Computational and   
algorithmic thinking   
in Mathematics

Unpacking the content descriptions

Level 7



Authorised and published by the Victorian Curriculum and Assessment Authority  
Level 7, 2 Lonsdale Street  
Melbourne VIC 3000

© Victorian Curriculum and Assessment Authority 2019

No part of this publication may be reproduced except as specified under the *Copyright Act 1968* or by permission from the VCAA. Excepting third-party elements, schools may use this resource in accordance with the [VCAA educational allowance](http://victoriancurriculum.vcaa.vic.edu.au/Curriculum/ContentDescription/VCCCTQ003#schools). For more information go to: <https://www.vcaa.vic.edu.au/Footer/Pages/Copyright.aspx>.

The VCAA provides the only official, up-to-date versions of VCAA publications. Details of updates can be found on the VCAA website: [www.vcaa.vic.edu.au](https://www.mathplayground.com/grade_1_games.html).

This publication may contain copyright material belonging to a third party. Every effort has been made to contact all copyright owners. If you believe that material in this publication is an infringement of your copyright, please email the Copyright Officer: [vcaa.copyright@edumail.vic.gov.au](mailto:vcaa.copyright@edumail.vic.gov.au)

Copyright in materials appearing at any sites linked to this document rests with the copyright owner/s of those materials, subject to the Copyright Act. The VCAA recommends you refer to copyright statements at linked sites before using such materials.

The VCAA logo is a registered trademark of the Victorian Curriculum and Assessment Authority.

Contents

[Explicitly teaching computational and algorithmic thinking 4](#_Toc23334146)

[Overview of the resource 5](#_Toc23334147)

[Programming patterns 7](#_Toc23334148)

[Additional teaching resources 21](#_Toc23334149)

[Appendix 1 23](#_Toc23334150)

[Appendix 2 24](#_Toc23334151)

[Appendix 3 26](#_Toc23334152)

Explicitly teaching computational and   
algorithmic thinking

The *Computational and algorithmic thinking – Unpacking the content descriptions* resources unpack the Victorian Curriculum F–10 Mathematics content descriptions that address computational thinking and algorithms at each level in the Patterns and algebra sub-strand of the Number and Algebra strand.

Each resource provides teachers with links between one Mathematics content description and extract from the achievement standard related to computational and algorithmic thinking and a teaching and learning activity that is designed to develop computational thinking and problem-solving skills and produce corresponding algorithms in a mathematical context. Teachers can also find excerpts from the Victorian Curriculum Mathematics and Digital Technologies glossaries in [Appendix 3](#Appendix3).

The resources have been developed with respect to teaching in the Mathematics learning area of the Victorian Curriculum and they also include suggestions how these activities could be extended to the Critical and Creating Thinking and Digital Technologies curriculums.

Teachers will find detailed ideas about how to integrate Mathematics with one or both of Digital Technologies and Critical and Creating Thinking in [Appendix 1](#Appendix1) and [Appendix 2](#Appendix2) respectively.

Overview of the resource

**Curriculum area and level:** Mathematics, Level 7

**Strand and sub-strand:** Number and Algebra, Patterns and algebra

**Content description:** Design and implement mathematical algorithms using a simple general purpose programming language ([VCMNA254](http://victoriancurriculum.vcaa.vic.edu.au/Curriculum/ContentDescription/VCMNA254))

**Achievement standard (extract):** Students develop … models for situations, make predictions based on these models, solve related equations and check their solutions.

**Title:** Programming patterns

**Timing:** Each activity could be completed separately or as a sequence:

Activity 1: 1 or 2 lessons (approx. 50–100 minutes)

Activity 2: 2 lessons (approx. 100 minutes)

Activity 3: 2 or 3 lessons (approx. 100–150 minutes)

**Description:** Students use a range of technologies and programming languages to construct geometric patterns.

The options included in this lesson cater to a variety of experience with simple programming languages and the use of dynamic geometry software (DGS). Students will be able to use one or a combination of:

Wolfram Language (Mathematica)

Python (Python Turtle)

Javascript

GeoGebra online.

**Learning objectives:** Students can:

use dynamic geometry software to create geometric shapes and patterns

define a tessellation and create an example using DGS

use a general-purpose programming language such as Python (using the Python Turtle), Javascript or Wolfram Language (Mathematica) to create geometric shapes and patterns

define the term ‘fractal’ and create examples using DGS (optional)

investigate the applications of fractals in nature (optional).

