# Visual to text coding LESSON 1: Temperature converter

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

**Year: 5-6, 7-8:** This is the first 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 **variables**, get **user input** and perform **maths operations**.

# 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://htmlcheatsheet.com/js/?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://repl.it/@JasonVearing/Currency-Converter?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 here in Word or PDF format.

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

* Download a sample rubric here in Word or PDF format.

# Learning hook



WikImages/pixabay

Did you know that NASA once lost a $125 million spacecraft because two teams were using different units of measurement?

Where do **you** look when you want to convert:

* inches into centimetres
* degrees Celsius (°C) into degrees Fahrenheit (°F)
* US dollars (USD) into Australian dollars (AUD)?

You may do an Internet search or use a relevant app. Do you think the code used in the program would be complicated? What might it include?

# Learning map and outcomes

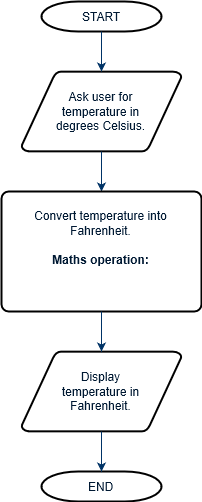
In this lesson, students will:

1. access an online programming environment for visual code (**Scratch**) and for general-purpose programming (**Python** or **JavaScript**)
2. learn basic programming skills to create variables and get user input
3. plan and code a program that converts degrees Celsius (°C) into degrees Fahrenheit (°F).

# Learning input

1. Step 1: As a class, or in teams, design the program as a flowchart. Students must research and fill in the **Maths operation**.

Image: Flow chart for temperature converter



1. Step 2: Once the flowchart is complete, write the program in pseudocode (structured English).

BEGIN

Display “Enter the temperature in degrees Celsius:”

**celsius** ← input from user

**fahrenheit** ← **celsius** × 9 ÷ 5 + 32

Display “The temperature is “, **fahrenheit**, “ degrees Fahrenheit.”

END

SIDEBAR – Starting pseudocode

Program designers use pseudocode (also called structured English) before coding an algorithm in a real, specific programming language.

The purpose of pseudocode is to clearly understand and communicate an algorithm, regardless of the final language used. Because it translates more readily into real code, it is often more popular than a flowchart.

Pseudocode has few strict rules, but here are some helpful hints to get started:

* Begin your algorithm with BEGIN. End it with END.
* Use a left-pointing arrow (←) to indicate assigning a value to a variable. For example:

**income** ← 5 × 12 The variable **income** will now contain the value 60.

At Years 7 and 8, the Australian Digital Technologies curriculum specifies: ‘Design algorithms involving nested control structures and represent them using flowcharts and pseudocode’ (ACTDIP029 ).

# Learning construction

For more on setting up and choosing a language, see *Setting up*.

To review concatenation, variables and user input and output, head back *The Basics* page.

**Step 1: Temperature converter**

It’s time to code our temperature converter. This will require storing what the user types in (user input). View the ‘Celsius converter’ video to learn how to create a program to convert temperature. This video covers the entire process of building the code. Solution code is provided for checking.

To help understand programming Temperature Converter, view this video on concatenation and this video on inputs and outputs. Final code: Scratch, Python, JavaScript

**Step 2: User-friendly output**

To make the program more user-friendly, use concatenation – joining text together – for a better output display. View the ‘Celsius converter (concatenation)’ video. This video covers the process of using concatenation to improve output. Solution code is provided for checking.

Final code: Scratch, Python, JavaScript

**Step 3: Tinker task**

Modify your temperature program to convert the other way – from Fahrenheit to Celsius.

Solution code: Python, JavaScript

# Challenge (optional)

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. Code a new program in Python or JavaScript that converts centimetres into inches.
   1. Write out the algorithm in pseudocode first.
   2. If it helps, code the program in Scratch before going on to Python or JavaScript.

Challenge early finishers to try more complex conversions, like capacity (square centimetres to litres) or battery capacity (amp hours to kilowatt hours for a given voltage).

Solution code: Python, JavaScript

1. Have a go at converting currency (eg USD to AUD). View the ‘Currency converter’ video. This video covers the process of creating a program to convert between USD to AUD. Solution code is provided for checking.

|  |
| --- |
| **Discussion:** For this conversion, the Google currency converter has an advantage over your own program. Why? Explain that:   * Online converters often have access to **live data** on currency conversion rates, which are constantly changing (unlike temperature or other measurement conversions). * Your program will likely need to rely on a fixed conversion rate for the calculation. |

Solution code: Scratch, Python, JavaScript

1. With more data, more complex calculations can be done. See the two videos below for more challenging ideas.

Solution code: Scratch, Python, JavaScript

# 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.)