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.

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.

<table>
<thead>
<tr>
<th>Learning resources and approaches</th>
<th>Visual programming</th>
<th>General purpose programming</th>
</tr>
</thead>
<tbody>
<tr>
<td>Coding platforms and environments</td>
<td>Scratch, Snap, Blockly, micro:bit, Pencil code and Tynker</td>
<td>Repl.it, Glitch, JSfiddle, and CodePen, W3Schools, Swift (Apple’s coding environment)</td>
</tr>
<tr>
<td>Online courses, tutorials and coding challenges</td>
<td>Code.org, Grok Academy, Scratch community</td>
<td>Grok Academy, W3Schools, Codecademy, DT Hub and Khan academy</td>
</tr>
</tbody>
</table>

**Related content**

Connect language of position and movement to students’ lives.

**Mathematics**

Connect language of position and movement to students’ lives.

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

Locate positions in two-dimensional representations of a familiar space; move positions by following directions and pathways Mathematics | AC9MR2SP02

Find more resources at [www.dthub.edu.au](http://www.dthub.edu.au)
A visual programming language is represented and manipulated as graphic blocks. These graphic blocks can be composed to form programs for an algorithm.

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.

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.

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
- Repeat until: SCORE is > 1000
- Repeat until: Temperature is < 20

In this example the program repeats until ANSWER = My pin (variable)
Implementation describes the automation of an algorithm, typically by writing a computer program or using appropriate software.

**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.

```python
# Pseudocode for a login program
stored_user = 'ari4382'
stored_pwd = 'r34%3!ab'

success = False

while not success:
    user = input('Enter your username: ')  # 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.
    password = input('Enter your password: ')

    if user == stored_user and password == stored_pwd:
        success = True
        print('Login successful. ')
        break
    else:
        print('Invalid credentials. Try again.')

print('Login unsuccessful.')
```

```javascript
// JavaScript pseudocode for a login program
storedUser = 'ari4382'
storedPwd = 'r34%3!ab'

success = false

while (!success) {
    user = prompt('Enter your username: ');
    password = prompt('Enter your password: ');

    if (user == storedUser && password == storedPwd) {
        success = true;
        console.log('Login successful.');
        break;
    } else {
        console.log('Invalid credentials. Try again. ');
    }
}
```

**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

**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.

![List of heights](image)

<table>
<thead>
<tr>
<th>Heights</th>
<th>129cm</th>
<th>130cm</th>
<th>149cm</th>
<th>150cm</th>
<th>160cm</th>
<th>165cm</th>
<th>171cm</th>
</tr>
</thead>
<tbody>
<tr>
<td>Employee</td>
<td>Roberts</td>
<td>86000.00</td>
<td>1995</td>
<td>5</td>
<td>4</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**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.

** 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

**Extra functions**

Generate Employee Report

**Specific functions or data**

Main program

- Main menu display
- Employee data, add, update, delete

**Main menu**

- Generate Employee Report
- Report
- Employee data, add, update, delete
- Extra functions
- Main menu display

**Salary:** $86,000

**DOB:** 4/5/1995

**Role:** manager

**On leave:** no

**Employee:** Roberts, “Angie”, 1995, 5, 4, 86000.00, “manager”, false

**Numbat, endangered species**

Digital Technologies Hub is brought to you by the Australian Government Department of Education. Creative Commons Attribution 4.0, unless otherwise indicated.