

LEGO® Education WeDo 2.0 Computational Thinking

Teacher's Guide



WeDo 2.0

Table of Contents

**Introduction to WeDo 2.0
Computational Thinking**

3-11

WeDo 2.0 in Curriculum

12-24

Assess with WeDo 2.0

25-37



Developing Computational Thinking with WeDo 2.0 Projects

In this chapter, you will discover how you can use WeDo 2.0 to develop computational thinking skills in a science context.





Develop Computational Thinking with LEGO® Education WeDo 2.0 projects

LEGO® Education is pleased to present these projects, which have been specifically designed for use in primary school classrooms to develop students' computational thinking skills.

Computational thinking is a set of skills that everybody can use to solve everyday life problems. In WeDo 2.0, these skills are developed throughout each phase of every project. Development opportunities have been identified for you in each of the projects, it is up to you to focus on the ones that are most relevant to you and your students.

Every project in WeDo 2.0 combines the use of the LEGO® bricks with an iconic programming language, enabling your students to find solutions to problems while being introduced to programming principles.

WeDo 2.0 develops computational thinking through coding activities, which bring students' creations to life, generating smiles and the desire to discover more.





Computer Science, Computational Thinking, Coding

While the science and engineering fields originated in the early ages of humankind, computer science has a much younger history. Nevertheless, this young discipline has influenced not only the way we approach science and engineering, but also the way we live our lives.

Computer Science is a STEM discipline, sharing attributes with science, technology, engineering, and mathematics.

All STEM disciplines present opportunities to develop a mindset and a lifelong set of practices. Among these practices, we find the ability to ask questions, to design solutions, and to communicate results.

Computational thinking is another one of these practices. It is a way in which we think and it is a way in which everybody can solve problems.

Computational thinking can be described as a group of skills, one of these skills being algorithmic thinking. “Code” or “coding” can be used to describe the action of creating an algorithm.

Coding is therefore one vehicle by which to develop computational thinking within a STEM context.

STEM Disciplines

Science, Technology, Engineering, Mathematics,
Computer Science

Develop a mindset and life long set of practices

1. Ask questions and solve problems.
2. Use models.
3. Design prototypes.
4. Investigate.
5. Analyse and interpret data.
6. Use computational thinking.

- a. Decompose
- b. Abstract
- c. Think algorithmically (code)
- d. Evaluate
- e. Generalise

7. Engage in argument from evidence.
8. Obtain, evaluate, and communicate information.



What is Computational Thinking?

The expression “computational thinking” was first used by Seymour Papert, but Professor Jeannette Wing is known to have popularised the idea. She defined computational thinking as:

“the thought processes involved in formulating problems and their solutions so that the solutions are represented in a form that can be effectively carried out by an information-processing agent.” (Wing, 2011)

Computational thinking is used in various fields and situations, and we use it in our daily lives. Computational thinking skills are present in science, engineering, and mathematics. These skills can be defined as the following:

Decomposition

Decomposition is the ability to simplify a problem into smaller parts in order to ease the process of finding a solution. By doing so, the problem becomes easier to explain to another person, or to separate into tasks. Decomposition frequently leads to Generalisation.

Example: When going on vacation, the preparation (or project) can be separated into subtasks: booking the airfare, reserving a hotel, packing a suitcase, etc.

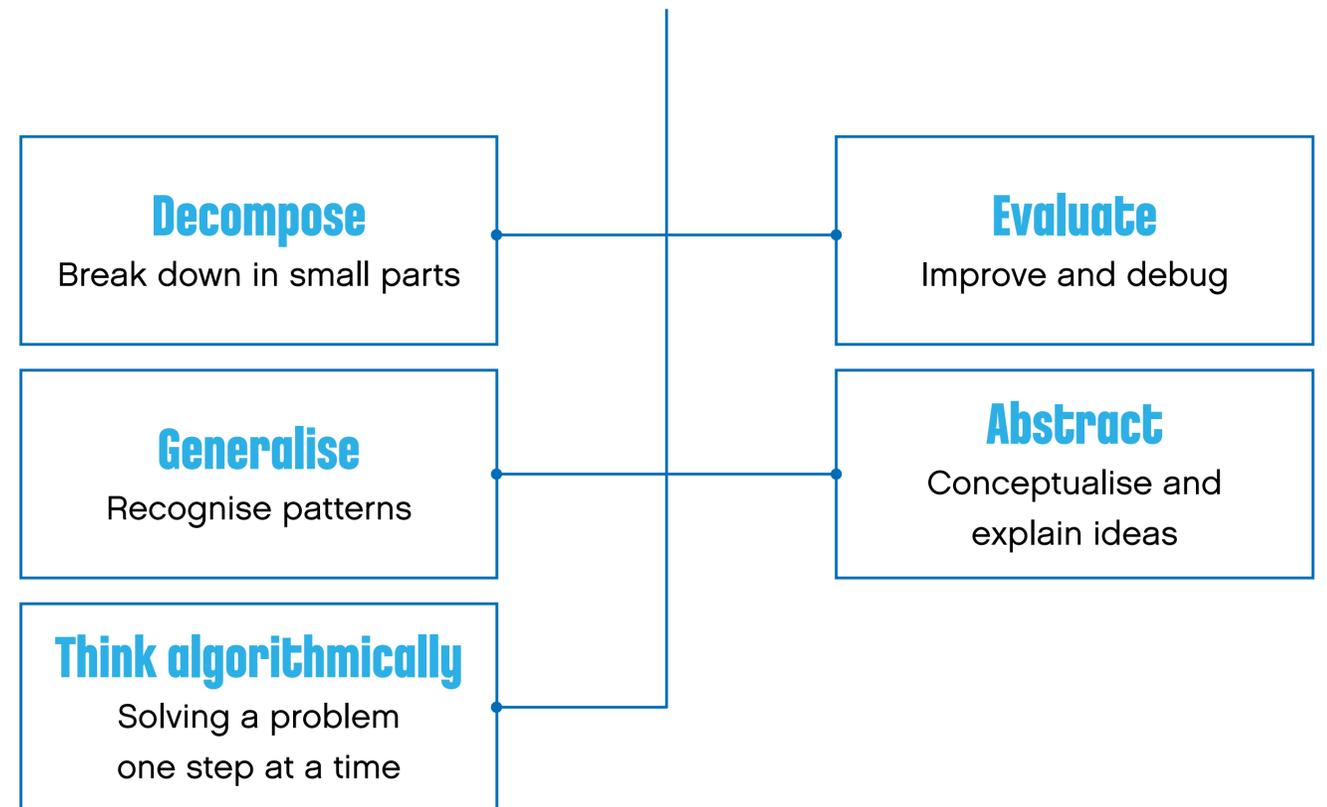
Generalisation (Pattern Recognition)

