

# THE VALUE OF CODING AND ROBOTICS IN THE FOUNDATION PHASE

**PRESENTED BY: CORNETTE ESTERHUIZEN**

DATE: 16 MARCH 2023

WEBINAR



**SAOU**

DIE VERANDERING IN ONDERWYS  
THE CHANGE IN EDUCATION



# Coding and robotics in South Africa – what schools will actually be teaching



basic education

Department:  
Basic Education  
REPUBLIC OF SOUTH AFRICA

PROPOSED AMENDMENTS TO THE CURRICULUM AND  
ASSESSMENT POLICY STATEMENT (CAPS) TO MAKE PROVISION  
FOR CODING AND ROBOTICS GRADES R- 9

Basic Education minister Angie Motshekga has called for comments to amend the Curriculum and Assessment Policy Statement (CAPS) to make provision for coding and robotics at South African schools.

Curriculum and Assessment Policy Statement  
Grades R-3  
**CODING AND ROBOTICS**

In a gazette and notice published on the department's website, Motshekga said that the subjects would form part of the curriculum at different school levels from Grade R to Grade 9.

The coding and robotics subjects are aimed at guiding and preparing learners to solve problems, think critically, work collaboratively and creatively, and function in a digital and information-driven world, the department said.

It added that learners will be able to apply digital and ICT skills and to transfer these skills to solve everyday problems and its possibilities.



### 1.4.1 Foundation Phase

(a) The instructional time in the Foundation Phase is as follows:

SUBJECT	GRADE R (HOURS)	GRADE 1-2 (HOURS)	GRADE 3 (HOURS)
Home Language	10	8/7	8/7
First Additional Language		2/3	3/4
Mathematics	7	7	7
Coding and Robotics	1	1	2
Life Skills:	6	6	7
• Beginning Knowledge	(1)	(1)	(2)
• Creative Arts	(2)	2	(2)
• Physical Education	(2)	2	(2)
• Personal and Social Well-being	(1)	(1)	(1)
<b>TOTAL</b>	<b>(24)</b>	<b>(24)</b>	<b>(27)</b>

## INTRODUCTION TO CODING AND ROBOTICS

### What is Coding and Robotics?

The Coding and Robotics subject is central to function in a digital and information-driven world; apply digital ICT skills and transfer these skills to solve everyday problems in the development of learners. It is concerned with the various inter-related areas of Information Technology and Engineering. The subject studies the activities that deal with the solution of problems through logical and computational thinking. Draft Policy

## Instructional time in the Foundation Phase

Instructional time for Grades R, 1 and 2 is 24 hours and for Grade 3 is 27 hours.

- Ten hours are allocated for languages in Grades R-2 and 11 hours in Grade 3. A maximum of 8 hours and a minimum of 7 hours are allocated for Home Language and a minimum of 2 hours and a maximum of 3 hours for Additional Language in Grades 1-2. In Grade 3 a maximum of 8 hours and a minimum of 7 hours are allocated for Home Language and a minimum of 3 hours and a maximum of 4 hours for First Additional Language.
- In Life Skills Beginning Knowledge is allocated 1 hour in Grades R – 2 and 2 hours as indicated by the hours in brackets for Grade 3



**In the Curriculum and Assessment Policy Statement (CAPS) the subject Coding and Robotics in Foundation Phase (Grades R-3) has been organised into five study areas:**

**Pattern Recognition,**

**Algorithms and Coding,**

**Robotics Skills,**

**Internet and e-communicating**

**Application Skills has been organised in this way in order to ensure that the foundational skills, values and concepts of early childhood development and of the subjects offered in Grades 4 - 9 are taught and developed in Grades R-3.**

## **Introduction – Draft policy**

**Beginning Knowledge and Personal and Social relationships are integrated in the topics. Coding and Robotics is a subject that transverses across the other core Foundation Phase subjects namely Languages (home and First Additional) and Mathematics that ultimately strengthens and supports them.**





## Specific Aims:

The Coding and Robotics subject is aimed at guiding and preparing learners to solve problems, think critically, work collaboratively and creatively, function in a digital and information-driven world, apply digital and ICT skills and to transfer these skills to solve everyday problems and its possibilities, including equipping learners for meaningful and successful living in a rapidly changing and transforming society.

Through Coding and Robotics learners are exposed to a range of knowledge, skills and values that strengthen their:

- ☐ aesthetic, creative skills and cognitive development, knowledge through engaging in dance, music, drama and visual art activities
- ☐ knowledge of digital and ICT skills supported by the technological process and computational thinking skills;

Understanding of the relationship between people and the environment, awareness of social relationships, and elementary science;

- ☐ physical, social, personal and emotional development.

## 2.3 Focus Content Areas:

The Coding and Robotics Foundation Phase subject consist of the following Knowledge Strands:

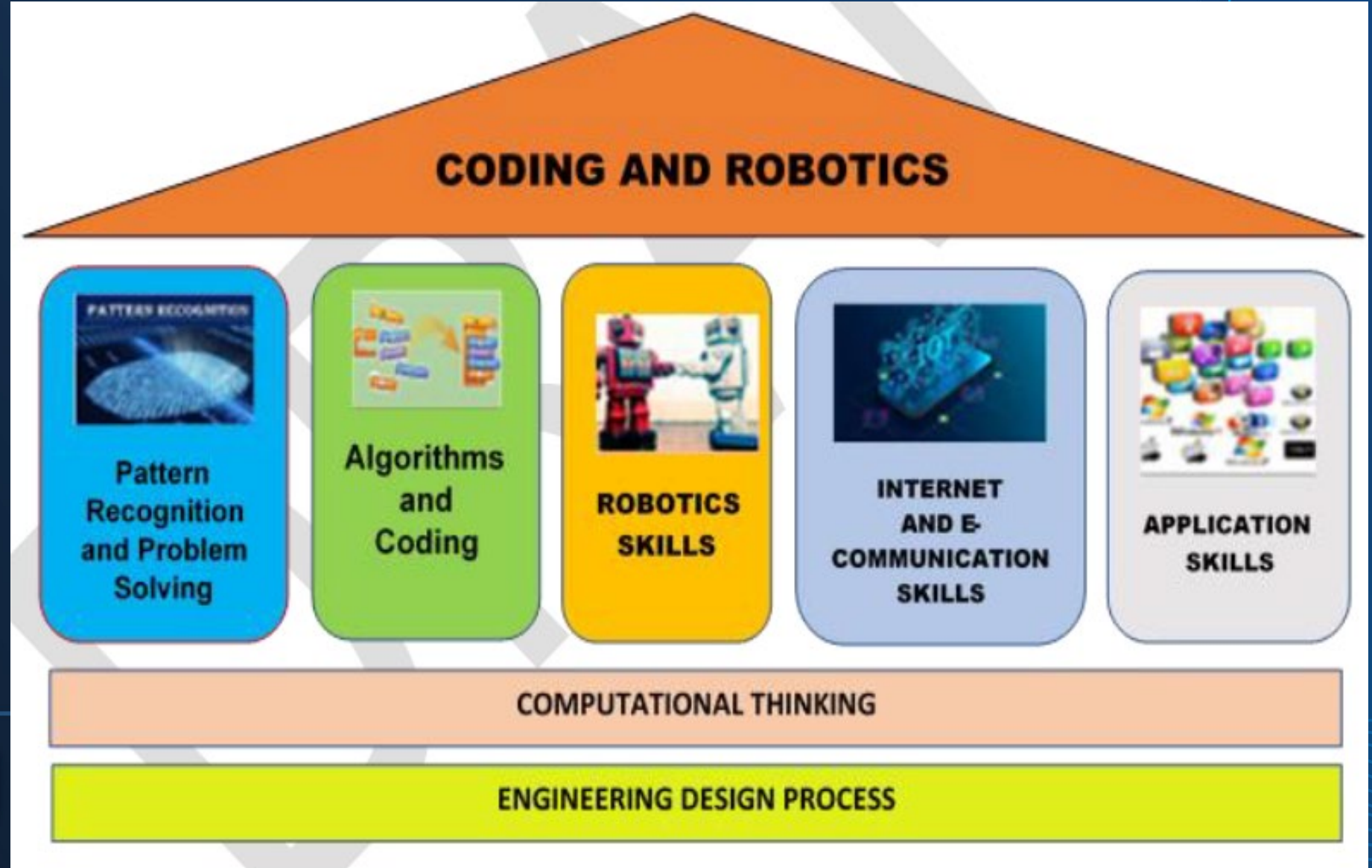
- ☐ Pattern Recognition and Problem Solving
- ☐ Algorithms and Coding
- ☐ Robotic Skills
- ☐ Internet and E-Communication skills
- ☐ Application Skills





Grade R – Grade 3

In the Foundation phase (Grades R-3) the subject has been organised into five study areas:





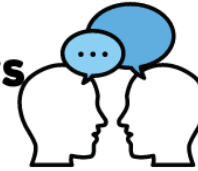
# Skills that childrens develop

Thanks to Robotics Education



**Creativity and Imagination**

**Developing new ways  
of communication**



**Team Work**

**Self-esteem**



**Adapting to the  
future**

**Proactive spirit**





# Foundation Phase Initiative

## Learning through play with Six Bricks



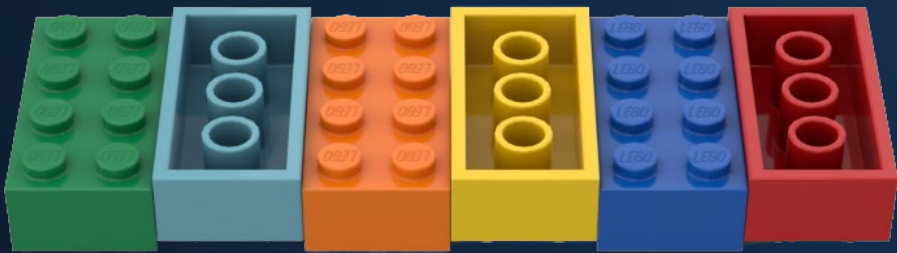
The **LEGO** Foundation



SAOU



Introduction to Six Bricks and the FPI Programme  
Introduction to Unplugged Coding  
Introduction to Computational Thinking  
Implementing of Computational Thinking Activities  
Using Six Bricks Mats  
Games  
Implementing the Programme in the Classroom





## What are the benefits of engaging in Six Bricks Activities?

When children start formal school education, they still require **concrete manipulatives** to help them to grasp difficult concepts, and they still need to use their **whole body** to learn.

Six Bricks on each child's desk offers that assistance and makes for a whole lot of learning through play!

When running activities with Six Bricks, it becomes very easy to see that the following skills can be developed in the children:

Working memory

Inhibitory control

Cognitive flexibility

Developmental skills –

Gross and fine motor skills

Children are not  
born with executive  
function skills.