**Resources:** Each student or pair will need a computer and access to the internet, or preinstalled software of one or more of GeoGebra, Mathematica or an equivalent (such as handheld or computer versions of Ti-Nspire or CASIO Classpad, Geometers’ Sketchpad).

**Tip:** The websites and software used in this activity are either free to use online or licensed for use by the Department of Education and Training (DET).

Programming patterns

Scaffold and support

Additional vocabulary/revision

Teachers are encouraged to use and clearly define the terms ‘coding’, ‘platform’, ‘programming language’, ‘implementation’ and ‘algorithm’ in this activity (see the [Victorian Curriculum Mathematics Glossary](http://victoriancurriculum.vcaa.vic.edu.au/LearningArea/LoadFile?learningArea=mathematics&subject=mathematics&name=Mathematics%20Glossary.docx&storage=Glossary) or [Appendix 3](#Appendix3)). Teachers could choose to define these terms themselves for the class or give students the opportunity to come up with their own agreed class definitions before, during or after the activity and compare them with the definitions in the glossary.

Some key vocabulary and background information have been included below.

**Algorithm**

An algorithm is a process that can be carried out mechanically, using a well-defined set of instructions, to perform a particular task or solve a type of problem. Examples of mathematical algorithms include processes for tasks such as ordering a set of numbers from smallest to largest, multiplying many-digit decimal numbers, factorising linear expressions, determining which of two fractions is larger, bisecting an angle, or calculating the mean of a set of numbers.

**Dynamic geometry software (DGS)**

This type of software provides an interactive digital environment for students to explore geometry. DGS allows students to easily and quickly construct and manipulate (using a mouse or touchscreen) geometrical objects and identify relationships between them. Examples of DGS include Cabri Geometry, Geometers’ Sketchpad, Mathematica and GeoGebra.

**Fractal**

Objects that have the same structures at different scales. This quality of being able to zoom in or out and still see the same structure is called ‘self-similarity’. The [Sierpinkski Triangle](https://fractalfoundation.org/resources/fractivities/sierpinski-triangle/) is an example of a fractal.

**Implementation**

Implementation is the process of translating an algorithm into a language.

**Pattern**

A repeated geometric design (created with shapes or objects) or a recurring sequence (of objects, shapes or numbers) arranged according to a rule.

**Platform**

The platform is the means by which an algorithm is implemented. This may be a mechanical device, a program running on a computer using a particular programming language, or an activity carried out by a person or robot.

**Programming**

A process by which algorithms are represented for implementation. For computers, this is done using a language such as C++, JavaScript, Python, Wolfram Language.

**Programming language**

Any language used to provide instructions for a computer to follow. Programming languages can be used to create computer programs which automate a given algorithm. Many programming languages exist, including Python, C++ and Fortran. Using a programming language to create a program is called coding.

Software and online resources

Wolfram software

Wolfram software is internationally recognised as a powerful STEM learning tool. It has applications in the areas of computation, problem modelling and programming. Victoria’s Department of Education and Training has a licence for the following Wolfram products:

* Wolfram Mathematica
* Wolfram Alpha Pro
* SystemModeler
* Mathematica Online
* Wolfram Programming Lab.

The licence covers these programs being used in Victorian schools. The programs can be used on school computers, teacher devices and student devices.

For access and further information, visit the Department of Education and Training’s [Wolfram software](https://www.education.vic.gov.au/about/programs/learningdev/vicstem/Pages/wolframsoftware.aspx) webpages.

Teacher instructions

In the following activities, students use a range of technologies and programming languages to construct geometric patterns.

Each student or pair will need a computer and access to the internet, or preinstalled software of one or more of:

* GeoGebra
* Mathematica
* or equivalent (such as handheld or computer versions of Ti-Nspire or CASIO Classpad, Geometers’ Sketchpad).

Activity 1 – Patterns and tessellations (technology-free)

Students explore the concept of tessellations. Once students have explored this, they can recreate tessellations and other geometric patterns using technology.

Explore and define

Define the following terms for students either directly or through student-led research and feedback within small groups and to the class:

* **Polygon**: a two-dimensional shape with straight sides, such as a triangle or rectangle. A circle, for example, is not a polygon.
* **Regular** **polygons**: shapes with all equal sides and angles at each vertex (corner). For example, the first four regular polygons are the equilateral triangle, square, regular pentagon and regular hexagon.