Generalisation is the ability to recognise the parts of a task that are known, or have been seen somewhere else. This frequently leads to easier ways of designing algorithms.

Example: Traffic lights work by repeating the same series of actions forever.

Computational thinking

Ways we solve problems





What is Computational Thinking?

Algorithmic Thinking

Algorithmic Thinking is the ability to create an ordered series of steps with the purpose of solving a problem.

Example one: when we cook from a recipe, we are following a series of steps in order to prepare a meal.

Example two: when playing with computers, we can code a sequence of actions that tell the computer what to do.

Evaluating or Debugging

This is the ability to verify whether or not a prototype works as intended, and if not, the ability to identify what needs to be improved. It is also the process a computer programmer goes through in order to find and correct mistakes within a program.

Example one: when we're cooking, we will periodically taste the dish to check whether or not it is seasoned correctly.

Example two: when we look for spelling mistakes and missing punctuation in our written work, we are debugging it so that it can be read correctly.

Abstraction

Abstraction is the ability to explain a problem or a solution by removing unimportant details. In other words, being able to conceptualise an idea.

Example: When describing a bicycle, we use only some details to describe it. We might mention its type and colour, and add more details for someone who has a real interest in bikes.



A process for developing Computational Thinking skills

Using an Engineering Design Process

When looking for solutions to a problem, engineers use a design process. They go through a series of phases that guide them toward a solution. During each of these phases, some of their skills are used or developed. It is those skills that we refer to as “computational thinking skills.”

In WeDo 2.0, students follow a similar process:

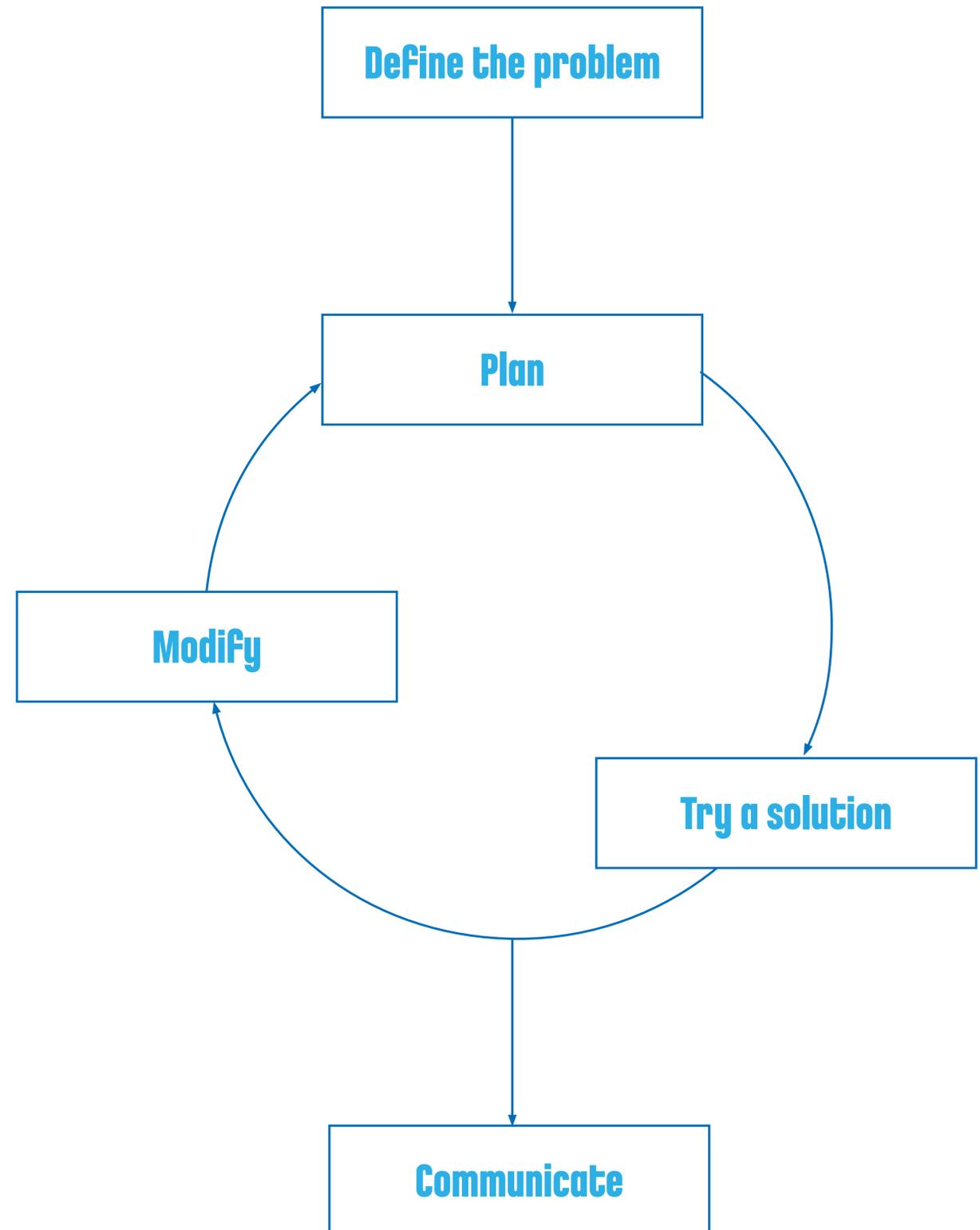
Defining the Problem

Students are presented with a topic that guides them to a problem or to a situation they wish to improve. Sometimes, a problem can have a lot of details. To make it easier to solve, the problem can be broken down into smaller parts.

By defining the problem in a simple way and by identifying some success criteria, students will develop a skill called “Decomposition.”

In other words:

- Is the student able to explain the problem by themselves?
- Is the student able to describe how they will evaluate whether or not they were successful in solving the problem?
- Is the student able to break down the problem into smaller and more manageable parts?





A process for developing Computational Thinking skills

Planning

Students should spend some time imagining different solutions to the problem, and then make a detailed plan for executing one of their ideas. They will define the steps they will need to go through in order to reach the solution. By identifying the parts of the task they might have seen before, they will develop a skill called “Generalisation.”

In other words:

- Is the student able to make a list of actions to program?
- Is the student able to identify parts of programs that he or she could use?
- Is the student able to reuse parts of programs?

Trying out

Each student is then tasked with creating the final version of their solution. In this phase of the process, they use iconic programming language to activate their LEGO® models. As the students code their ideas, they develop their Algorithmic Thinking skills.

In other words:

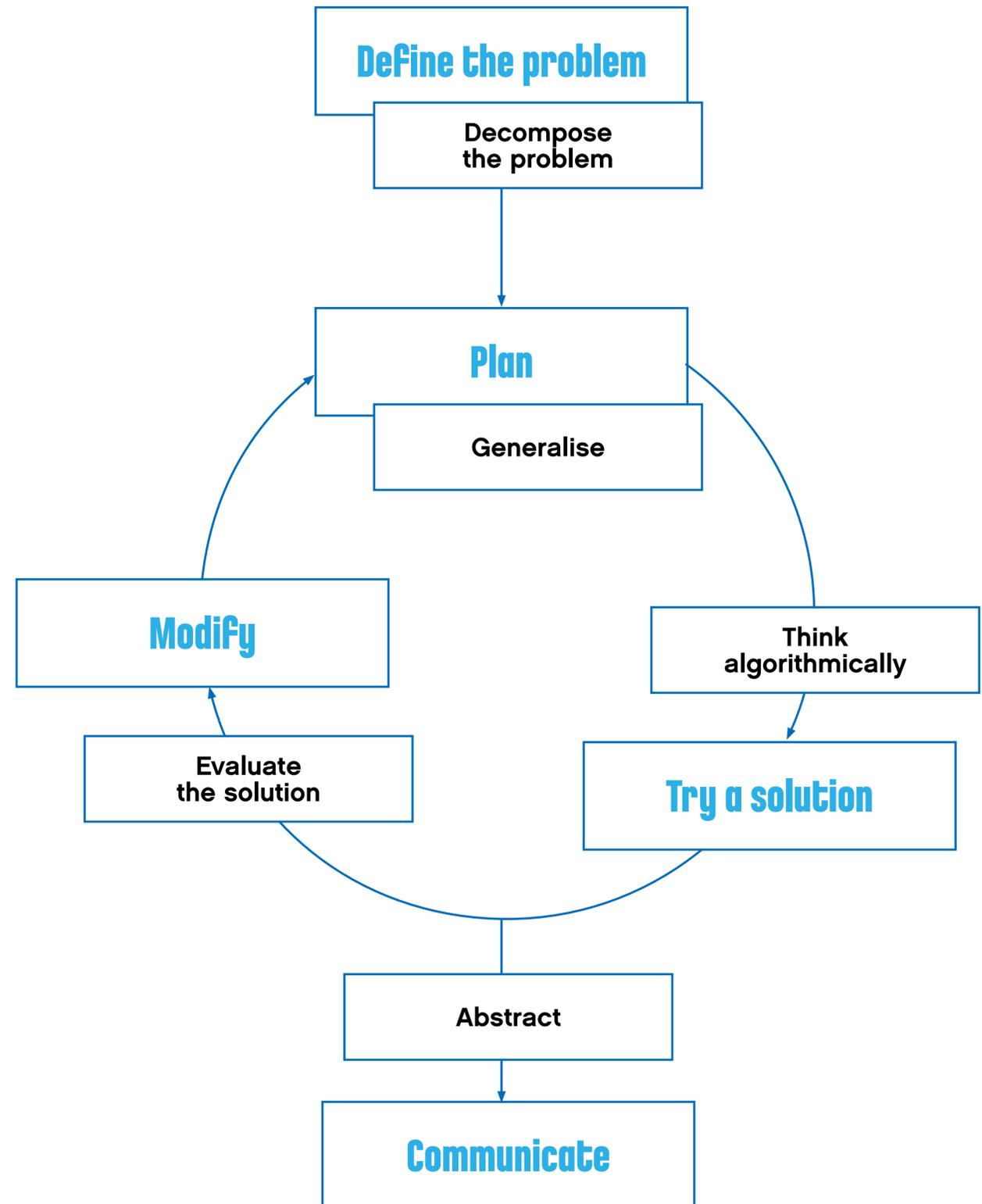
- Is the student able to program a solution to a program?
- Is the student able to use sequence, loops, conditional statements, etc.?