**BUT:** They are born  
with the potential to  
develop them.

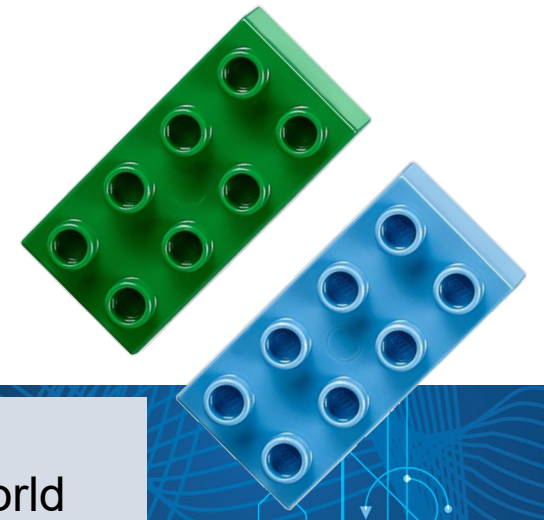
And continues into  
early childhood and is  
shaped by their  
experience.

It is a slow process  
that begins with  
infancy.



# What is coding and robotics?

- We love working with computers because they seem to be able to do anything, *BUT computers cannot think for themselves*
- That is why *we need coders*: **Coders write code (instructions – the language) for a computer – telling the computer how to solve problems**
- We call this code *a 'program'*
- **Learning how to code is one of the core skills** or competences for children to develop problem solving skills – and not just for computing!
- Robotics is a 3D tool that we use to help students understand IT concepts
- **Robotics is made of 3 parts**: a machine, programming, and sensors





# Coding vs Programming

# What is the difference?



- Coding is the act of *writing the instructions* that you send to a computer to do something for you
- Programming is the *process of creating an outline* and structure for the program's code that follows certain *standards*, before the actual code is written to perform the task it needs to perform
- Coding forms a part of programming
- FOR A CODER NO PROBLEM IS TOO BIG
- Coders love mysteries –they break big problems into smaller ones and solve each problem on a time

By writing a set of instructions to get from the door to your desk, you are using code (go left, go right, sit down, etc.); but in understanding, planning and developing the code for you to get from the door to your desk, you are programming (should it be written in English, is the person able to walk, how many obstacles do we have to avoid, etc.).



© CanStockPhoto.com - csp34972853



# Why unplugged coding ?

**Unplugged ... coding doesn't always have to be attached to a computer**

Simply put, coding is used for communicating with computers. People use coding to give computers and other machines instructions on what actions to perform. Can we teach children the basics of creating instructions or code without the use of technology? Yes!



- Teach programming concepts using games or activities that can be done offline using physical objects, such as blocks, paper and markers
- Help learners understand abstract concepts through physical objects that can be touched, manipulated, and described - make learning tangible
- Activities can include different sensory approaches - physical movement, using music, manipulating objects, drawing pictures
- Unplugged lessons are particularly useful for building and maintaining a collaborative classroom environment
- The biggest part of Unplugged Coding is teaching students to be confident problem solvers!



# Where does it fit?

## CAPS Curriculum Teaching Plan



At this point, the CAPS curriculum is not finalized but based on the draft curriculum, Unplugged Coding with Six Bricks can provide for the following:

- The new CAPS Coding and Robotics Curriculum is currently designed to target 5 specific areas:
  1. **Pattern Recognition and Problem Solving**  
Many unplugged activities to teach these topics
  2. **Algorithms and Coding**  
Many unplugged activities to teach these topics
  3. **Robotics**  
Students use creative thinking and imagination to apply coding concepts to robotics  
Build machines that they code physically without computers
  4. **Internet and E-Communications**  
Teaching the concepts so that students have a basic understanding
  5. **Applications**  
Limited scope for unplugged coding



**CPA**

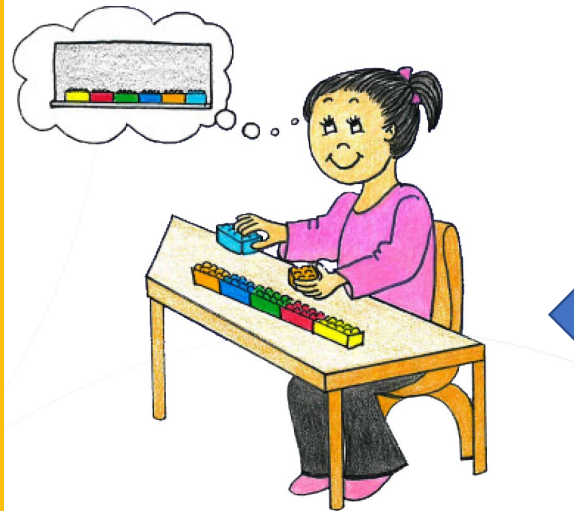
**Apply the  
concepts**



**Use your body as a  
tool to introduce  
the concept**

**We will follow a simple  
process to teach:**

- 1. Use your body to  
learn the concepts**
- 2. Use Six Bricks (or  
other resources) to  
understand the  
concepts**
- 3. Students work on  
their own or in pairs  
/ groups to apply the  
concepts**



**Investigate the concept using  
concrete manipulatives**



# Computational Thinking

## What is computational thinking?

**The process of breaking down a problem into simple enough steps that even a computer would understand**

**Computational Thinking (CT) is a problem-solving process that uses logical and analytical skills to follow directions or instructions – just like programming.**

**Fundamental skill useful for everyone**

**Let's think like a computer using a problem-solving process**

### BACK TO BACK ACTIVITY

Teaching points: Students will need to express abstract concepts to each other. Tell the computer how to do what they want them to do. Discuss concepts in such a way that the recipient understands is an important part of programming.



21st century skills are based on reasoning and logical thinking

The ability of coding is called “algorithmic thinking” and “computational thinking”

In performing the coding process, it is important to follow the steps of comprehending, analyzing, solving the problems, and making the results as algorithms, establishing the correct algorithm, and encoding the algorithm with a program over the language.





# What is Computational Thinking?

Computational thinking allows us to take a complex problem, understand what the problem is and develop possible solutions.

We can then present these solutions in a way that a computer, a human, or both, can understand.

There are four key parts to computational thinking :

## Decomposition



Breaking something into smaller parts.

## Pattern Recognition



Looking for similarities and trends.

## Abstraction



Focusing on what's important, ignoring what is unnecessary.

## Algorithm Design

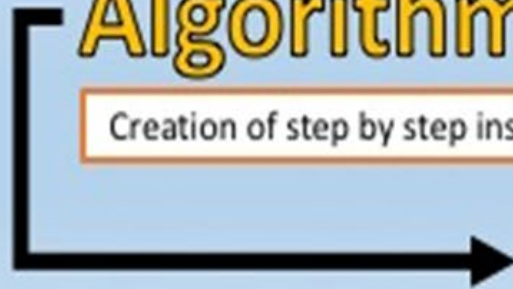


Creation of step by step instructions to solve a problem.

## Debugging



Fixing errors within your algorithm.





# The Computational Thinkers

## concepts



### Logic

Predicting & analysing



### Evaluation

Making judgements



### Algorithms

Making steps & rules



### Patterns

Spotting & using similarities



### Decomposition

Breaking down into parts



### Abstraction

Removing unnecessary detail



## approaches



### Tinkering

Changing things to see what happens



### Creating

Designing & making



### Debugging

Finding & fixing errors



### Persevering

Keeping going



### Collaborating

Working together



# What is computational thinking?

**Problem-solving skills can be improved by computational thinking processes. Through computational thinking, we can explain the problem and use simple methods or formulas to solve the problem by computer computation**

**Computational thinking allows us to take a complex problem, understand what the problem is and develop possible solutions.**

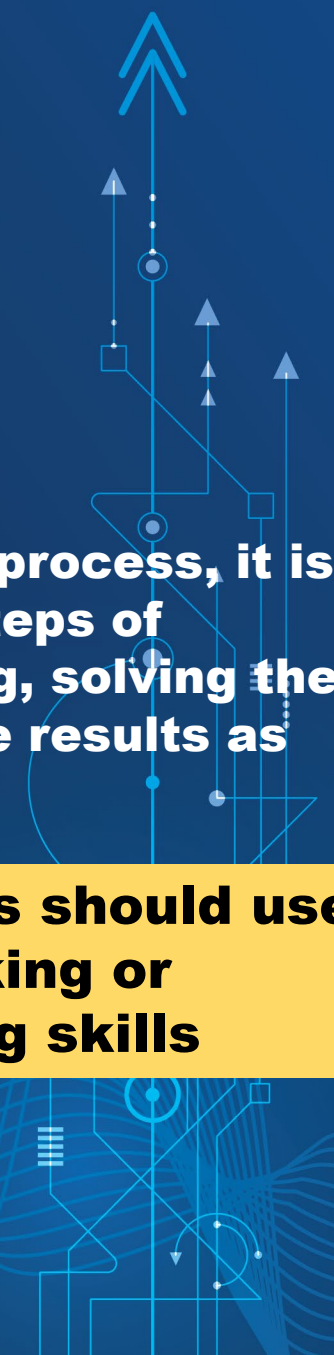
**We can then present these solutions in a way**

**This broad problem-solving technique includes four elements: decomposition, pattern recognition, abstraction and algorithms**

**Students will need to express abstract concepts to each other.**

**In performing the coding process, it is important to follow the steps of comprehending, analyzing, solving the problems, and making the results as algorithms**

**In this process, coders should use their algorithmic thinking or computational thinking skills**

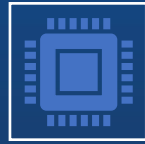




## The four cornerstones:

Decomposition  
Pattern recognition  
Abstraction  
Algorithms

&  
Evaluation and  
debugging



**Breaking down a problem** - breaking down a complex problem into smaller, more manageable parts (decomposition)



**What is important?** – focusing on the important information only, ignoring irrelevant detail (abstraction)



**What is the same?**– looking for similarities among and within problems (pattern recognition)



**Can we code?** - developing a step-by-step solution to the problem, or the rules to follow to solve the problem , algorithmic thinking.



**Did we get it right?** – evaluate process, find errors then problem solve solutions (evaluation & debugging)

**Its not even thinking like a computer, as computers do not, cannot think. CT enables you to work out exactly what to tell the computer what to do.**

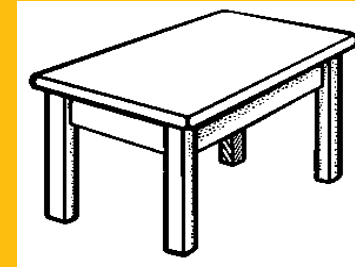
SAOU



**Young children can learn the basic concepts of coding.** These concepts are important stepping stones not only for learning to code, but for developing skills like critical thinking and problem solving.