Polygons are (usually) named for the number of sides they have (prefix in **bold**):

|  |  |  |  |
| --- | --- | --- | --- |
| **Shape** | **Number of sides** | **Shape** | **Number of sides** |
| **Tri**angle | 3 | **Hept**agon/**Sept**agon | 7 |
| **Quad**rilateral | 4 | **Oct**agon | 8 |
| **Pent**agon | 5 | **Non**agon | 9 |
| **Hex**agon | 6 | **Dec**agon | 10 |

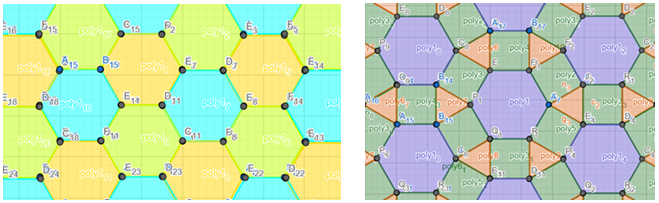
* **Irregular polygons**: shapes with not all equal sides and angles at each vertex (corner). For example, the following are irregular polygons:
* **Concave** and **convex** **polygons**: Polygons may be described as concave if they have any interior angles greater than 180° (any sides that ‘cave in’) and convex if all interior angles are less than 180°. The triangle below could be described as a regular convex triangle. The second shape is an example of an irregular concave hexagon.

Triangle and irregular concave hexagon

**Tip:** Students may work through this [Maths is fun](https://www.mathsisfun.com/geometry/polygons.html) resource on polygons.

* **Tessellation** (tiling): A tessellation is a repeated pattern in the plane or on a surface where shapes completely fill all the space around a given point where their boundaries meet. For example, a honeycomb is a tessellation using hexagons. Tiling patterns are tessellations using rectangular tiles or brick pavers in paths, mosaics in buildings, quilts and art.

A regular tessellation is created by tessellating regular polygons. If more than one regular polygon is used, it is a semi-regular tessellation. Examples of a regular tessellation of hexagons, and a semi-regular tessellation of triangles, hexagons and squares, are below.

Geo

*Images created using GeoGebra*

Create a visual display

Students should create a visual display to summarise the different types of polygons. Encourage students to find polygons in the world around them and/or online and to collect photos or images and to classify the polygons accordingly.

**Discussion prompts (small groups or whole class)**

* What are the most common polygons you see in buildings and other large manufactured structures (bridges, cranes etc.)? Why do you think these might be seen so often?
* Which shapes in your images are not polygons? Why?
* Can you find any examples of tessellations? What kind of tessellation are they? Why?

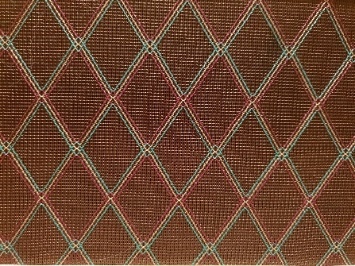
► **Cross-curricular links**

See [Appendix 1](#Appendix1) and [Appendix 2](#Appendix2) for ways to link this activity to the explicit teaching of Digital Technologies and/or Critical and Creative Thinking.

Activity 2 – Drawing using dynamic geometry software

In this activity, students use dynamic geometry software to create geometric shapes.

**Tip:** The DGS discussed in this lesson are GeoGebra and Mathematica, but any equivalent system could be used.

Students create their own patterns or find patterns online or in the world around them to replicate. Some examples of geometric patterns student could recreate are below:

GeoGebra

Students access [GeoGebra](https://www.geogebra.org/geometry) and use it do draw geometric shapes. In addition, teachers and students can explore [student and teacher resources](https://www.geogebra.org/materials). Students can view this [online video](https://youtu.be/vx3DrQvcNeI) demonstrating how to draw basic shapes in GeoGebra.

**Tip:** Students can sign up to GeoGebra for free. Students will be able to save their creations if they have a login.

Mathematica

Victorian schools can access Mathematica using the licence held by the Department of Education and Training. For access and further information, visit the Department of Education and Training’s [Wolfram software](https://www.education.vic.gov.au/about/programs/learningdev/vicstem/Pages/wolframsoftware.aspx) webpages.

Run this activity in two parts. Students may work in pairs or individually.