Modifying

Students will evaluate their solution according to whether or not their program and model meet the success criteria. Using their Evaluation skills, they will determine whether they need to change, fix, debug, or improve some part of their program.

In other words:

- Is the student making iterations of their program?
- Is the student fixing problems in their program?
- Is the student able to judge if the solution is linked to the problem?





A process for developing Computational Thinking skills

Communicating

Students will present the final version of their solution to the class, explaining how their solution meets the success criteria. By explaining their solution with the right level of detail, they will develop their Abstraction and communication skills.

In other words:

- *Is the student explaining the most important part of their solution?*
- *Is the student giving enough detail to enhance comprehension?*
- *Is the student making sure to explain how their solution meets the success criteria?*





Developing Computational Thinking through coding

In order to develop their Algorithmic Thinking, students will be introduced to some programming principles. As they develop their solutions, they will organise a series of actions and structures that will bring their models to life.

The most common WeDo 2.0 programming principles students will use are:

1. Output

Output is something that is controlled by the program students are writing. Examples of outputs for WeDo 2.0 are sounds, lights, display, and turning motors on and off.

2. Input

Input is information that a computer or device receives. It can be entered through the use of sensors in the form of a numeric or text value. For example, a sensor that detects or measures something (such as distance) converts that value into a digital input signal so it can be used in a program.

3. Events (Wait for)

Students can tell their program to wait for something to happen before continuing to execute the sequence of actions. Programs can wait for a specific amount of time, or wait for something to be detected by a sensor.

4. Loop

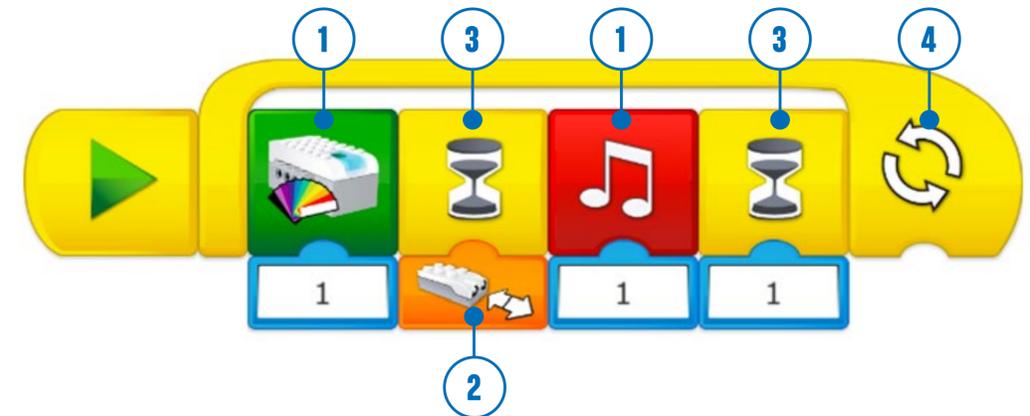
Students can program actions to be repeated either forever, or for a specific length of time.

5. Functions

Functions are a group of actions that are to be used together in specific situations. For example, the group of blocks that could be used to make a light blink would together be called, “the blink function”.

6. Conditions

Conditions are used by students in order to program actions that are to be executed only under certain circumstances. Creating conditions within a program means that some part of the program will never be executed if the condition is never met. For example, if the Tilt Sensor is tilted left, the motor will start, and if the sensor is tilted right, the motor will stop; if the Tilt Sensor never tilts left, the motor will never start and if it never tilts right, then the motor will never stop.



WeDo 2.0 in Curriculum

The LEGO® Education WeDo 2.0 projects combine LEGO® bricks with Australian Curriculum: Science. All of the WeDo 2.0 projects are designed to develop students' computational thinking skills.





Computational Thinking in curriculum

The world is changing, and whether we realise it or not, technology and computer science shape nearly every aspect of our lives. Students are rapidly becoming active citizens, and equipping them with the right set of skills has become one of the nation's first priorities.

Computational thinking is a set of skills that is spreading worldwide, becoming a key practice to develop in relation to technology. Already identified by the Australian Curriculum as a practice essential to the Australian Curriculum: Digital Technologies. Computational thinking has found roots in many other national curriculums around the world with an emphasis on the development of computational thinking skills.

These important skills can be developed through engaging activities or projects that are rooted in real life problem-based situations. To support this development, LEGO® Education is adding a dedicated series of computational thinking projects to the science projects that are already available in WeDo 2.0.



Visual overview of Guided Projects

1. Moon Base

This project is about designing a solution in which a robot would be able to assemble a base on the moon.

2. Grabbing Objects

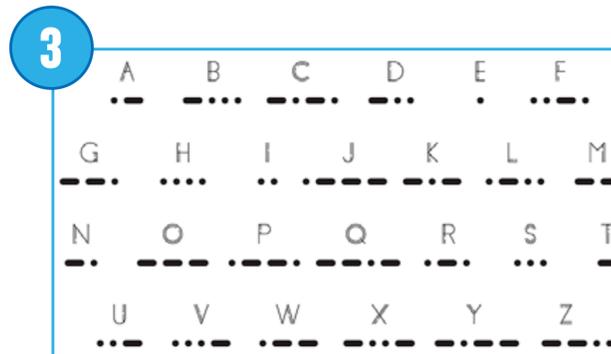
This project is about designing a solution for a prosthetic arm that is able to move small objects around.