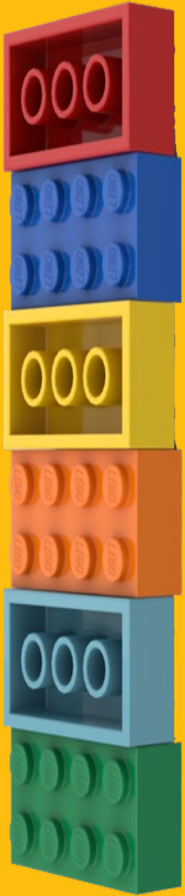
There are 5 components of CT. Each component is as important as the others. They are like legs on a table - if one leg is missing, the table will probably collapse.

**Algorithmic Thinking – developing a step by step solution to the problem or the rules to follow to**



**Computational thinking** involves taking a complex problem and breaking it down into a series of small, more manageable problems (**decomposition**). Each of these smaller problems can then be looked at individually, considering how similar problems have been solved previously (**pattern recognition**) and focusing only on the important details, while ignoring irrelevant information (**abstraction**). Next, simple steps or rules to solve each of the smaller problems can be designed and a step by step solution is created (**algorithms**). Testing solutions, especially if they are not working, is an ongoing process – requiring perseverance and problem solving know how (**evaluation & debugging**)





# COMPUTATIONAL THINKING

## DECOMPOSITION

BREAK DOWN DATA AND PROBLEMS INTO SMALLER PARTS

## PATTERN RECOGNITION

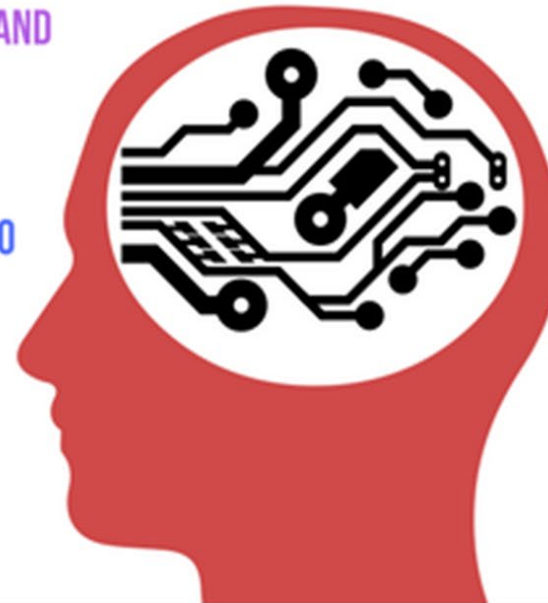
OBSERVE PATTERNS AND TRENDS IN DATA

## ALGORITHMS

DETERMINE WHAT STEPS ARE NEEDED TO SOLVE A PROBLEM

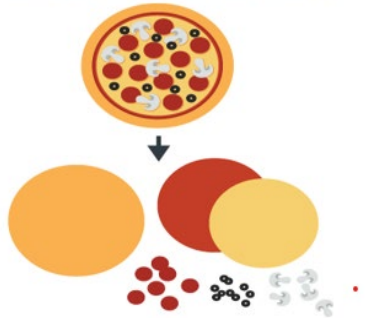
## ABSTRACTION

REMOVE DETAILS AND EXTRACT RELEVANT INFORMATION





## Decomposition

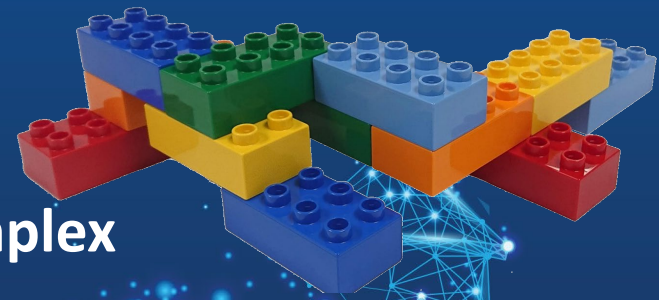


What are the different parts that create this?

# Decomposition

Decomposition is all about breaking down a complex problem into smaller, more manageable parts.

Problem solving requires breaking down before building up again.



2D to 3D

Think about where you will start building this model – top/bottom/left/right?

How many bricks do you see? Which colour brick has more than 1 brick in the model?

How many full bricks (all 8 studs) can you see?

The idea of **decomposition** is to work backward from end to beginning and to **break large problems into smaller ones** that can more easily be solved. We all encounter problems on a day-to-day basis. Some are small and easy to solve, and there are some which are larger, more complex, and difficult. **Through coding, children learn to think and learn about different situations that are not the norm.** They learn to analyse options and have to come up with a way to solve any challenges they come across. These problem-solving skills are a great benefit in their day to day lives and can help them to solve real-life situations.

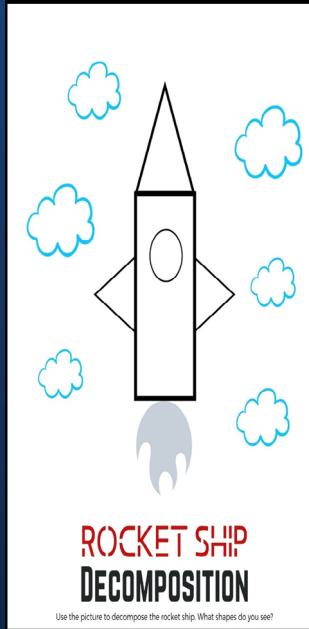
Computer programmers can't just look at a large problem and fix it right away

# SAOU



# Decomposition

They have to break it down into the smallest possible pieces, solve each one of those little problems, then build those smaller solutions together to solve the larger problem they started with.



**The idea of decomposition is to work backward from end to beginning and to break large problems into smaller ones that can more easily be solved.**

**Through coding, children learn to think and learn about different situations that are not the norm.**

**These problem-solving skills are a great benefit in their day to day lives and can help them to solve real-life situations.**

**Computer programmers can't just look at a large problem and fix it right away.**

***For example think about making an omelette for your breakfast. Before you can get to the finished product, you have to get out a frying pan and put it on the hob, crack your eggs into a cup, whisk them, pour them into the pan, fry them, add seasoning, etc.***



**SAO U**



# Abstraction

Abstraction is the removal of unnecessary information so a computer program can be as efficient as possible.



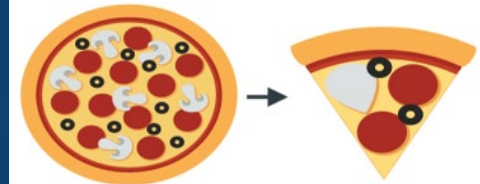
computing-made-simple.co.uk

## Computational Thinking

### What is important? –

Focusing on the important information only, ignoring irrelevant detail (abstraction)

### Abstraction



How do you know what this is?



# Abstraction

- **Abstraction is all about focusing only on the important details, while ignoring irrelevant information.**
- **It simplifies problems and prevents unnecessary repetition.**
- **Pulling out specific differences to make one solution work for multiple problems.**
- **Use only the simplest representations of the problem that are necessary in order to solve it.**

Playing Pictionary – creating simple, quick sketches for a partner to guess what they are representing. In doing this, they learn that they are ignoring unimportant details and only including that which is most important – this is abstracting.

PICTIONARY



SAOUI







# WHAT IS COMPUTATIONAL THINKING?



Computational thinking = solving problems logically



## Abstraction

Focusing on what's important, ignoring what is unnecessary.

To help you remember, think of...



Example:

You have been asked to draw a cat. Your image must be representative of all types of cats.

To draw a basic cat, we **do** need to know that it has a tail, fur and eyes. These characteristics are **relevant**.

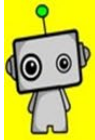
We **don't** need to know what sound a cat makes or that it likes fish. These characteristics are **irrelevant** and can be **filtered out**.



For more ideas, follow  
@RobbotResources

SAOU





## Application of Computational Thinking Across the Curriculum!

# Pattern Recognition

Looking for similarities or trends

Primary Terminology – “Patterns”

Early Years Foundation Terminology – “Matching”

“That word sounds like...”

Critically review an existing piece of work.

Give feedback comparing work to specific criteria

Correct application of Male/Female tenses

Sudoku

Solitaire

Chess tactics

Logic puzzles

Does the star wars and superman theme tunes sound the same?

Preferred playing positions in a sport

What do platform computer games have in common?

Pattern and sequence matching

Spot the difference

Identify gradients /contours in an OS map to measure steepest route

Code breaking

Days of the week/Month

Times tables

What drawing technique would be best to use for that style of image?

“What tactics worked well the last time we played them?”

Identify similar riffs within a song

Word search

Sorting and classifying activities

Fit shapes into correctly shaped holes

Maths patterns, e.g. Fibonacci series (1,1,2,3,5,8,13,21...)

## Pattern recognition

SAOU



# Pattern Recognition

What is the same?–

Looking for similarities among and within problems (pattern recognition)

By analysing a sequence, we can learn to spot patterns.

This helps us reduce the amount of work we need to do as we can reuse code for the same problem or section. This makes the problem much easier to solve!