1. Students recreate basic polygons using software and label these using the text feature, such as ‘convex regular hexagon’. Students can save their work if they are logged in or capture their work using a screenshot.
2. Students undertake research online and create a folio of geometric patterns and tessellations (the patterns don’t all have to be tessellations). Their folio could include drawings and photographs if time permits. Encourage students to *recreate* their patterns using the chosen software. Remind students to save their work or take a screenshot.

At the end of the lesson, students print their work to create a classroom demonstration of what can be created using dynamic geometry software.

**Discussion prompts (small groups or whole class)**

* Are there any shapes that are more difficult to recreate then others? Why?
* What are the advantages of drawing geometric shapes using this kind of software? What are the disadvantages?
* What type of polygons are in your design?
* Are there tessellations that don’t work? Why? What is it about the shapes used that prevent you tiling the plane?

► **Cross-curricular links**

See [Appendix 1](#Appendix1) and [Appendix 2](#Appendix2) for ways to link this activity to the explicit teaching of Digital Technologies and/or Critical and Creative Thinking.

Challenge and extend

Students can visit the following pages for inspiration for GeoGebra creations:

* [Islamic art](https://www.geogebra.org/m/aaech4za#material/sszxptns)
* [spiralling circle](https://www.geogebra.org/m/UmFd3AC7)
* [quilt pattern](https://www.geogebra.org/m/Y5YUyTSg)
* [tangram](https://www.geogebra.org/m/zrGERP88).

Students with coding experience can experiment using the Wolfram computer language to draw shapes. See here for [instructions](https://www.wolfram.com/language/fast-introduction-for-math-students/en/geometry/) on how to use Mathematica’s programming language to create geometric shapes, and use these to develop patterns as shown in [Wolfram Demonstrations Project](https://demonstrations.wolfram.com/).

Activity 3 – Coding geometric shapes

In this activity, students will be coding geometric shapes and patterns using a simple programming language.

Depending on students’ experience, they may wish to code using either Python with the Python Turtle (simpler interactive lessons) or Javascript (interactive lessons with feedback and activities). Students might also like to try both coding languages and compare the two for ease of use or output.

**Tip:** Remind students that writing code is an algorithmic process because they are creating a logical sequence of steps (the coded instructions) to reach an outcome (the geometric shapes or pattern).

Coding with Python (1 or 2 lessons)

Python is a very useful and widely used general-purpose programming language.

The following interactive activity introduces students to the Python language through Python Turtle:

* Coding shapes with the [Python Turtle](https://codeclubprojects.org/en-GB/archived/turtle-power/)

As for Activity 2, students recreate basic polygons using Javascript. Students are then encouraged to recreate geometric patterns using the Javascript. Remind students to save their work or take a screenshot.

At the end of the activity, students print their work to produce a classroom demonstration of what can be created using dynamic geometry software.

**Discussion prompts**

* Are there any shapes that are more difficult to recreate then others? Why?
* What are the advantages of drawing geometric shapes using this code? What are the disadvantages?
* How does coding this way differ from using the dynamic geometry software? Are there advantages? Are there disadvantages?
* What aspects of the Python language are familiar to you? Why?
* What was difficult about using this language for you? What did you find easier?
* What strategies did you use to overcome any difficulties you had with these activities?

Coding with Javascript using Khan Academy (2 or 3 lessons)

Look at [Intro to Javascript: Drawing and Animation course](https://www.khanacademy.org/computing/computer-programming/programming). These lessons are interactive with a short instructional video followed by an online activity with feedback and solutions. Students can work through the three sections:

* Intro to programming
* Drawing basics
* Coloring.

**Tip:** Students may like to sign up (free) to be able to save their progress.

Students create geometric shapes including circles, squares, lines, triangles and change outlines and colours.

This [video](https://www.youtube.com/watch?v=HESALw_8HX8) demonstrates how to draw and colour basic shapes using Javascript and the the Khan Academy coding platform.

As for Activity 2, students recreate basic polygons using Javascript. Encourage students to recreate geometric patterns using the Javascript. Remind students to save their work or take a screenshot.

At the end of the lesson, have students print their work to produce a classroom demonstration of what can be created using dynamic geometry software.

**Discussion prompts**

* Are there any shapes that are more difficult to recreate then others? Why?
* What are the advantages of drawing geometric shapes using this code? What are the disadvantages?
* How does coding this way differ from using the dynamic geometry software? Are there advantages? Are there disadvantages?
* What aspects of the Python language are familiar to you? Why?
* What was difficult about using this language for you? What did you find easier?
* What strategies did you use to overcome any difficulties you had with these activities?