3. Send Messages

This project is about designing a solution for exchanging information using a system of signals organised in patterns.

4. Volcano Alert

This project is about designing a device for improving the monitoring of volcanic activity in order to guide scientific exploration.





Visual overview of Open Projects

5. Inspection

This project is about designing a solution in which a robot is able to inspect narrow spaces, guiding its motion with sensors.

6. Emotional Design

This project is about designing a solution in which a robot can display positive emotions when interacting with people.

7. City Safety

This project is about designing a solution to improve safety in a city.

8. Animal Senses

This project is about modelling how animals use their senses to interact with their environment.

5



6



7



8





Potential Flow to develop Computational Thinking skills

You can organise the projects as you wish. Each project highlights opportunities for developing computational thinking skills, and it is up to you to focus on the ones that are most relevant to you and your students. Here is one suggested sequence, which is based on an increasing level of complexity in the programming concepts covered:

Getting Started

Use two lessons of 45 minutes each to introduce your students to WeDo 2.0.

Lesson 1, Milo, the Science Rover

Lesson 2, combine Milo's Motion Sensor, Milo's Tilt Sensor, and Collaborating

Guided Projects

Use two lessons of 45 minutes each, during which students will program a sequence of actions.

Lesson 3, Moon Base (Explore and Create phase)

Lesson 4, Moon Base (Test and Share phase)

Use two lessons of 45 minutes each, during which students will use sensors (inputs).

Lesson 5, Grabbing Objects (Explore and Create phase)

Lesson 6, Grabbing Objects (Test and Share phase)

Use two lessons of 45 minutes each, during which students will use sensors (inputs), loops, and parallel programming.

Lesson 7, Send Messages (Explore and Create phase)

Lesson 8, Send Messages (Test and Share phase)

Use two lessons of 45 minutes each to introduce your students to conditions, and how to integrate all of the other programming principles.

Lesson 9, Volcano Alert (Explore and Create phase)

Lesson 10, Volcano Alert (Test and Share phase)

Open Projects

Use two or three lessons of 45 minutes each to make your own project based on one of the suggested Open Projects. This project should integrate all of the programming principles, as well as the computational thinking skills developed during the Guided Projects.



Potential Flow to develop Computational Thinking skills

Getting Started

Introduce your students to WeDo 2.0



45 minutes

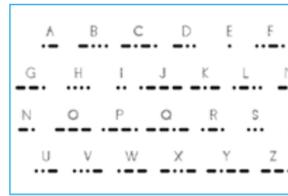


45 minutes



Guided Project - Send Messages

Students will use sensors (inputs), loops, and parallel programming.



Using a condensed lesson flow
2 x 45 minutes



Guided Project - Moon Base

Students will program sequences of actions.



Using a condensed lesson flow
2 x 45 minutes



Guided Project - Volcano Alert

Students will be introduced to conditions, and to other programming principles.



Using a condensed lesson flow
2 x 45 minutes



Guided Project - Grabbing Objects

Students will use sensors (inputs).



Using a condensed lesson flow
2 x 45 minutes



Open Projects





Australian Curriculum: Science Content Descriptors

Year 2

Science Inquiry Skills

Planning and conducting

- **AC SIS038** Participate in guided investigations to explore and answer questions

Communicating

- **AC SIS042** Represent and communicate observations and ideas in a variety of ways

Science as a Human Endeavour

Nature and development of science

- **AC SHE034** Science involves asking questions about, and describing changes in, objects and events

Year 3

Science Inquiry Skills

Planning and conducting

- **AC SIS054** With guidance, plan and conduct scientific investigations to find answers to questions, considering the safe use of appropriate materials and equipment

Communicating

- **AC SIS060** Represent and communicate observations, ideas and findings using formal and informal representations

Science as a Human Endeavour

Nature and development of science

- **AC SHE050** Science involves making predictions and describing patterns and relationships

Year 4

Science Inquiry Skills

Planning and conducting

- **AC SIS065** With guidance, plan and conduct scientific investigations to find answers to questions, considering the safe use of appropriate materials and equipment

Communicating

- **AC SIS071** Represent and communicate observations, ideas and findings using formal and informal representations

Science as a Human Endeavour

Nature and development of science

- **AC SHE061** Science involves making predictions and describing patterns and relationships

Year 5

Biological sciences

- **AC SSU043** Living things have structural features and adaptations that help them to survive in their environment

Science Inquiry Skills

Planning and conducting

- **AC SIS086**

Communicating

- **AC SIS093**

Science as a Human Endeavour

Nature and development of science

- **AC SHE081** Science involves testing predictions by gathering data and using evidence to develop explanations of events and phenomena and reflects historical and cultural contributions

Year 6

Science Inquiry Skills

Earth and space sciences

- **AC SSU096** Sudden geological changes or extreme weather conditions can affect Earth's surface

Planning and conducting

- **AC SIS103**

Communicating

- **AC SIS110**

Science as a Human Endeavour

Nature and development of science

- **AC SHE098** Science involves testing predictions by gathering data and using evidence to develop explanations of events and phenomena and reflects historical and cultural contributions

Use & influence of science

- **AC SHE100** Scientific knowledge is used to solve problems and inform personal and community decisions