Pattern recognition is looking for similarities and differences between and within problems

Once patterns are ascertained, it is easier to define and solve the problem

The patterns can be recreated using algorithms

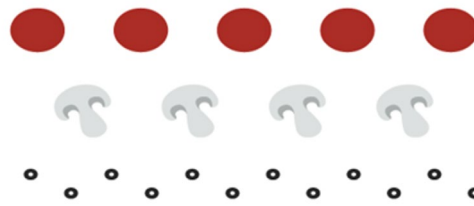
The repetition of the patterns is looping, and can be further extended to concept such as reiteration and reuse

Loop (Repeat the pattern)

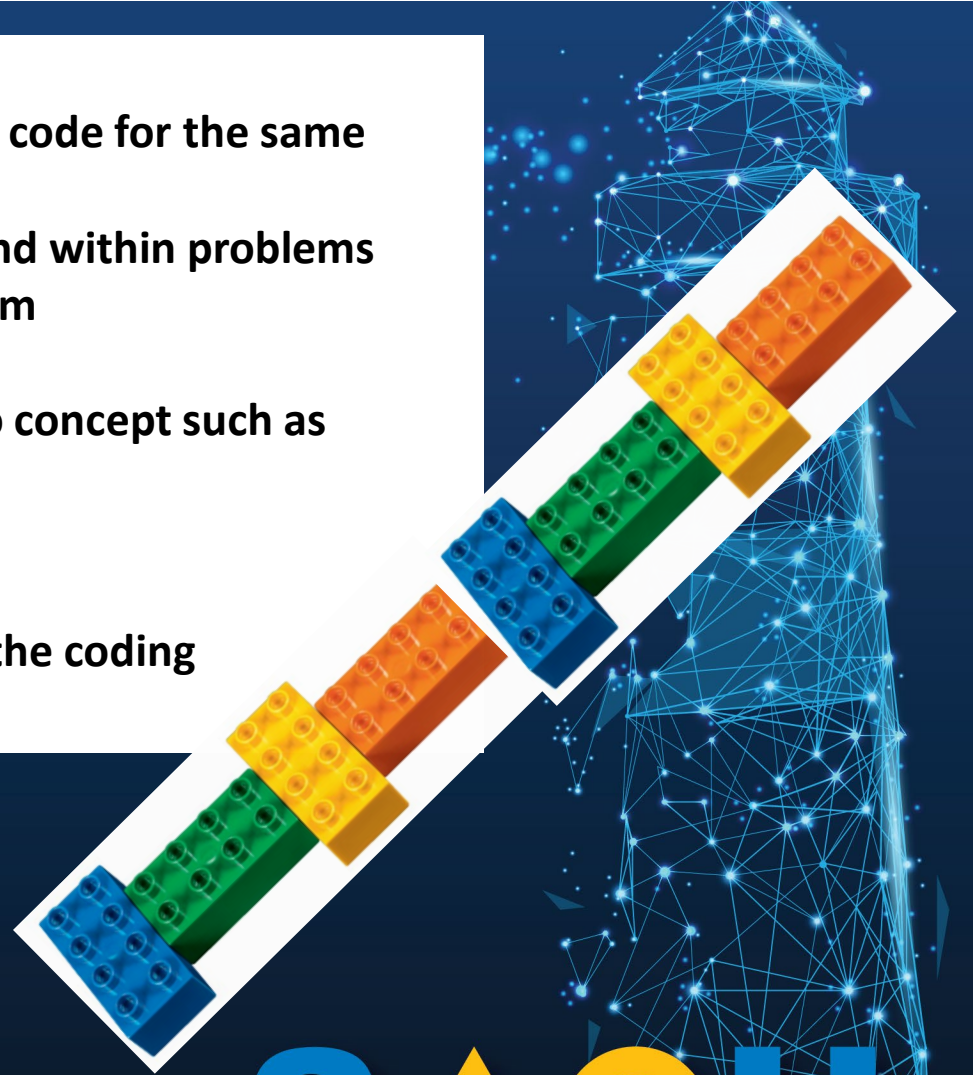
You could use the coding tokens now if you feel confident. Will do with the coding mat later on)

Young children can learn the basic concepts of coding. These concepts are important stepping stones not only for learning to code, but for developing skills like critical thinking and problem solving.

## Pattern Recognition



Do you notice anything that repeats?



# SAOUI

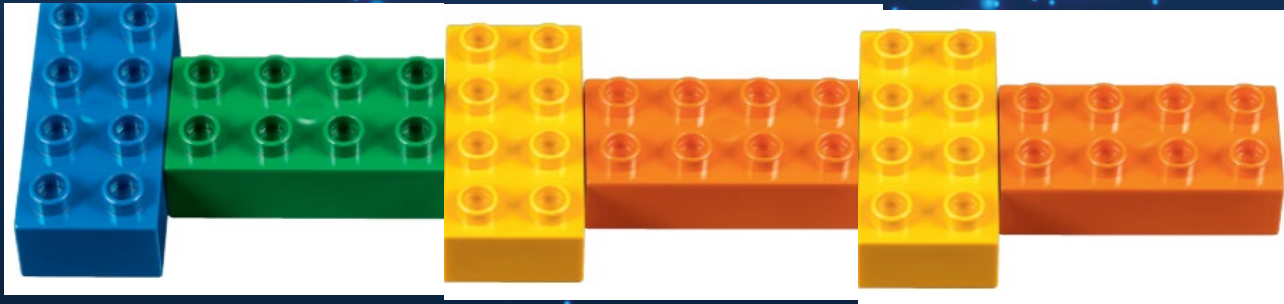


# Sequencing

**A sequence is a particular order of instructions in order to complete a specific task.**

**As a coder, it is important to be able to look for similarities among and within problems.**

**Spotting the similarities makes it easier of us to solve the problem!**



**Extend the code (pattern)**

**Build a colour sequence— groups of 4**

**Build a 3D upright sequence — groups of 4  
Tower**

**Create your own sequence of code**

What is the same?  
What is different?

Dice  
Loop  
Bug

SAO U



# SAOOU

By analysing a sequence, we can learn to spot patterns. This helps us reduce the amount of work we need to do as we can reuse code for the same problem or section. This makes the problem much easier to solve!

Pattern recognition is looking for similarities and differences between and within problems

Once patterns are ascertained, it is easier to define and solve the problem

The patterns can be recreated using algorithms

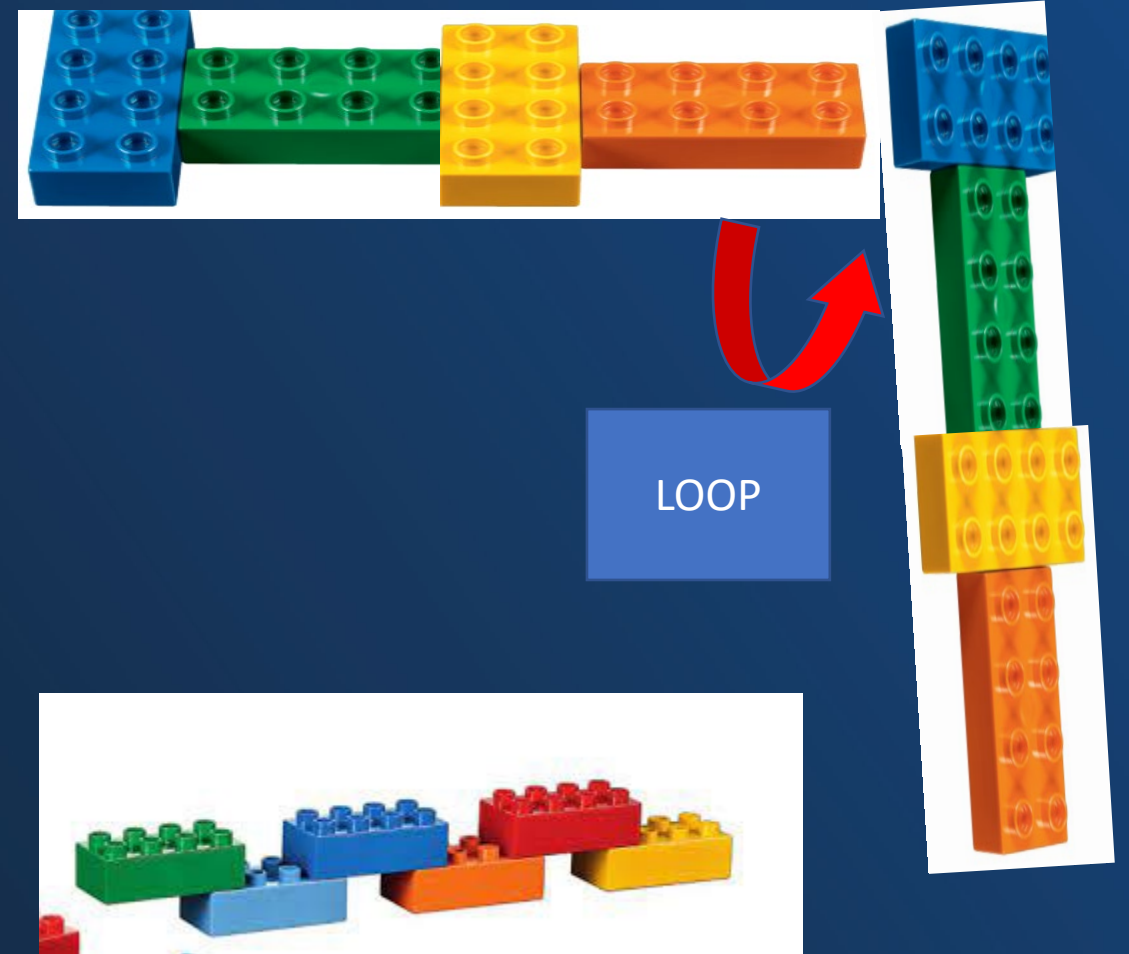
The repetition of the patterns is looping, and can be further extended to concept such as reiteration and reuse

**Loop (Repeat the pattern)**

You could use the coding tokens now if you feel confident. Will do with the coding mat later on)

When we **write code**, we have to get the **sequence of steps right**. A sequence is the order of tasks. When designing an algorithm, the order of tasks matters.

Computers aren't smart and need to be given instructions in a specific order or they won't be able to properly execute the command.



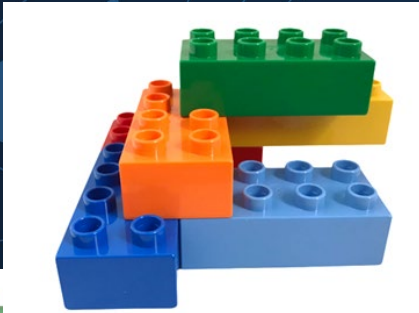


# Pattern Recognition



**Computers aren't smart and need to be given instructions in a specific order or they won't be able to properly execute the command.**

**By analysing a sequence, we can learn to spot patterns. This helps us reduce the amount of work we need to do as we can reuse code for the same problem or section. This makes the problem much easier to solve!**



**Pattern recognition is looking for similarities and differences between and within problems  
Once patterns are ascertained, it is easier to define and solve the problem**

**The patterns can be recreated using algorithms**

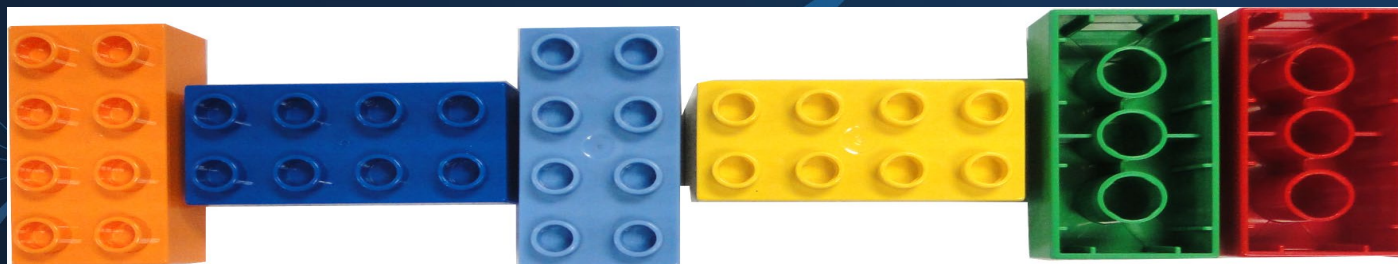
**The repetition of the patterns is looping, and can be further extended to concept such as reiteration and reuse**