► **Cross-curricular links**

See [Appendix 1](#Activity1) and [Appendix 2](#Activity2) for ways to link this activity to the explicit teaching of Digital Technologies and/or Critical and Creative Thinking.

Challenge and extend – Fractals

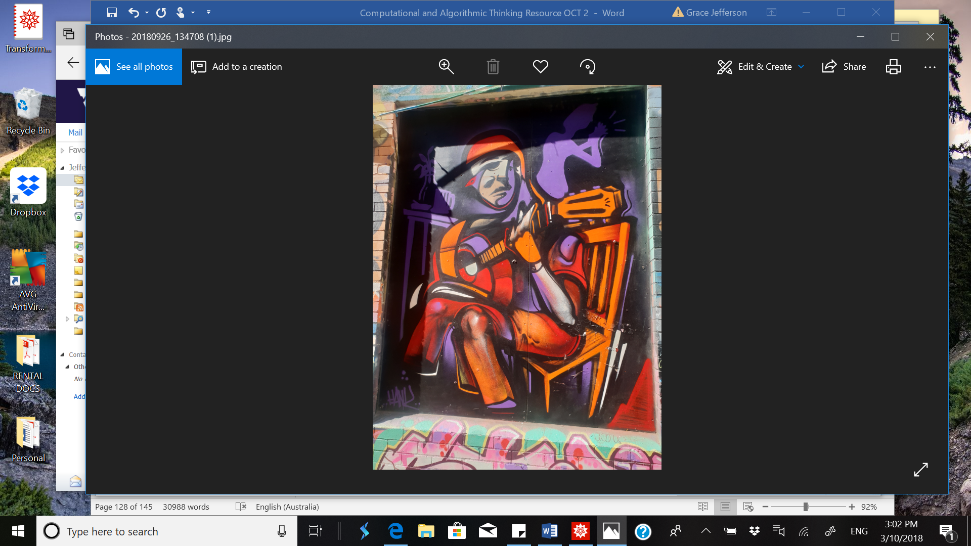
Fractals are objects that have the same structures at different scales. This quality of being able to zoom in or out and still see the same structure is called ‘self-similarity’. For example, consider the Sierpinski Triangle:

This is a repeating pattern of triangles. It can be recreated by taking an equilateral triangle, scaling down the original image to a smaller triangle half its original height, placing this new triangle in each corner of the larger triangle and ‘cutting out’ the remaining area. This process is repeated each time to create the pattern. This pattern could also be created using other shapes.

The video [Measuring the Australian Coastline](http://education.abc.net.au/home#!/media/154882/) (ABC Splash) is a great starter video to introduce students to the idea of fractals.

Students research to find a fractal pattern to recreate using any of the DGS or coding they have used in the other activities. There are plenty of fractals in nature (found online or out in the world) such as the baobab or boab tree (see picture at [ABC News](https://www.abc.net.au/news/2018-08-07/boabs-come-africa-baobabs-evolution/10060946)).

[Artful Math](https://artfulmath.com/fun-math/fractals-in-nature/) has more examples of fractals. Students may wish to provide a brief description of their chosen fractal (history, why it is a fractal, etc.) to accompany their digital recreation.

Challenge and extend – Art, technology and geometry

Students could extend this investigation by researching an artwork and using their DGS to recreate the image. They could also provide some background on the artist and artwork, and reflect on the method they chose for their work, difficulties faced, and strategies for overcoming these.

Reflection

Students should share and compare their experiences of using both the dynamic geometry software and coding with others.

* Was software easier to use?
* Does one type of software have advantages over another for different uses (e.g. is one easier for drawing triangles but harder for moving shapes?)

With the class, in small groups or individually, ask students to reflect on their algorithmic thinking.

Ask:

* How do the coding algorithms (processes) work in each activity?
* What does the code do well? Why?
* What doesn’t it do so well? Why?
* What do you think was the purpose of this investigation?

Also explore the problem-solving aspect of this task and ask the students to discuss the challenges they met and discuss different strategies that they used to overcome these. Ask:

* What were some challenges you faced in this task? How did you overcome these (what were some strategies)?
* What were some skills you learnt to help in a similar situation?
* Can you think of other applications for using dynamic geometry software or coding to create shapes?