Overview of Guided Projects organised by Science and Engineering Practices

	1 Moon Base	2 Grabbing Objects	3 Send Messages	4 Volcano Alert
Practice One: Ask questions and define problems	●	●	●	●
Practice Two: Develop and use models				
Practice Three: Plan and carry out investigations				
Practice Four: Analyse and interpret data				
Practice Five: Use mathematics and computational thinking	●	●	●	●
Practice Six: Construct explanations and design solutions	●	●	●	●
Practice Seven: Engage in argument from evidence	●	●	●	●
Practice Eight: Obtain, evaluate, and communicate information	●	●	●	●



Overview of Open Projects organised by Science and Engineering Practices

	5 Inspection	6 Emotional Design	7 City Safety	8 Animal Senses
Practice One: Ask questions and define problems	●	●	●	●
Practice Two: Develop and use models				●
Practice Three: Plan and carry out investigations				
Practice Four: Analyse and interpret data				
Practice Five: Use mathematics and computational thinking	●	●	●	●
Practice Six: Construct explanations and design solutions	●	●	●	
Practice Seven: Engage in argument from evidence	●	●	●	●
Practice Eight: Obtain, evaluate, and communicate information	●	●	●	●



Curriculum Overview of Projects organised by Australian Curriculum: Digital Technologies Content Descriptors

	Knowledge and Understanding				Process and Production Skills			
	Digital Systems	Representation of data	Collecting, managing and analysing data	Investigating and defining	Generating and designing	Producing and implementing	Evaluating	Collaborating and managing
	■ Year F-2 ■ Year 3-4 ■ Year 5-6							
1. Moon Base	■ ACTDIK001 ■ ACTDIK007 ■ ACTDIK014			■ ACTDIP004 ■ ACTDIP010 ■ ACTDIP017	■ ACTDIP019	■ ACTDIP011 ■ ACTDIP020		
2. Grabbing Objects	■ ACTDIK001 ■ ACTDIK007 ■ ACTDIK014			■ ACTDIP004 ■ ACTDIP010 ■ ACTDIP017	■ ACTDIP019	■ ACTDIP011 ■ ACTDIP020		
3. Send Messages	■ ACTDIK001 ■ ACTDIK007 ■ ACTDIK014	■ ACTDIK008 ■ ACTDIK015		■ ACTDIP004 ■ ACTDIP010 ■ ACTDIP017	■ ACTDIP019	■ ACTDIP011 ■ ACTDIP020		
4. Volcano Alert	■ ACTDIK001 ■ ACTDIK007 ■ ACTDIK014	■ ACTDIK008		■ ACTDIP004 ■ ACTDIP010 ■ ACTDIP017	■ ACTDIP019	■ ACTDIP011 ■ ACTDIP020		
5. Inspection	■ ACTDIK001 ■ ACTDIK007 ■ ACTDIK014			■ ACTDIP004 ■ ACTDIP010 ■ ACTDIP017	■ ACTDIP019	■ ACTDIP011 ■ ACTDIP020		
6. Emotional Design	■ ACTDIK001 ■ ACTDIK007 ■ ACTDIK014			■ ACTDIP004 ■ ACTDIP010 ■ ACTDIP017	■ ACTDIP019	■ ACTDIP011 ■ ACTDIP020		
7. City Safety	■ ACTDIK001 ■ ACTDIK007 ■ ACTDIK014			■ ACTDIP004 ■ ACTDIP010 ■ ACTDIP017	■ ACTDIP019	■ ACTDIP011 ■ ACTDIP020		
8. Animal Senses	■ ACTDIK001 ■ ACTDIK007 ■ ACTDIK014			■ ACTDIP004 ■ ACTDIP010 ■ ACTDIP017	■ ACTDIP019	■ ACTDIP011 ■ ACTDIP020		



Australian Curriculum: Digital Technologies Content Descriptors

Year F-2

Knowledge and Understanding

Digital Systems

ACTDIK001 Recognise and explore digital systems (hardware and software components) for a purpose

Process and Production Skills

Investigating and defining

ACTDIP004 Follow, describe and represent a sequence of steps and decisions (algorithms) needed to solve simple problems

Year 3-4

Knowledge and Understanding

Digital Systems

ACTDIK007 Identify and explore a range of digital systems with peripheral devices for different purposes, and transmit different types of data

Representation of data

ACTDIK008 Recognise different types of data and explore how the same data can be represented in different ways

Process and Production Skills

Investigating and defining

ACTDIP010 Define simple problems, and describe and follow a sequence of steps and decisions (algorithms) needed to solve them

Producing and implementing

ACTDIP011 Implement simple digital solutions as visual programs with algorithms involving branching (decisions) and user input

Year 5-6

Knowledge and Understanding

Digital Systems

ACTDIK014 Examine the main components of common digital systems and how they may connect together to form networks to transmit data

Representation of data

ACTDIK015 Examine how whole numbers are used to represent all data in digital systems

Process and Production Skills

Investigating and defining

ACTDIP017 Define problems in terms of data and functional requirements drawing on previously solved problems

Generating and designing

ACTDIP019 Design, modify and follow simple algorithms involving sequences of steps, branching, and iteration (repetition)

Producing and implementing

ACTDIP020 Implement digital solutions as simple visual programs involving branching, iteration (repetition), and user input

Assessing Computational Thinking Skills

There are many ways in which you can monitor and assess your students' progress through a WeDo 2.0 project.

This section offers the following tools to help you in your assessments:

- Documentation pages
- Self-assessment statements
- Anecdotal record grid
- Observation rubrics grid





Student-led assessment

Documentation Pages

Each project will ask students to create documents to summarise their work.