**Loop (Repeat the pattern)**

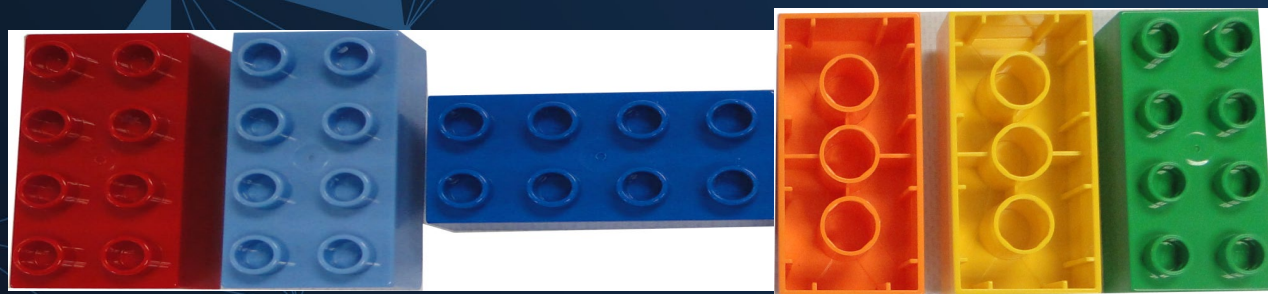


Patterns are not just found in math, but in art, nature and music, too. Being able to identify, recognise & build upon sequences will help children in other subjects, like science & geography.

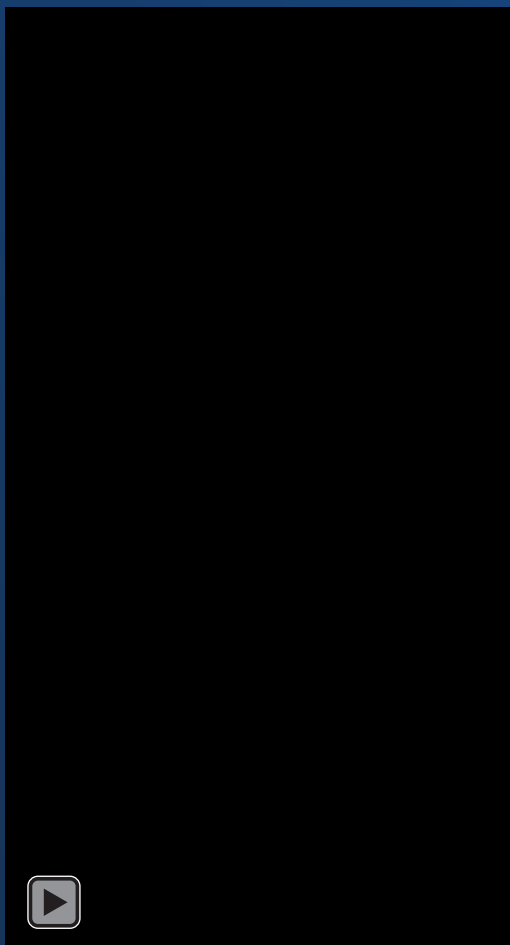
# SΔOU



CLAP CLICK CLAP CLICK CROSS CROSS



CLAP CLAP CLICK CROSS CROSS CLAP







# WHAT IS COMPUTATIONAL THINKING?



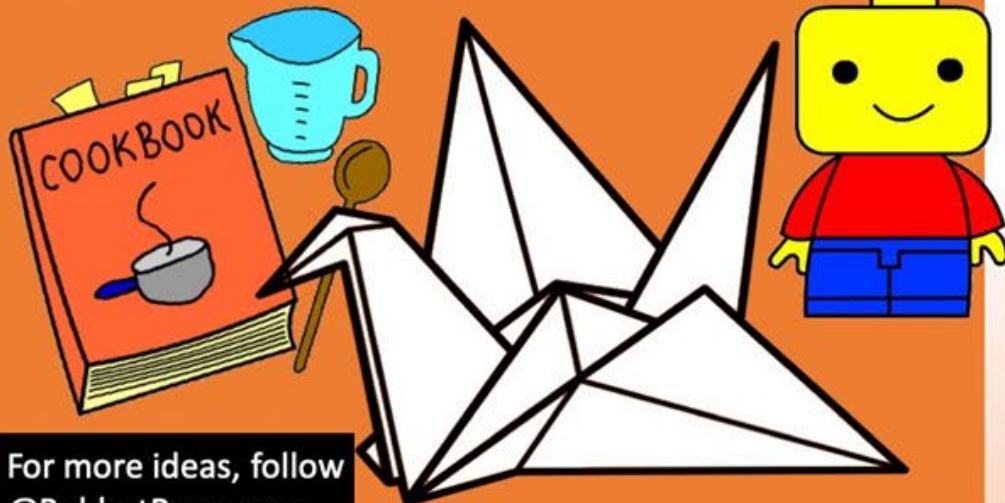
**Computational thinking = solving problems logically**



## Algorithm Design

The creation of a set of step-by-step instructions to solve a problem.

To help you remember, think of...



**Example:**

Someone stops you and asks for directions to the nearest shop. You need to give them clear, step-by-step instructions on the quickest route.



For more ideas, follow  
@RobbotResources



## Examples of Algorithmic thinking

**To give the exact instruction  
If they mis one step the  
outcomes would be wrong.  
That is coding!!**

### INTEGRADE WITH OTHER SUBJECTS

**Languages:** Students apply new vocabulary and practice speaking skills to direct another student to perform a task, whether it's ordering coffee at a café or navigating from one point in a classroom to another.

- Arts:** Students create instructions for drawing a picture that another student then has to use to recreate the image.

- English Language Arts:** Students map a flow chart detailing steps for determining whether to use a colon or dash in a sentence.

- Mathematics:** In a word problem, students develop a step-by-step process for how they answered a question that can then be applied to similar problems.

Common examples include: the recipe for baking a cake, the method we use to solve a long division problem, the process of doing laundry, and the functionality of a search engine are all examples of an algorithm.

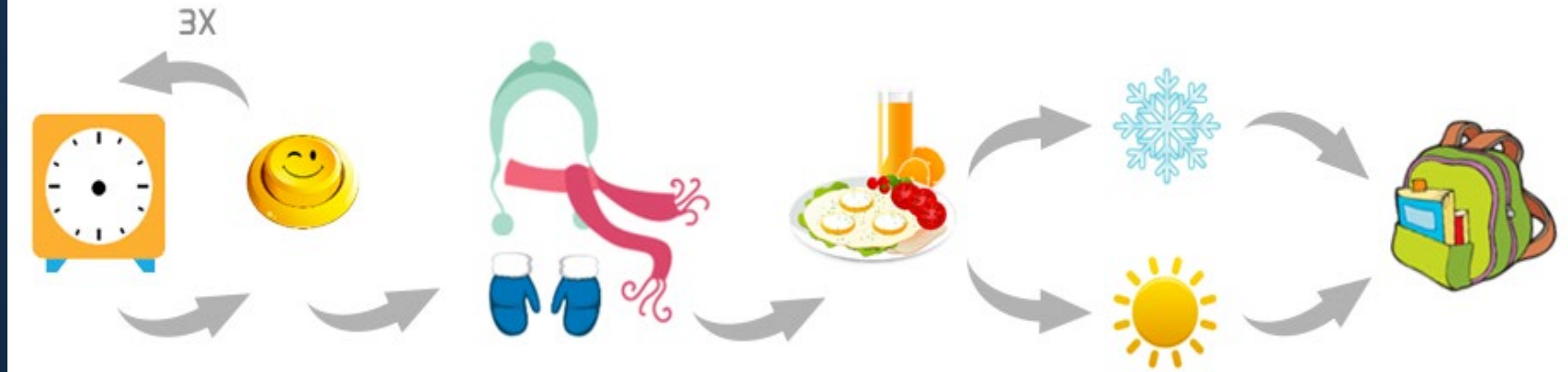


### Tying Your Shoes

Any step-by-step process that is completed the same way every time is an algorithm. A good example of this in everyday life is tying your shoes. There are a limited number of steps that effectively result in a traditional shoelace knot (known as the "bunny rabbit" or "loop, swoop and pull" knot). Chances are that you and your students follow one of these algorithms every time you tie your shoes.

