DIGITAL TECHNOLOGIES HUB

Australian Curriculum V9.0 **Digital Technologies**

Programming

Implementation describes the automation of an algorithm, typically by writing a computer program or using appropriate software

ACARA, 2022

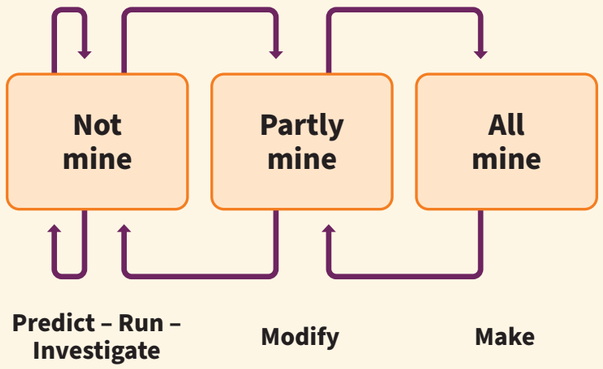
Teaching programming Years 3–10

View your year band for an overview of the content related to programming. In some cases it includes suggested related content so you can integrate with another learning area.

Use-Modify-Create is one approach to support and guide your students. A learner runs (uses) an existing program to see what it does, then modifies it and, when able, creates a new project of their own.

Pair programming is another useful pedagogy. Two students, a driver and a navigator, work side by side. The driver controls the programming actions while the navigator assists with instructions for programming, looks for errors and monitors progress against the algorithm. After some time the roles are reversed.

Use-Modify-Create adapted



For suggested resources

https://bit.ly/ProgrammingYears3to10

From Years 3–6 students are expected to use a visual programming language which is often referred to as block-based programming. From Years 7–10 students are expected to use a general purpose programming (GPP) language such as Python, JavaScript or C++.

Here are a range of resources and approaches that you might use to implement programming.

Learning resources and approaches	Visual programming	General purpose programming
Coding platforms and environments	Scratch, Snap, Blockly, micro:bit, Pencil code and Tynker	Replit, Glitch, JSFiddle, and CodePen, W3Schools, Swift (Apple's coding environment)
Online courses, tutorials and coding challenges	Code.org, Grok Academy, Scratch community	Grok Academy, W3Schools, Codeacademy, DT Hub and Khan academy
Project-based learning	micro:bit, Programmable robots, App inventor	Arduino, Raspberry pi, micro:bit, Programmable robots and robotic kits

Foundation

This concept does not appear in the Australian Curriculum: Digital Technologies in Foundation.

There is related content in Mathematics.

Use and interpret everyday language of location and direction, such as: between, near, next to, forward, toward, above, below, on top of, under.

Describe the position of an object compared to another object

Related content

Connect language of position and movement to students' lives.

Mathematics

Describe the position and location of themselves and objects in relation to other people and objects within a familiar space Mathematics | AC9MFSP02

Years 1–2

This concept does not appear in the Australian Curriculum: Digital Technologies in Years 1–2.

A pre-cursor to using a formal programming language is to use a push button programmable floor robot.

This enables students to use computational thinking, learn basic programming concepts and implement simple algorithms. (See Algorithms).

There is related content in Mathematics.

Provide relevant contexts to use locational and directional language such as quarter, half-turns, left and right, forwards and backwards, clockwise and anticlockwise, and use steps to describe distance.

Related content

Using a floor robot to move along a pathway.

Mathematics

Give and follow directions to move people and objects to different locations within a space | Mathematics AC9M1SP02

Locate positions in two-dimensional representations of a familiar space; move positions by following directions and pathways | Mathematics AC9M2SP02

DIGITAL TECHNOLOGIES HUB

Australian Curriculum V9.0 **Digital Technologies**

Programming

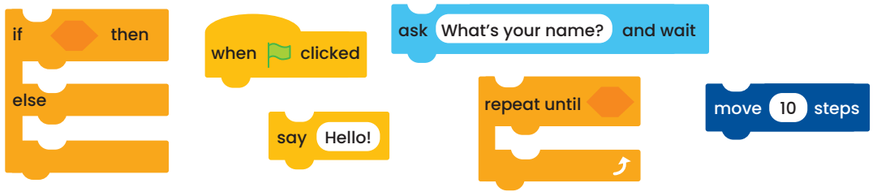
Implementation describes the automation of an algorithm, typically by writing a computer program or using appropriate software

ACARA, 2022

Years 3–4

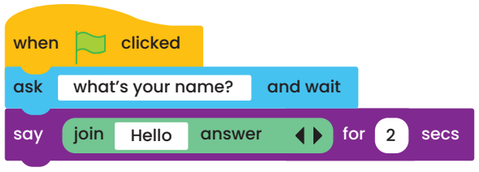
From a sequence of steps, I can make working visual programs that can take an input, repeat things and make decisions while running.

