https://scratch.mit.edu/)
  + [repl.it](https://repl.it/): an online environment suited to Python
  + [JSFiddle](https://jsfiddle.net/): an online environment suited to JavaScript
* Cheat sheets listing basic commands for coding:
* Python 3 [cheatsheet](https://groklearning-cdn.com/resources/cheatsheet-python-1.pdf) (from Grok Learning)
* JavaScript [CheatSheet](https://htmlcheatsheet.com/js/) (Tip: Press the little blue tabs to move **Variables**, **Basics**, **Strings** and **Data Types** to the top.)