# Visual to text coding Lesson 5: **Password generator**

*Please refer to the online lesson plan on the DT Hub to access all website links and additional resources.*

This is the fifth 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 beginner teacher and/or students to see how to code each of the simple programs step-by-step in all three languages: Scratch, Python and JavaScript.

This lesson may take two to three 45-minute periods. It introduces how to create and use **arrays** (also called **lists**).

# 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) . |
| 7–8 | Design algorithms involving nested control structures and represent them using flowcharts and pseudocode (AC9TDI8P05) .  Trace algorithms to predict output for a given input and to identify errors (AC9TDI8P06) . |

# 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



1. Many experts advise that passwords should not be re-used for different sites and services.

In pairs, try to estimate how many different sites and services you currently use that require a login. These might include:

* email
* banking
* social media
* games
* shopping
* school
* personal device access
* Wi-Fi.

WARNING: For security reasons, students should **never** share passwords with each other, nor should they disclose the method they personally use for managing or remembering their passwords*.*

1. Next, consider which of those services you would consider to be **important**. That is, if someone else were to gain access, they would have key identity, health or financial information about you.

As a class, discuss the dangers of using the same password for two different services. What might be a solution, and what are the pros and cons?

1. This clip from ABC’s *The Checkout* discusses the problem of remembering passwords, and one solution suggested by some experts.

# 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 creating and using arrays (also called lists)
3. explore the design of a password generator that combines three words, with each word chosen randomly from a separate list
4. plan and code the password generator.

# Learning input

1. Begin by watching the video introducing arrays.
2. Examine this pseudocode carefully, then write the output of the program, that is, exactly what it will display on screen:

BEGIN

**mountains** ← [‘Mount Everest’, ‘Mount Kosciuszko’, ‘Mont Blanc’]

**continents** ← [‘Asia’, ‘Australia’, ‘Europe’]

**heights** ← [8848, 2228, 4809]

Display “Here’s some information about ”, **mountains**[2]

Display “It can be found on the continent of ”, **continents**[2]

Display “Its height in metres is ”, **heights**[2]

Display “But the tallest mountain in the world is ”, **mountains**[0]

END

**SOLUTION:**

Here’s some information about Mont Blanc

It can be found on the continent of Europe

Its height in metres is 4809

But the tallest mountain in the world is Mount Everest

SIDEBAR – Positions in arrays

**Scratch:** In Scratch, items in a list are numbered **starting from 1**

**Python:**

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:** JavaScript array elements are also numbered **starting from 0**.

var capitals = ['Canberra', 'Paris',

'London', 'Kinchasa',

'Rome'];

alert(capitals[4]);

It is easy to forget that the position of the last item is 1 less than the array’s length. **Off-by-one** bugs are among the most common coding bugs, even for experienced programmers.

# Learning construction

For more on setting up and choosing a language, see Lesson 1.

**Step 1: Solution demonstration**

This video demonstrates coding a simple decision maker program, then expands it to a Magic 8 Ball program.

**Step 2: Coding the program**

This video demonstrates coding the solution in Scratch, Python and JavaScript. Refer to the video to help code the program. Create the program using your preferred programming language. Check the solution code to help trace errors.

Solution code: Scratch, Python, JavaScript

**Step 3: Tinker task**

Modify your password generator program to incorporate more word choices. There should be 10 adjectives, 8 nouns and 12 numbers.

Next, add an array of special characters (‘$’, ‘!’, ‘#’, ‘%’, ‘\*’) and choose one of these at random to join to the end of the generated password.

Lastly, edit your program so that it asks the user at the beginning if they want to include the special character or not.

Solution code: Scratch, Python, JavaScript

# 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 to generate and display information about a fantasy game character.
2. Write out the algorithm in pseudocode first.
3. If it helps, code the program in Scratch before going on to Python or JavaScript.

Choose the following basics at random from pre-made lists:

* + a first name (eg Ugbar, Clint, Luglug, Rohan, Marielle, Elinore)
  + a surname (eg Rizrad, Deepforge, Ravenspell)
  + a race (eg human, dwarf, troll, elf).

Choose some stats as random numbers between 1 and 15:

* strength
* intelligence
* dexterity
* charisma.

Challenge early finishers to improve the program. Here are some suggestions for improvements:

* Use specialised name lists depending on the race of the character.
* Add race bonuses (eg trolls can add +3 to their strength value).
* Develop a system for standardising characters. If the user got very unlucky with all their stat values, they should be boosted, or the player can roll again.

Solution code: Scratch, Python, JavaScript

1. Create a gameshow challenge where the player must choose one of three doors to win a prize. They can choose Door no. 0, Door no. 1 or Door no. 2.

The prizes behind the doors are already set by creating an array. For example:

|  |  |
| --- | --- |
| 0 | a lifetime supply of gum ointment! |
| 1 | a new car! |
| 2 | a board game to take home and play with the family! |

The player chooses a door simply by entering the number 0, 1 or 2, then their prize is displayed. Note: if using Scratch, the door numbers can be 1, 2 and 3.

Challenge early finishers to improve the Python or JavaScript version so that the player can enter 1, 2 or 3 instead of 0, 1 and 2 without changing the array.

Solution code: Scratch, Python, JavaScript

1. (OPTIONAL) Create a different type of password generator that builds up from single letters, numbers and special characters.

Many online services have strict rules about passwords. For example:

“*Passwords should be 10 characters long. They should contain at least one character from each of the following sets: lowercase letters, uppercase letters, numerals and special characters. Passwords should not contain dictionary words or common names*.”

Your new password generator must follow the rules above. This does not require any new knowledge than for the password generator introduced in this lesson, but it will take more work to type out the contents of all four arrays.

For example:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **lowercase** | |  | **uppercase** | |
| 0 | a |  | 0 | A |
| 1 | b |  | 1 | B |
| 2 | c |  | 2 | C |
| 3 | d |  | 3 | D |
| … | … |  | … | … |

1. (Optional) Create a shopping list helper that uses an array. Prompt the user three times to type in an item, then display the complete list.

eg Enter the first item: **apples**

Enter the second item: **milk**

Enter the third item: **frozen vegies**

Your shopping list contains apples, milk and frozen vegies.

For this challenge, you could start with an array containing three “nothing” strings, then replace them with actual words.

Alternatively, you could investigate how to add items to an array that begins empty. See the Python and JavaScript cheat sheets in Resources below.

# Resources

* Setting up online environments
* Online environments for coding in each language
  + Scratch
  + repl.it : an online environment suited to Python
  + JSFiddle : an online environment suited to JavaScript
* Cheat sheets listing basic commands for coding:
  + Python Cheatsheet (from Grok Learning)
  + JavaScript CheatSheet (Tip: Press the little blue tabs to move Variables, Basics, Strings and Data Types to the top.)