A **visual programming language** is represented and manipulated as graphic blocks. These graphic blocks can be composed to form programs for an algorithm.

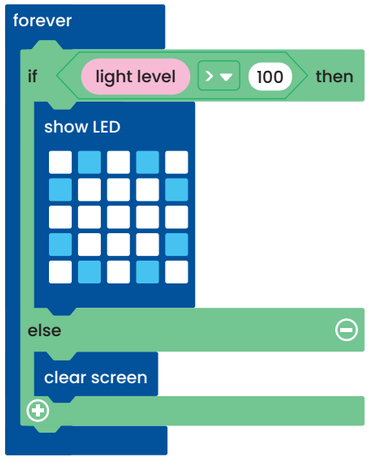


The program is a sequence of steps that includes decisions (**branching**) and specific steps that may be repeated (**iteration**).

An **input** enables the user to interact with the program. The 'ask' block enables the user to enter some input. The input can be any text string or number and the user's response 'answer' block is stored and can be used in the program.



Input can also be data from a sensor, for example, a temperature or light sensor.



Control structures are the code blocks or keywords that allow the program to make decisions, or to repeat parts. Some examples:

- A character performs an action IF the temperature is higher than 35 degrees.
- A robot keeps moving forward WHILE it is not touching an object.
- IF the light level is > 100 then show LEDs on screen. **Forever** loops can be used when sensing the environment.

Achievement standard	Students follow and describe simple algorithms involving branching and iteration and implement them as visual programs.
Content descriptions	Implement simple algorithms as visual programs involving control structures and input | AC9TDI4P04

Years 5–6

The visual programs I make now also include variables to remember data I want to use in the program.

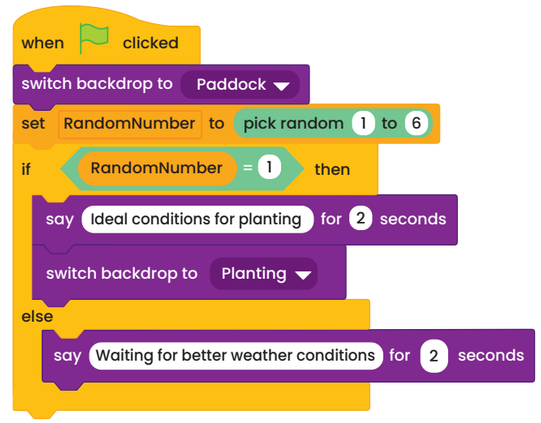
Branching allows for the creation of multiple paths within a program, which can be triggered by user **input** or other factors. Decisions can be made, such as IF-THEN blocks. Data including user input can be stored as **variables** and used throughout the program.

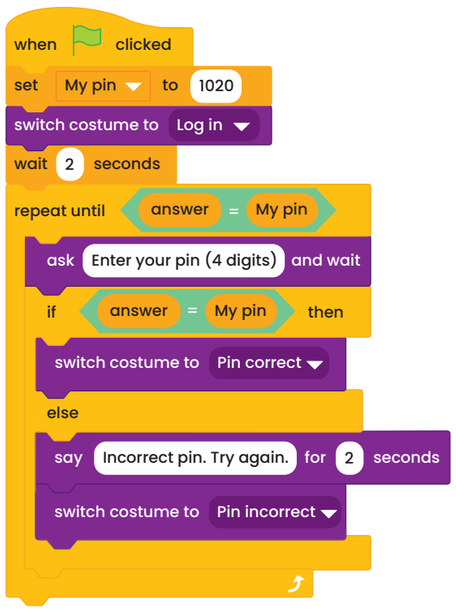
Creating and using a variable is like having a code 'bucket' to store a number or text.

Some examples of a **variable** include:

- guess (stores a player's guess for a secret number)
- pin (stores a four digit number)
- feedback (stores specific feedback to display for a correct or incorrect answer).