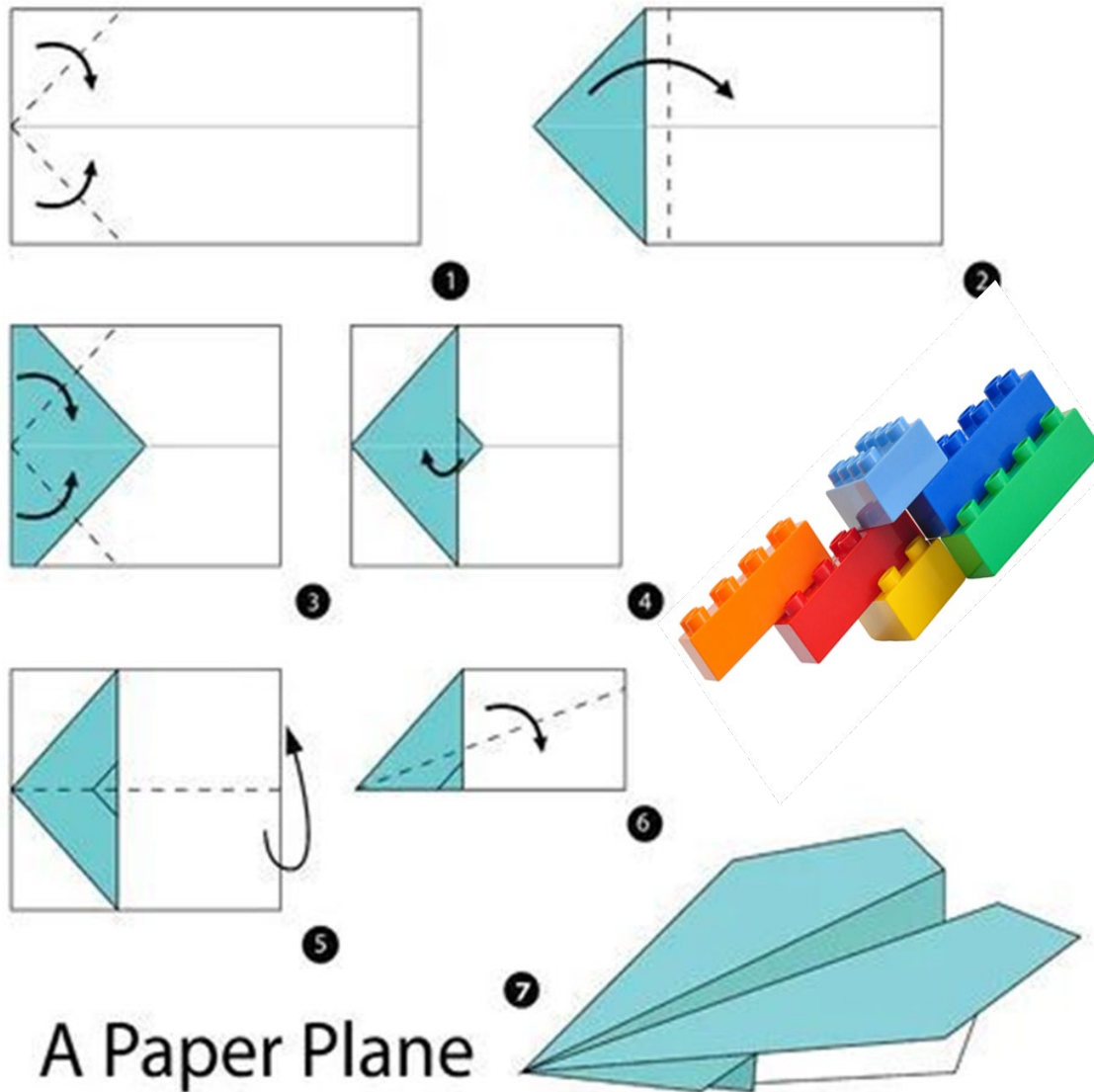


## Examples of Algorithmic thinking



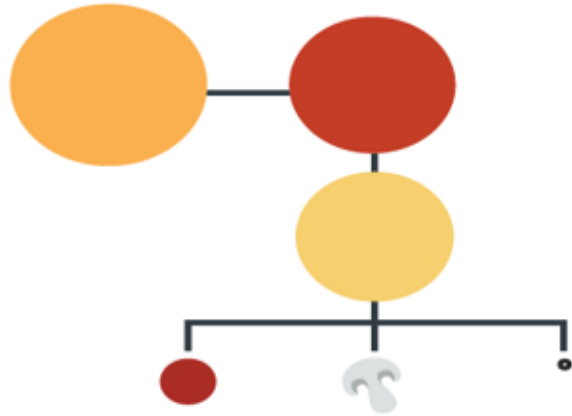


If you are [folding origami cranes](#), your first one might take a while as you get used to it. By the third crane, your brain will recognise the process and be able to apply it faster. The next time you do origami of any sort, you will be able to recognise different folds and apply what you learned from the crane



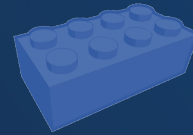
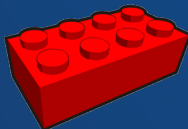
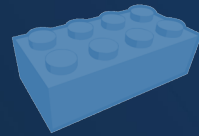


## Algorithm



What steps do you need to follow to create this?

**Algorithmic thinking is a critical skill and enables learners to construct new algorithms to solve given problems.**



### Cup of Tea Algorithm: Set of precise instructions to solve a problem or make something happen.



### Process

The answer that the participants provide is not important.

What is important is the process of discussing which line of code is correct:

**SAOU**



**Algorithmic thinking – people who solve jigsaw puzzles regularly may often devise their own strategies for solving puzzles quickly. What is yours?**

**This activity shows that teachers can already use things around their classroom to teach coding – we just need to use the right words and meaning!**

**Decomposition – Chunking up a problem into smaller more manageable chunks is an effective way to solve a problem.**

**Evaluation – testing out our strategies for solving puzzles and improving them along the way.**



**Abstraction – the completed puzzle forms an image, a model/representation of something and can hide the complexity of all the separate pieces required to fit together to complete it.**

**Generalisation – If we arrive at one strategy to solve a single puzzle quickly and effectively, will the same strategy work for other jigsaw puzzles or will we need to adjust our algorithm to accommodate multiple puzzle sets?**

**UNDOES**



# Evaluating and debugging

Evaluate process, find errors then problem solve solutions

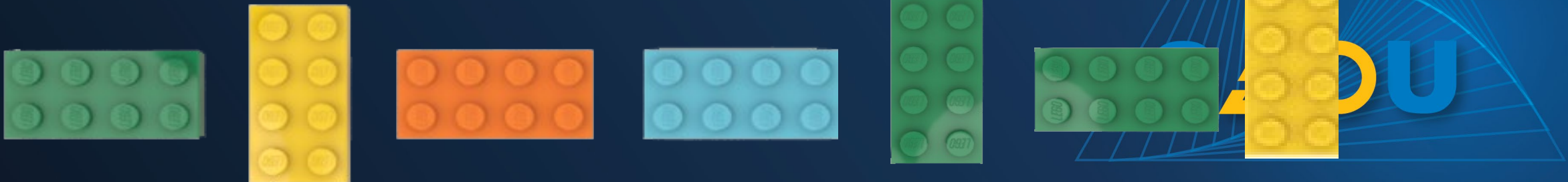
A single error in a line of code can cause a computer program to go haywire or stop working entirely. Luckily, coding has a process for dealing with errors called debugging. Debugging means finding and fixing errors in code. Mistakes in programming are known as bugs

Back in 1945 Grace Hopper, one of the female pioneers of computer science, found an error caused by a moth trapped between the points at Relay 70, Panel F, of a Calculator being tested at Harvard University. She removed the moth, and attached it to her test logbook, writing 'First actual case of bug being found', and so popularised the term 'debugging' for testing and fixing a computer program.


- Evaluation is to see whether the thing we made does what we wanted it to do.
- Debugging is figuring out how it does not work as we want it to.
- The process itself consists of identifying the cause of an error and fixing it.

## Debug Six Bricks

- Put your bricks in a 0 or 1 position (vertical and horizontal)
- We will show you a sequence of bricks below
- Can you figure out which parts of your sequence is different?
- Can you debug your Six Bricks code?

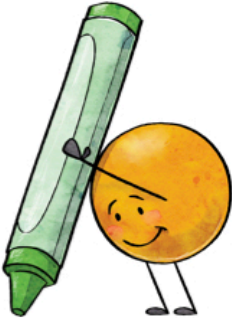







# **CODING**

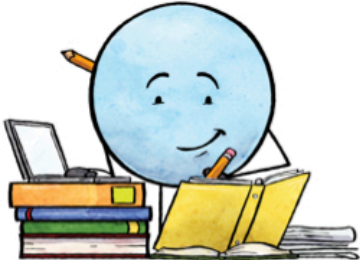
## **Teaches Kids**



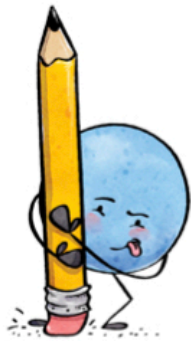
Risk Taking [www.carlyandadam.com](http://www.carlyandadam.com) Creativity




Critical Thinking



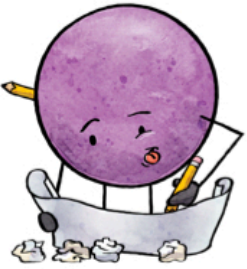
Story Telling and Retelling



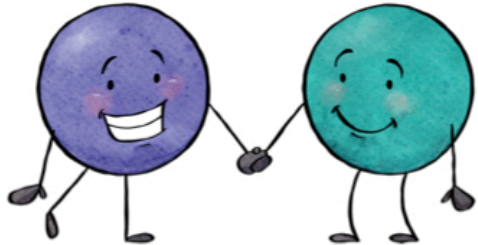
Problem Solving



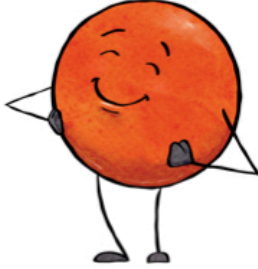
Math Processes



Perseverance



Teamwork



Self-Confidence

*carly and Adam*

**SAOU**

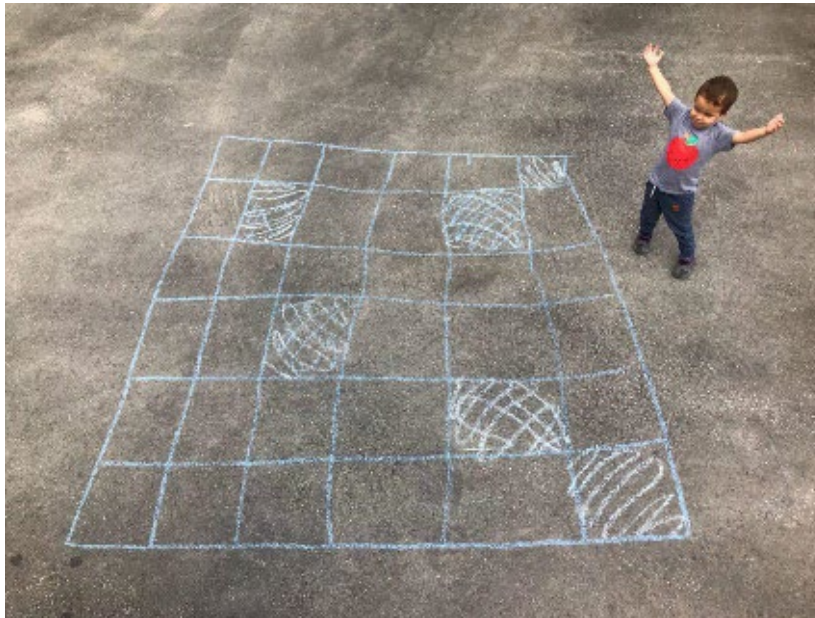


# Coding

## Computational thinking

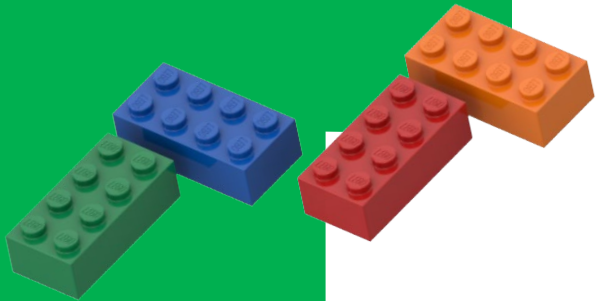


- Create a square grid (chalk, tape) - e.g., 6 x 6 square
- Colour some of the squares. These squares are the collection zones
- The programmer must code (tell) the robot to collect all the bricks
- The robot must bring the bricks to the end zone
- The robot can start anywhere on the grid. The programmer then must direct the robot to land on one of the brick squares.
- The programmer can only speak in 'code', meaning that they must tell the robot EXACTLY how to get there.