To have a complete science report, it is essential that students:

- Document their work using various types of media
- Document every step of the process
- Take the time to organise and complete their document

It is most likely that the first document your students will complete will not be as good as the next one. You can support them by:

- Giving feedback and allowing them time to see where and how they can improve some parts of their document.
- Allowing them to share their documents with each other. By communicating their scientific findings, students will be engaged in the work of scientists.

Self-Assessment Statements

After each project, students should reflect on the work they have done. Use the following page to encourage reflection and set goals for the next project.





Student self-assessment rubric

Name: _____

Class: _____

Project: _____

Directions: Circle the brick that shows how well you did. The bigger brick, the better you did.

I defined the question or problem.				
I built a LEGO® model and programmed a solution.				
I tested my solution and made improvements.				
I documented and shared my ideas.				

Project Reflection

One thing I did really well was: _____

One thing I want to improve on for next time is: _____



Teacher-led assessment

Developing students' science, engineering, and computational thinking skills requires time and feedback. Just as in the design cycle, in which students should understand that failure is part of the process, assessment should provide feedback in terms of what students did well and where they can improve. Problem-based learning is not about succeeding or failing. It is about being an active learner and continually building upon and testing ideas.

Giving feedback to students in order to help them develop their skills can be done in various ways. At each phase of the WeDo 2.0 projects, we have provided examples of rubrics that can be used by:

- Observing students' behaviour, reaction, and strategies
- Asking questions about their thought processes

As students often work in groups, you can give feedback both on a team level and on an individual level.

Anecdotal Record Grid

The anecdotal record grid lets you record any type of observation you believe is important for each student. Use the template on the next page to provide feedback to students as needed.





Anecdotal record grid

Name: _____

Class: _____

Project: _____

1. Emerging	2. Developing	3. Proficient	4. Accomplished
			

Notes:



Teacher-led assessment

Observation Rubrics

Examples of rubrics have been provided for every Guided Project. For every student, or every team, you can use the observation rubrics grid to:

- Evaluate student performance at each step of the process
- Provide constructive feedback to help the student progress

The observation rubrics provided in the Guided Projects can be adapted to fit your needs. The rubrics are based on these progressive stages:

1. Emerging

The student is at the beginning stages of development in terms of content knowledge, ability to understand and apply content, and/or demonstration of coherent thoughts about a given topic.

2. Developing

The student is able to present basic knowledge only (e.g., vocabulary), and cannot yet apply content knowledge or demonstrate comprehension of the concepts being presented.

3. Proficient

The student has concrete levels of comprehension of the content and concepts and can adequately demonstrate the topics, content, or concepts being taught. The ability to discuss and apply this knowledge outside the required assignment is lacking.

4. Accomplished

The student can take concepts and ideas to the next level, apply concepts to other situations, and synthesise apply, and extend knowledge to discussions that include extensions of ideas.

▶ Suggestion

Use the observation rubrics grid on the next page to keep track of your students' progress.





Observation rubrics grid

Class:		Project:			
Students' Names		Explore	Create	Test	Share
1					
2					
3					
4					
5					
6					
7					
8					
9					
10					
11					
12					
13					
14					
15					

To be used with the rubrics described on the following page: (1) emerging, (2) developing, (3) proficient, (4) accomplished.



Assessing project phases - general rubrics

You can use these assessment rubrics to give general feedback on a scale of 1 to 4 at the end of each phase of a project.

Explore Phase

In the Explore phase, feedback should relate to whether or not the student is actively involved in the discussion by asking and answering questions, and their level of understanding of the problem.

1. The student is unable to provide answers to questions or adequately participate in discussions.
2. The student is able, with prompting, to provide answers to questions or adequately participate in discussions.
3. The student is able to provide adequate answers to questions and participate in class discussions.
4. The student is able to extend explanations in class discussions.

Test Phase

During the Test phase, make sure that the student works well on a team, justifies his/her best solution, and uses the information collected in the Explore phase.

1. The student is unable to work well on a team, justify solutions, and use information collected for further development.
2. The student is able to work on a team, collect and use information with guidance, or, with help, to justify solutions.
3. The student is able to work on a team and contribute to the team discussions, justify solutions, and collect and use information about the content.
4. The student can justify and discuss solutions that allow for the collection and use of information.

Share Phase

During the Share phase, make sure that the student is able to describe their solution using the right vocabulary and the right level of detail.

1. The student does not use evidence from his/her findings in connection with ideas shared during the presentation and does not follow established guidelines.
2. The student uses some evidence from his/her findings, but the justification is limited. Established guidelines are generally followed but may be lacking in one or more areas.
3. The student provides adequate evidence to justify his/her findings and follows established guidelines for presenting.
4. The student fully discusses his/her findings and thoroughly utilises appropriate evidence to justify his/her reasoning while following all established guidelines.



Assessing Computational Thinking Skills

Name: _____

Class: _____

Decomposition	1. Emerging	2. Developing	3. Proficient	4. Accomplished	Notes
Describe the problem in your own words.	The student is unable to describe the problem in their own words. <input type="checkbox"/>	The student is able, with prompting, to describe the problem in their own words. <input type="checkbox"/>	The student is able to describe the problem in their own words. <input type="checkbox"/>	The student is able to describe the problem in their own words and starts to decompose the problem into smaller parts. <input type="checkbox"/>	
Describe how you will know whether or not you have found a successful solution to the problem.	The student is unable to describe success criteria. <input type="checkbox"/>	The student is able, with prompting, to describe success criteria. <input type="checkbox"/>	The student is able to describe success criteria. <input type="checkbox"/>	The student is able to describe success criteria with a high level of detail. <input type="checkbox"/>	
Describe how you can break the problem down into smaller parts.	The student is unable to break down the problem. <input type="checkbox"/>	With prompting, the student is able to break down the problem into smaller parts. <input type="checkbox"/>	The student is able to break down the problem into smaller parts. <input type="checkbox"/>	The student is able to break down the problem into smaller parts and can describe the links between each of the parts. <input type="checkbox"/>	