**Tip:** Reflection is an important aspect of any computational-thinking focused activity because it encourages students to consider the different aspects of the task, such as defining the problem and breaking the task down, selecting tools and processes, problem-solving, teamwork and verifying their solution. This helps students reflect on the process of their own learning (meta-learning) and how the skills they have used might transfer to other contexts.

Additional teaching resources

* [Code.org](https://code.org/educate/algebra) has partnered with Bootstrap to develop online interactive lessons that explore algebraic and geometric concepts through computer programming. The two ten-hour courses from Code.org focus on concepts like order of operations, the Cartesian plane, function composition and definition, and solving word problems.
* [Code by math](https://codebymath.com/index.php/welcome/lesson_menu) is a series of interactive coding lessons exploring a range of mathematical concepts including testing divisibility, generating random numbers and drawing cartesian shapes.
* [Grok Learning](https://groklearning.com/course/maths-yr7/) offers a Python programming course which covers material from the Australian Curriculum: Mathematics Year 7 and explores connections between mathematics and computing. The first two lessons, Programming basics: Talking to your computer and Programming basics: Calculating things, can be accessed for free.
* [How Stuff Works](https://science.howstuffworks.com/math-concepts/fractals.htm) explores fractals and has a ‘How to make your own fractal’ activity.
* [Microsoft](https://education.microsoft.com/Story/Lesson?token=qoxA8) has a lesson that explores fractals, focusing on the Koch Curve and Koch Snowflake.
* [Maths is fun](https://www.mathsisfun.com/sierpinski-triangle.html) explores how to create a Sierpinski Triangle.

Appendix 1

Suggestions for explicitly teaching Digital Technologies (stimulus only)

**Curriculum area:** Digital Technologies

**Strand:** Creating Digital Solutions

**Band:** Levels 7 and 8

**Content descriptions:**

* Design algorithms represented diagrammatically and in English, and trace algorithms to predict output for a given input and to identify errors ([VCDTCD042](https://victoriancurriculum.vcaa.vic.edu.au/technologies/digital-technologies/curriculum/f-10#level=7-8&search=c2ce161e-d5d7-4277-b73a-babc3a4387ce))
* Develop and modify programs with user interfaces involving branching, iteration and functions using a general-purpose programming language ([VCDTCD043](https://victoriancurriculum.vcaa.vic.edu.au/technologies/digital-technologies/curriculum/f-10#level=7-8&search=307a03f5-1d2c-4e6d-91dd-27fdec0e3d34))

**Achievement standard (extract):** They design … algorithms incorporating branching and iterations, and develop, test, and modify digital solutions.

**Suggestions that extend Activities 1–3 :**

* Comparing algorithms with statements in a:
* visual programming language
* general-purpose programming language
* Creating an algorithm as English statements for a common task where decisions and repetition are made, for example entering in a class set of test scores
* Tracing algorithms to check accuracy, predicting output based on given input (desk-checking) and identifying any errors
* Transforming simple algorithms into programs using a nominated general-purpose programming language
* Creating programs that incorporate all three control structures (sequence, branching and iteration)
* Developing programs where the output is dependent on user input

See also [Unpacking Digital Technologies Content Descriptions](https://www.vcaa.vic.edu.au/Documents/viccurric/digitech/Unpacking_the_Content_Descriptions/Unpacking_Digital_Technologies_Content_Descriptions-7-8.docx) for Levels 7 and 8.

Appendix 2

Suggestions for explicitly teaching Critical and Creative Thinking (stimulus only)

**Curriculum area:** Critical and Creative Thinking

**Strand:** Questions and possibilities

**Band:** Levels 7 and 8

**Content description:** Synthesise information from multiple sources and use lateral thinking techniques to draw parallels between known and new solutions and ideas when creating original proposals and artefacts ([VCCCTQ034](http://victoriancurriculum.vcaa.vic.edu.au/Curriculum/ContentDescription/VCCCTQ034))

**Achievement standard (extract):** Students demonstrate flexibility in thinking by using a range of techniques in order to repurpose existing ideas or solutions to meet needs in new contexts.