Code:

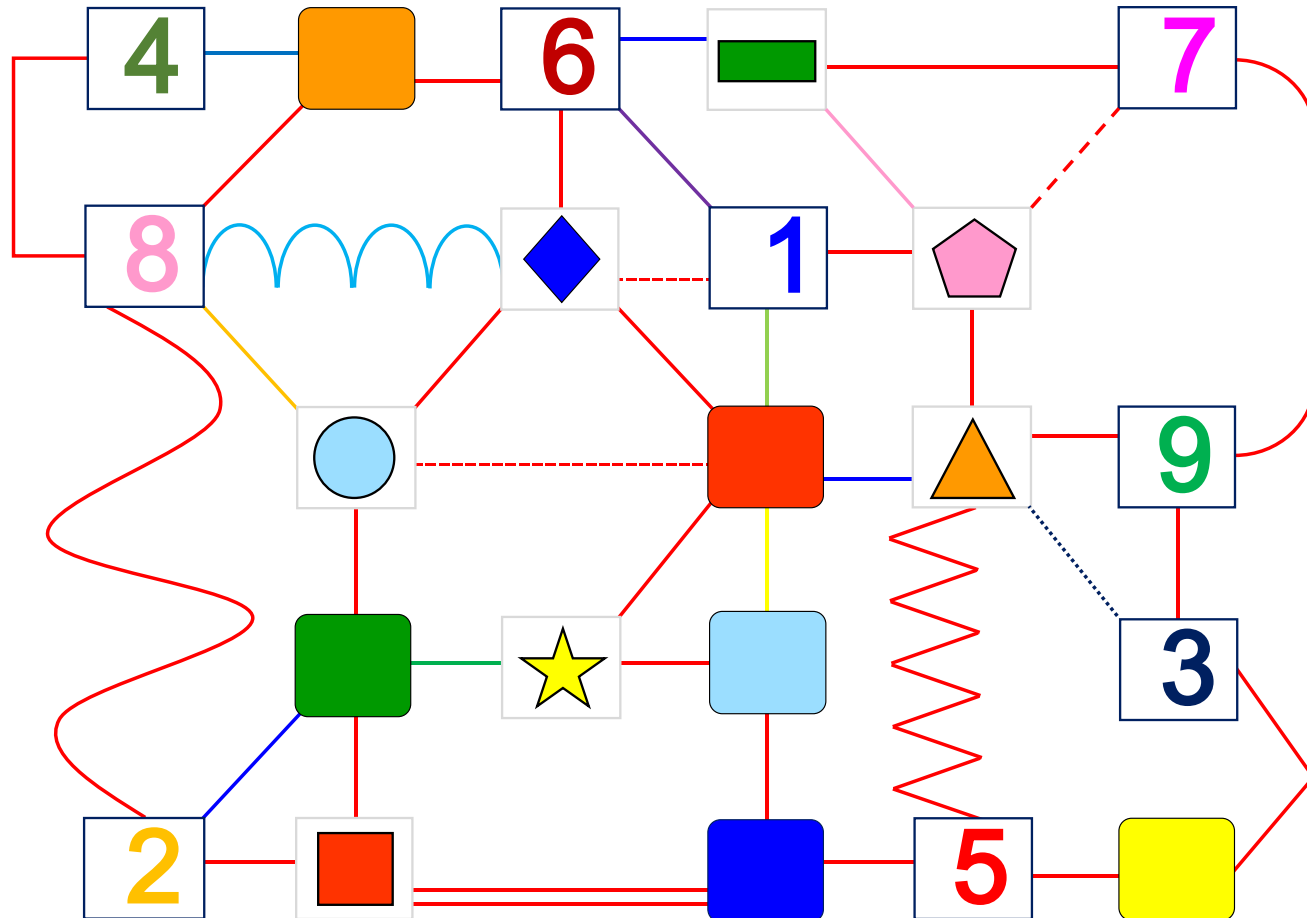
- Take x steps forward
- Take x steps backward
- Now Turn
- Turn left
- Turn right





# Moving Around Mat - Algorithms

**An algorithm is a set of instructions given to a computer so that the computer will execute a specific task. We use algorithms every day and all the activities we do can be explained in a series of algorithms.**



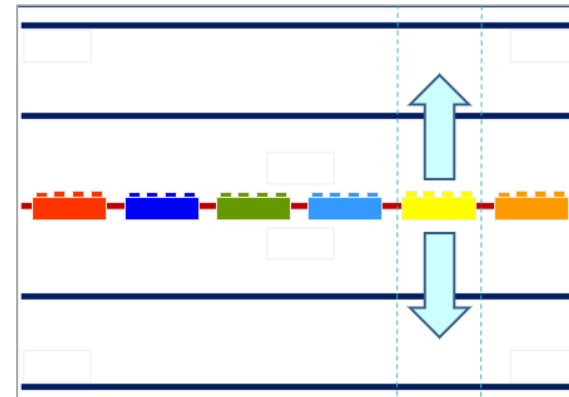
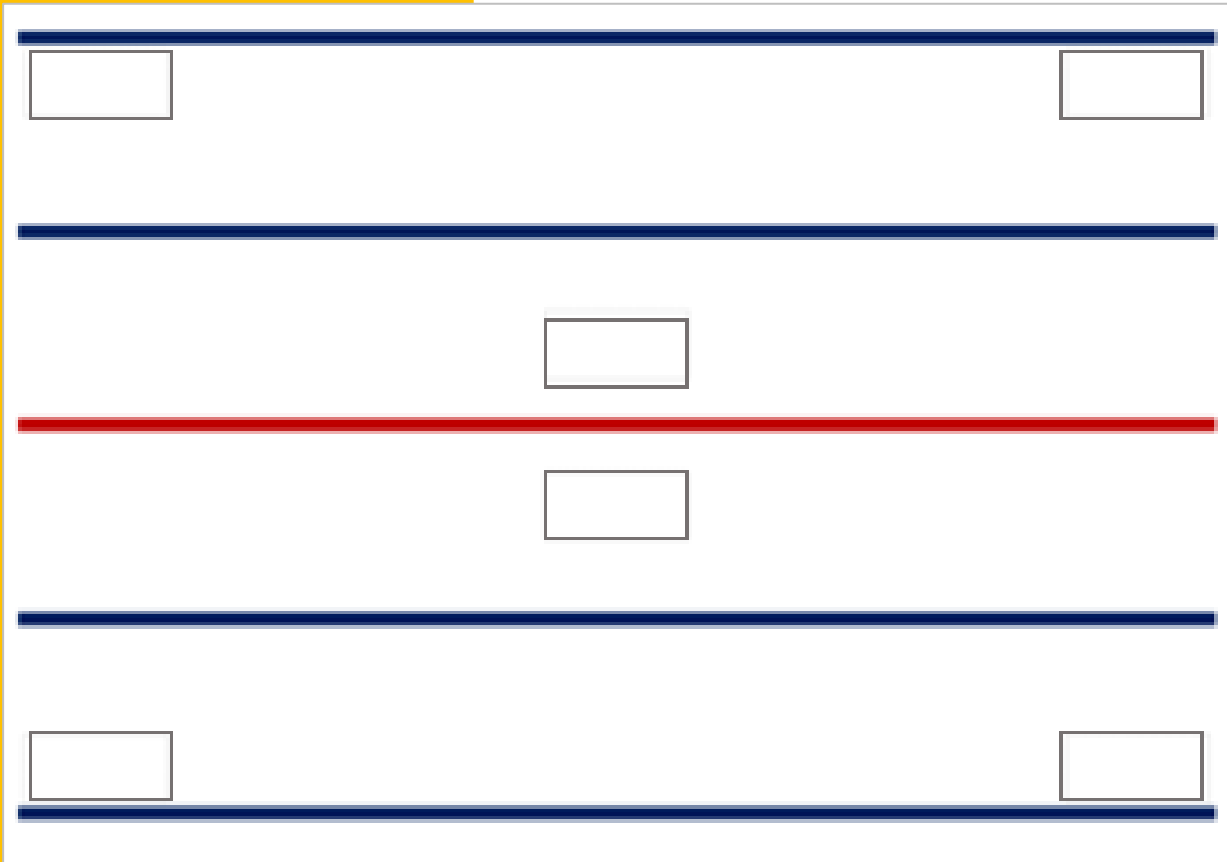




# Developing computational thinking

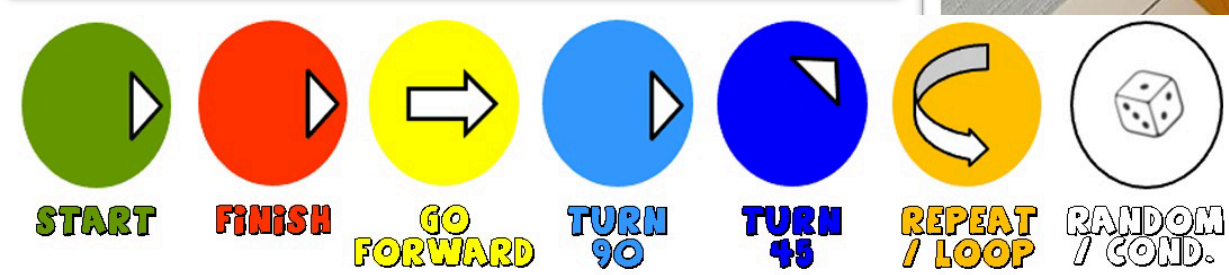
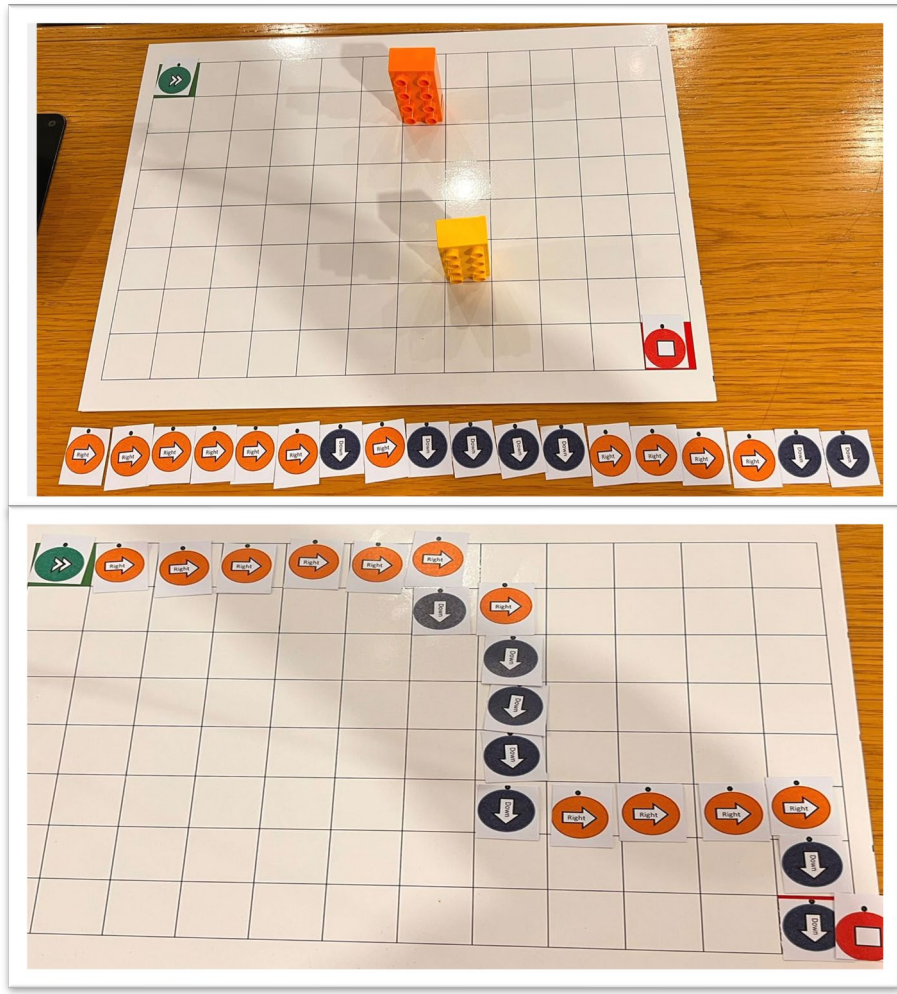
## Crossing the B-Line