Assessing Computational Thinking Skills

Name: _____

Class: _____

Generalisation	1. Emerging	2. Developing	3. Proficient	4. Accomplished	Notes
Describe which program you have used from the Program Library (or elsewhere) and why.	The student is unable to describe which program has been used and why. <input type="checkbox"/>	The student is able to identify which program has been used. <input type="checkbox"/>	The student is able to describe which program has been used and why. <input type="checkbox"/>	The student is able to describe, in detail, which program has been used and what modifications have been made to it. <input type="checkbox"/>	
Observe how your students recognise patterns, or reuse concepts they have seen before.	The student is unable to recognise patterns, or reuse concepts seen before. <input type="checkbox"/>	With prompting, the student is able to recognise patterns, or reuse concepts seen before. <input type="checkbox"/>	The student is able to recognise patterns, or reuse concepts seen before. <input type="checkbox"/>	The student is able to recognise patterns, or reuse concepts of their own. <input type="checkbox"/>	



Assessing Computational Thinking Skills

Name: _____

Class: _____

Algorithmic Thinking	1. Emerging	2. Developing	3. Proficient	4. Accomplished	Notes
Describe the list of actions to program.	The student is unable to make a list of actions. <input type="checkbox"/>	With prompting, the student is able to make a list of actions. <input type="checkbox"/>	The student is able to make a list of actions. <input type="checkbox"/>	The student is able to make a detailed list of actions to help them develop their program. <input type="checkbox"/>	
Describe how you have programmed your solution.	The student is unable to describe the program. <input type="checkbox"/>	With prompting, the student is able to describe the program. <input type="checkbox"/>	The student is able to describe the program. <input type="checkbox"/>	The student is able to describe the program, providing extensive details about each component. <input type="checkbox"/>	
Describe the programming principles used in your solution (e.g., output, inputs, events, loops, etc.).	The student is unable to describe the programming principles used in their solution. <input type="checkbox"/>	With prompting, the student is able to describe the programming principles used in their solution. <input type="checkbox"/>	The student is able to describe the programming principles used in their solution. <input type="checkbox"/>	The student is able to describe, with extensive comprehension, the programming principles used in their solution. <input type="checkbox"/>	



Assessing Computational Thinking Skills

Name: _____

Class: _____

Evaluation	1. Emerging	2. Developing	3. Proficient	4. Accomplished	Notes
					
Describe what happened when you executed your program, and whether or not it was what you expected.	The student cannot describe what happened. <input type="checkbox"/>	With prompting, the student is able to describe what happened, and compare it to what was expected. <input type="checkbox"/>	The student is able to describe what happened, and compare it to what was expected. <input type="checkbox"/>	The student is able to describe what happened, compare it to what was expected, and is already finding solutions. <input type="checkbox"/>	
Describe how you have fixed the problems in your program.	The student cannot describe how they have fixed the problems. <input type="checkbox"/>	With prompting, the student can describe how they have fixed the problems. <input type="checkbox"/>	The student can describe how they have fixed the problems. <input type="checkbox"/>	The student can describe, in extensive detail, how they have fixed the problems. <input type="checkbox"/>	
Describe how your solution is linked to the problem.	The student is unable to describe how their solution is linked to the problem. <input type="checkbox"/>	With prompting, the student is able to describe how their solution is linked to the problem. <input type="checkbox"/>	The student is able to describe how their solution is linked to the problem. <input type="checkbox"/>	The student is able to describe, in extensive detail, how their solution is linked to the problem. <input type="checkbox"/>	
Describe how you have tried new ways of solving the problems throughout the project.	The student is unable to describe other ways tried throughout the project. <input type="checkbox"/>	The student is able, with prompting, to describe other ways tried throughout the project. <input type="checkbox"/>	The student is able to describe other ways tried throughout the project. <input type="checkbox"/>	The student is able to describe other ways tried throughout the project, and is able to describe why each option has not been considered. <input type="checkbox"/>	



Assessing Computational Thinking Skills

Name: _____

Class: _____

Abstraction	1. Emerging	2. Developing	3. Proficient	4. Accomplished	Notes
Describe the most important part of your solution.	The student is not able to describe their solution. <input type="checkbox"/>	With prompting, the student is able to describe their solution. <input type="checkbox"/>	The student is able to describe their solution. <input type="checkbox"/>	The student is able to describe their solution, focusing on the most important part of the solution. <input type="checkbox"/>	
Describe the most important details of your solution.	The student is not able to provide any details about their solution. <input type="checkbox"/>	With prompting, the student is able to provide details about their solution. <input type="checkbox"/>	The student is able to discuss details of their solution, but some of the details are not essential. <input type="checkbox"/>	The student is able to discuss the most important details of their solution. <input type="checkbox"/>	
Describe how your solution met the initial criteria.	The student is unable to describe how their solution met the initial criteria. <input type="checkbox"/>	With prompting, the student is able to describe how their solution met the initial criteria. <input type="checkbox"/>	The student is able to describe how their solution met the initial criteria. <input type="checkbox"/>	The student is able to describe, with extraordinary clarity, how their solution met the initial criteria. <input type="checkbox"/>	

LEGO® Education WeDo 2.0



LEGOeducation.com

LEGO and the LEGO logo are trademarks of the/son des marques de commerce du/son marcas registradas de LEGO Group. ©2017 The LEGO Group. 2017.01.01. - VI.