**Suggestions that link to the activities:**

* Examining a range of sources, such as Islamic art and quilt patterns, as inspiration for creating their own geometric designs
* Reflecting on the range of sources used and other contexts where the same approach could be taken

**Curriculum area:** Critical and Creative Thinking

**Strand:** Reasoning

**Band:** Levels 7 and 8

**Content description:** Examine how to select appropriate criteria and how criteria are used in clarifying and challenging arguments and ideas ([VCCCTR039](http://victoriancurriculum.vcaa.vic.edu.au/Curriculum/ContentDescription/VCCCTR039))

**Achievement standard (extract):** They explain and apply a range of techniques to test the strength of arguments.

**Suggestions that link to the activities:**

* Discussing how criteria can be used in mounting arguments about the advantages and disadvantages of different design tools and what should be considered when selecting criteria for this purpose

**Curriculum area:** Critical and Creative Thinking

**Strand:** Meta-Cognition

**Band:** Levels 7 and 8

**Content descriptions:**

* Consider a range of strategies to represent ideas and explain and justify thinking processes to others ([VCCCTM040](http://victoriancurriculum.vcaa.vic.edu.au/Curriculum/ContentDescription/VCCCTM040))
* Examine a range of learning strategies and how to select strategies that best meet the requirements of a task [(VCCCTM041)](https://victoriancurriculum.vcaa.vic.edu.au/Curriculum/ContentDescription/VCCCTM041)

**Achievement standard (extract):** Students use a range of strategies to represent ideas and explain and justify thinking processes to others. They evaluate the effectiveness of a range of learning strategies and select strategies that best meet the requirements of a task.

**Suggestions that link to the activities:**

* Identifying different ways to represent ideas, such as drawings and mock-ups, and discussing their different applications and strengths or weaknesses (evaluation)
* Following the completion of activities students reflect on how they completed the activities and the process of their own learning, including what their best strategies might be if they encountered other contexts (see the [Tip](#Tip) in the Reflection section)

Appendix 3

Excerpts from the Victorian Curriculum Mathematics and Digital Technologies glossaries

There are some commonalities in the terms used when explicitly teaching computational and algorithmic thinking in Mathematics and Digital Technologies; however, there are also some subtle but important differences in the definitions of terms. Some of these common terms and their definitions are listed below, under the two different curriculum areas.

If you are going to explicitly teach Mathematics, please refer to the [Victorian Curriculum Mathematics Glossary](https://victoriancurriculum.vcaa.vic.edu.au/LearningArea/LoadFile?learningArea=mathematics&subject=mathematics&name=Mathematics%20Glossary.docx&storage=Glossary). If you are also going to explicitly teach Digital Technologies, refer to the [Victorian Curriculum Digital Technologies Glossary](https://victoriancurriculum.vcaa.vic.edu.au/LearningArea/LoadFile?learningArea=technologies&subject=digital-technologies&name=Digital%20Technologies%20Glossary.docx&storage=Glossary).

Mathematics

Algorithm

An **algorithm** is a process that can be carried out mechanically, using a well-defined set of instructions, to perform a particular task or solve a type of problem. Examples of mathematical algorithms include processes for tasks such as ordering a set of numbers from smallest to largest, multiplying many-digit decimal numbers, factorising linear expressions, determining which of two fractions is larger, bisecting an angle, or calculating the mean of a set of numbers.

Algorithmic thinking

**Algorithmic thinking** is the type of thinking required to design, test and evaluate problem-solving processes in a systematic way, using algorithms.

Coding

A process by which algorithms are represented for implementation. For computers, this is done using a coding language such as block coding, C++, JavaScript, Python, Wolfram Language.

Computational thinking

In this context, computational thinking is considered to be linked to algorithmic thinking. This type of thinking is usually considered specific to computers which involves solving problems, designing systems and implementation.

Sequence (number)

A **sequence** is an ordered set of elements such as numbers, instructions or objects. From an algorithmic point of view, a sequence is an ordered set of instructions or actions.

Unplugged

A commonly used term for computational thinking activities carried out without digital technology. “Unplugged” representations of algorithms may include structured mathematical processes, English representations (steps) or flowcharts.

Digital Technologies

Algorithm

A description of the steps and decisions required to solve a problem. For example, to find the largest number in a list of positive numbers:

1. Note the first number as the largest
2. Look through the remaining numbers, in turn, and if a number is larger than the number found in 1, note it as the largest.
3. Repeat this process until complete. The last noted number is the largest in the list.

Flowcharts are often useful in visualising an algorithm.

Computational thinking

A problem-solving method that involves various techniques and strategies in order to solve problems that can be implemented by digital systems, such as organising data logically, breaking down problems into components, and the design and use of algorithms, patterns and models.