At a foundational level, before moving to technology, it is important that children work with objects or manipulatives to understand space, directionality, process, sequencing, forwards/backwards, left/right, up/down, etc. Thinking computationally is not programming - Simply put, programming is a set of instructions that a computer must follow – the B-Line mat practices this. Children have to listen to a set of instructions and follow or execute these using the bricks. At the end of the set of instructions, children must evaluate where they went right or wrong (debug).





# Introducing the coding mats



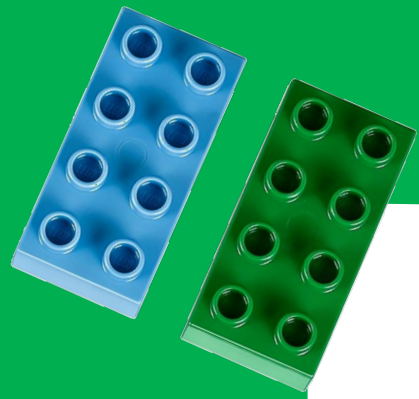




# Coding

## Coding Maze Mat

	A	B	C	D	E	F	G	H	I	J	K	
1												1
2												2
3												3
4												4
5												5
6												6
7												7
8												8
	A	B	C	D	E	F	G	H	I	J	K	



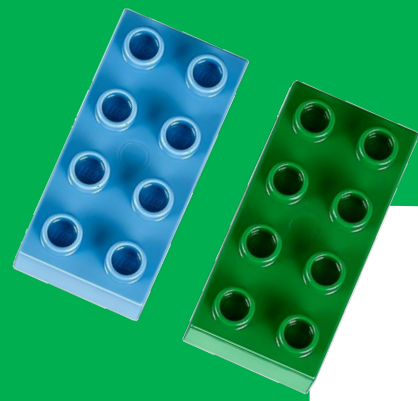




# Coding

## Computational thinking with symbols



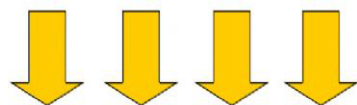
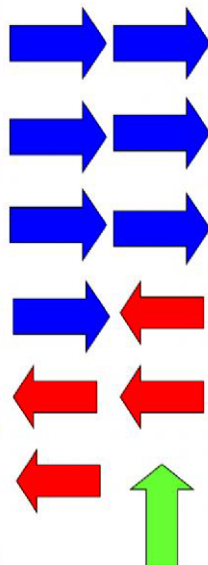
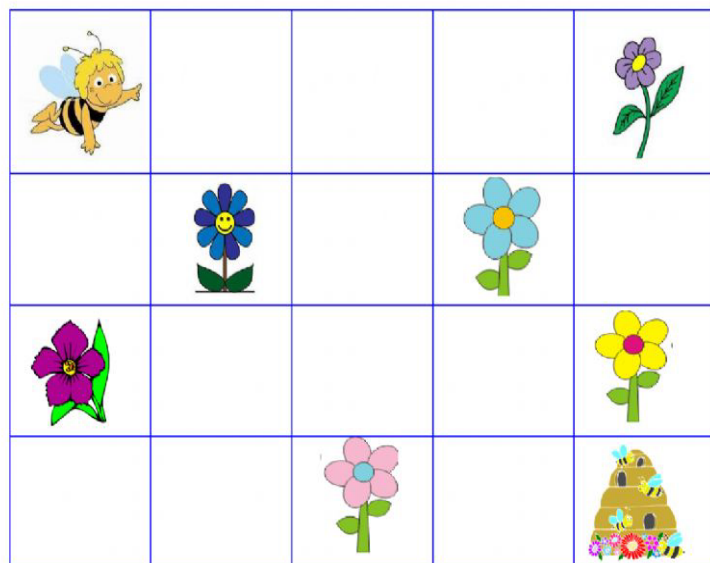







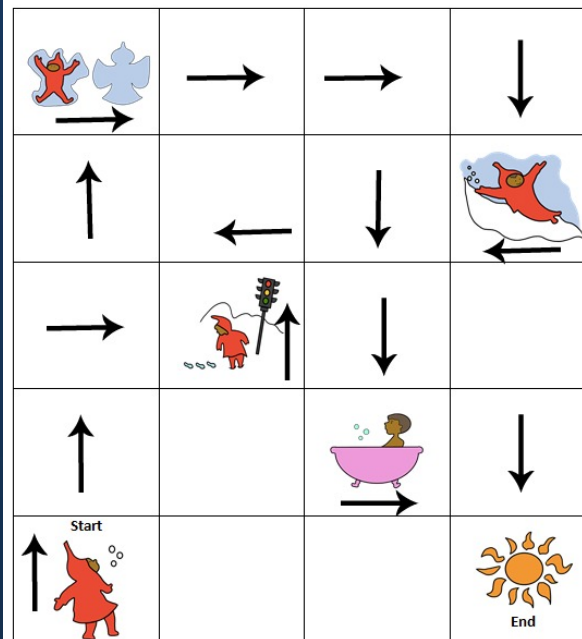


# SAOU



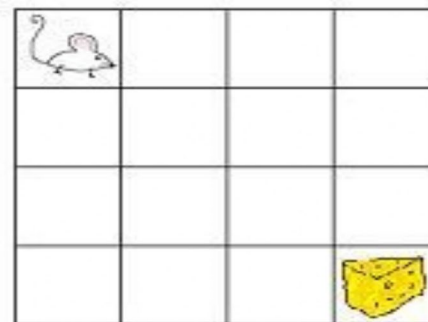
L'APE VUOLE RACCOGLIERE TANTO POLLINE.  
MOSTRALE LA STRADA PER ARRIVARE ALL'ALVEARE.  
ATTENTO A TOCCARE TUTTI I FIORI.

## The Snowy Day Algorithm Coding Activity



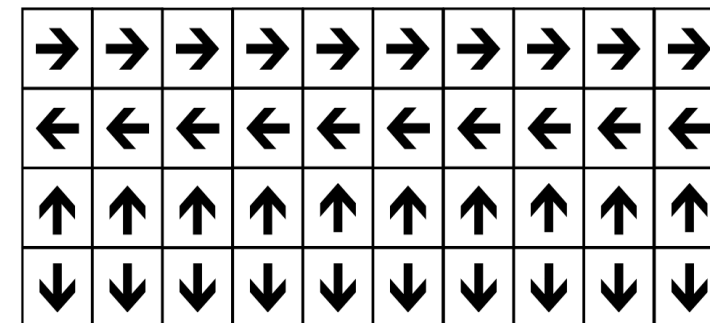
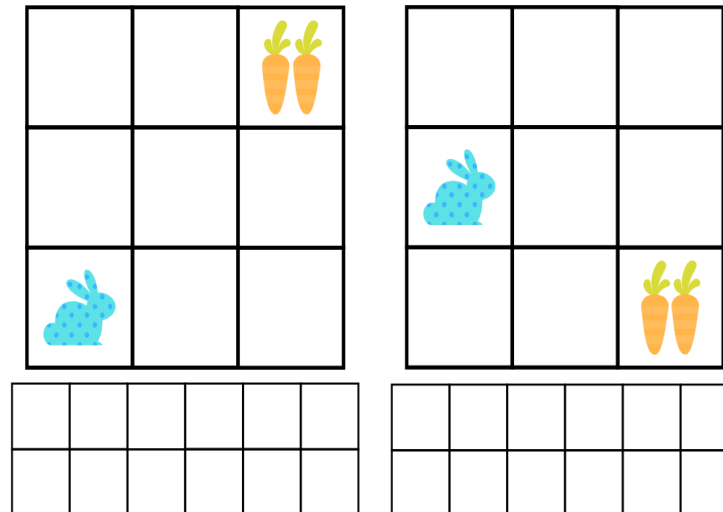
Created by JDaniel4mom.com

## Find the code



1. → → ↓ ↓ ↓
2. ↓ ↓ → ↓ → →
3. → ↓ → → ↓ →
4. → ↓ → ↓ → → ↓











## Rabbit To Carrot





# 10 Reasons to Teach Coding

By Brian Aspinall @mrspinall

- 5 Coding is inclusive & builds self-confidence. 
- 6 Coding supports many principles of mathematics. 
- 7 Coding teaches problem-solving and critical/analytical thinking skills. 
- 8 Coding is a new type of literacy and will be a large part of future jobs. 
- 9 Coding develops teamwork & collaborative skills. 
- 10 Coding can help humanity. 
- 1 Coding allows students to create content, not just consume it. 
- 2 Coding empowers students and gives them tools to express themselves in really cool ways. 
- 3 Coding teaches storytelling with games and animations. 
- 4 Coding is a place for students to take risks & fail safely. 

**BONUS:** Coding gives you SUPERPOWERS!



I code! What's  
your superpower?

The job I will be doing in 10 years'  
time, probably doesn't exist yet,  
but I know I'm ready for the  
future because I'm part of building it."





# SAOU



012 023 1333



saou@saou.co.za



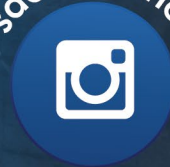
076 127 1921



SAOU National



saouteachers



SAOU YouTube