In this example a random number is stored as a variable. IF random number =1, then the farmer starts planting.





A REPEAT UNTIL block can be used when a certain action or set of actions will be executed repeatedly until a specific condition is met. Examples may include:

Repeat until:

- ANSWER = value
- SCORE is > 1000
- Temperature is < 20

In this example the program repeats until ANSWER = My pin (variable)

Achievement standard	Students design algorithms involving complex branching and iteration and implement them as visual programs including variables.
Content descriptions	Implement algorithms as visual programs involving control structures, variables and input | AC9TDI6P05

DIGITAL TECHNOLOGIES HUB

Australian Curriculum V9.0 **Digital Technologies**

Programming

Implementation describes the automation of an algorithm, typically by writing a computer program or using appropriate software

ACARA, 2022

Years 7–8

I can implement programs in a language like Python or JavaScript.
I can use functions to organise my code.

General purpose programming languages include Python and JavaScript, as well as others like C++, C#, Java, Perl and Swift.

This pseudocode is part of a simple login and shown in Python and JavaScript.

pseudocode

```
success ← false
WHILE NOT success DO
    username ← INPUT('Enter your username:')
    password ← INPUT('Enter your password:')
    IF username = storedUsername AND password = storedPassword THEN
        success ← true
    ELSE
        DISPLAY 'Invalid credentials. Try again.'
    ENDIF
ENDWHILE
DISPLAY 'Login successful.'
```

Python

```
stored_user = 'ari4382'
stored_pwd = 'r34%3!ab'
success = False
while not success:
    user = input('Enter your username:')
    pwd = input('Enter your password:')
    if username == storedUser and pwd == storedPwd:
        success = True
    else:
        print('Invalid credentials. Try again.')
print('Login successful.')
```

JavaScript

```
storedUser = 'ari4382'
storedPwd = 'r34%3!ab'
let success = false;
while (!success) {
    let user = prompt('Enter your username:');
    let pwd = prompt('Enter your password:');
    if (user == storedUser && pwd == storedPwd) {
        success = true;
    } else {
        console.log('Invalid credentials. Try again.');
    }
}
console.log('Login successful.');
```

Functions are lines of code that have been separated out as a group, so that they can be used and reused whenever needed.

For example, in the login program, you could use a function called **login** to handle the credentials checking. Then, this function can be called upon whenever a user needs to log into the system. Define the **login** function that takes the username and password as **parameters** and returns a login status. The program could also include multifactor authorisation using a function.

Debugging

Is your program behaving unexpectedly? Try these strategies for finding bugs:

- **pair coding** – two coders can be better than one
- **print values** – insert a temporary 'print' line in your code to output a value at the right time
- **step through code** – use your coding software to pause line-by-line as it runs your code

Achievement standard	Design and trace algorithms and implement them in a general-purpose programming language.
Content descriptions	Implement, modify and debug programs involving control structures and functions in a general purpose programming language | Digital Technologies AC9TDI8P09

Years 9–10

My programs are capable of structuring data and functionality in more organised and complex ways.

Modular programs have code across different sections or files. This allows:

- more flexibility
- stronger organisation
- easier collaboration with other programmers.

Data structures store related variables together in the most suitable ways.

For instance, a program can utilise a list or array to store a set of numbers or words. Once data is stored in structures like these, sorting algorithms can be written to place them in numerical or alphabetical order.

[130, 129, 160, 149, 165, 130, 129, 160, 149, 165, 171]

Main program

Main menu display

Specific functions or data

Employee data, add, update, delete

Extra functions

Generate Employee Report

Or it can utilise a record or object to store data, for example, all an employee's attributes – such as name, date of birth, salary and role.

A record or object stores this data.

Employee("Roberts", "Angie", 1995, 5, 4, 86000.00, "manager", false)

Object oriented programming languages (OOP) allow programs to be structured in a particular way, where data is stored together with the actions to be performed on it. Programming languages like Python and JavaScript already include this capability.

In OOP, an "EndangeredSpecies" class can have attributes like name and population, and methods like "increasePopulation" and "decreasePopulation" to manage and track the species' status.

Numbat, endangered species

Achievement standard	Design and validate algorithms and implement them, including in an object-oriented programming language.
Content descriptions	Implement, modify and debug modular programs, applying selected algorithms and data structures, including in an object oriented programming language | Digital Technologies AC9TDI10P09