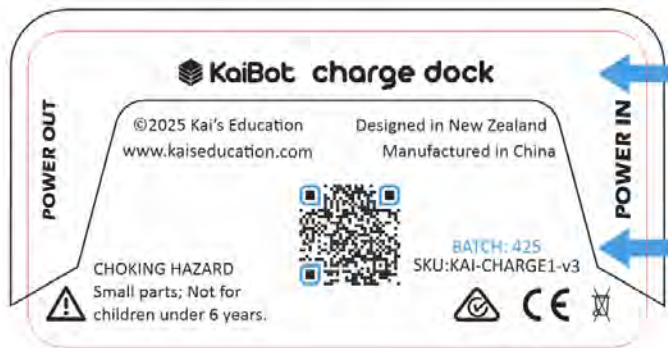


b.1 Charge via Autonomous Charging Dock



KaiBot exhibits remarkable intelligence in self-charging, capable of autonomously driving itself to its companion charging dock when placed on the purple charge tile, much like a mini-Roomba.

1. Snap and align the purple magnetic tile to the charging dock. Check that the "charge dock" icon  is next to the charge dock.
2. Connect the magnetic USB cable to the **left-hand side** of the dock and provide USB power, check that the green LED on the charge dock lights up, otherwise try the other side of the charge dock. Charge Docks can be nested together, refer to nested charge docks.
3. Switch on KaiBot & simply place him onto the purple charge tile any way you like. KaiBot is super smart and can drive itself onto the charging dock without any help, no matter how you place it. (as long as the purple tile is aligned correctly)
4. Once charging begins, the lights will flash green and when it's fully charged, it will drive itself off.
5. Snap the blue KaiTiles to the other side of the purple charge tile
6. Using the Kainundrum lite or Kainundrum.com you can view the charge dock in the virtual World and you can also drive to the charge dock.

b.2 Nest multiple Charge Docks together

Watch this short [video](#) on how to use the Charge Docks.

With the Charging docks you can connect multiple docks together via their magnetic connections. You then only need to use one charging cable and you will be able to charge multiple KaiBots.

Using a standard USB port (.5-1 amp) with one Charging dock (.5 amps) will take about 2 hours to charge and you will have 1 hour playtime

When nesting 4 Charging docks together you can use a higher USB amp (2 amp) and it will still take 2 hours to charge all 4 KaiBots.



C. Charging Case for 10 x KaiBots



- Plug the included mains cable into the back of the included charge adapter in the case.
- Confirm that all charging docks show a green LED to indicate power is active.
- Switch off all KaiBots before charging. This ensures that when the mains switch is turned off, all bots remain powered down, fully charged, and ready for use.

Charging Tips

If KaiBot is completely flat, switch it off, and refer to charging via USB cable section, leave on charge for around 2 hours to restore battery condition.

KaiBot can also be charged when switched off. Again push KaiBot onto the dock and he'll switch on and start charging or use the included charging cable.

■ **When KaiBot is full**, he will move away from the dock and stop charging. If KaiBot is full, he will not automatically navigate to the charging dock when placed on the purple charge tile.

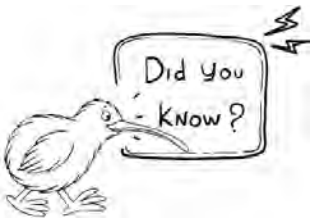
■ **Nap time**, If you don't keep KaiBot entertained, he'll start getting sleepy. After 10 minutes of inactivity, he'll turn off the screen and go into a light sleep to conserve battery. To wake him up, just move him around on a coding card or tile. After 30 mins of inactivity, KaiBot goes into a deep sleep and can only be woken by switching it off and back on again. In the deep sleep mode, the battery from a full charge can last around 2 days.

■ **KaiBot making funny noises, clicking and not switching on correctly?** KaiBot's battery is exhausted and needs a good 2-hour charge. Even if the battery says full, leave KaiBot on charge for a couple of hours when this happens.

■ **Not charging**. If the charging dock does not have power (green LED) then KaiBot will fail to dock and charge.

■ **Not docking correctly**: If the charging dock is not aligned correctly to the purple tile, then KaiBot might fail to dock correctly.

■ **Note**: Magnets located inside the dock, purple charge tile and Kai-Tiles should help align up, but you might have to correct it slightly for better docking.

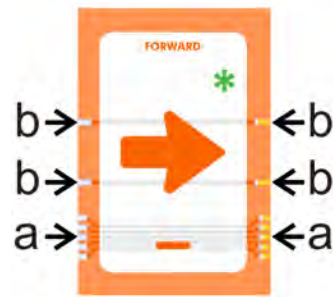
 New Zealand launches rockets into space?	New Zealand firm Rocket Labs launched satellites into orbit from its own private launchpad on New Zealand's Mahia Peninsula.
---	---

Coding Cards

There are 3 editions of coding cards:

- A. Basic cards included with each KaiBot.
- B. Advanced Coding Cards, available separately.
- C. Braille Coding Cards, available separately.

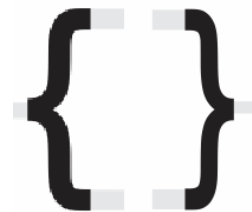
Elements of Coding Cards



Connectors

These cards also have a coating of magic dust from New Zealand that lets KaiBot know what card he is reading. The cards are specially printed to have a coordinate system and card ID, so KaiBot can read each card.

- a. All the coding cards use a generic connecting system, much like a circuit board. They connect to each other from left(silver) to right(gold).
- b. The middle connectors help to show what card can be connected next.



What do Curley Braces do?

Curly braces play a big role in code structure within popular programming languages such as Python, C++ and more. In languages like C curly braces {} are used to create program blocks used in flow control. In Python, curly braces are used to define a data structure called a dictionary.



What does the green asterisk * mean?

Any card that displays the green asterisk, lets you know you can add a number card directly afterwards or a “read variable” or a “read tile number” card.

Tip: Placing two numbers cards together makes an integer. E.g. (Forward *) (2) (2) would move forward 22 spaces.

Types of Coding Cards

Object-Oriented Programming (OOP) is a style of programming that involves structuring code into logical, self-contained objects. In object-oriented programming, data structures or objects are defined, each with its own properties or attributes. Each object can also contain its own procedures or methods.



The Structure of a Coding Card Program

Data Types and Variables

Programming is all about manipulating data, but what is data? Data is information that we store in our computer programs. For example, your name is a piece of data, and so is your age. The color of your hair, how many siblings you have, where you live, all of these things are data. In the coding cards, we only have one type of data and that is **Numbers**. In coding languages like Python or Javascript, you can also have other data types like Strings and Booleans.

Strings are used to represent text like your name. A string can look like this:

```
"Hello KaiBot"
```

Booleans are a value that can either be True or False. E.g. Are you wearing a hat. Boolean looks like this:

```
true;
```

Numbers within the coding cards are treated as integers. Fractions and decimals are not integers. A **Number** can be a positive or negative **Number**. E.g. -7 is an integer. A **Number** in Python looks like this:

```
99;
```

Variables

In computer programming, a variable has a name and contains a value. Think of a box containing a number of marbles. The box is the **name** and the number of marbles in the box is a **Number**. A variable can be of different data types, **Numbers**, **strings** and **Booleans**. A variable might look like this in Python:

```
Box = 75;
```

```
First_Name = "KaiBot";
```

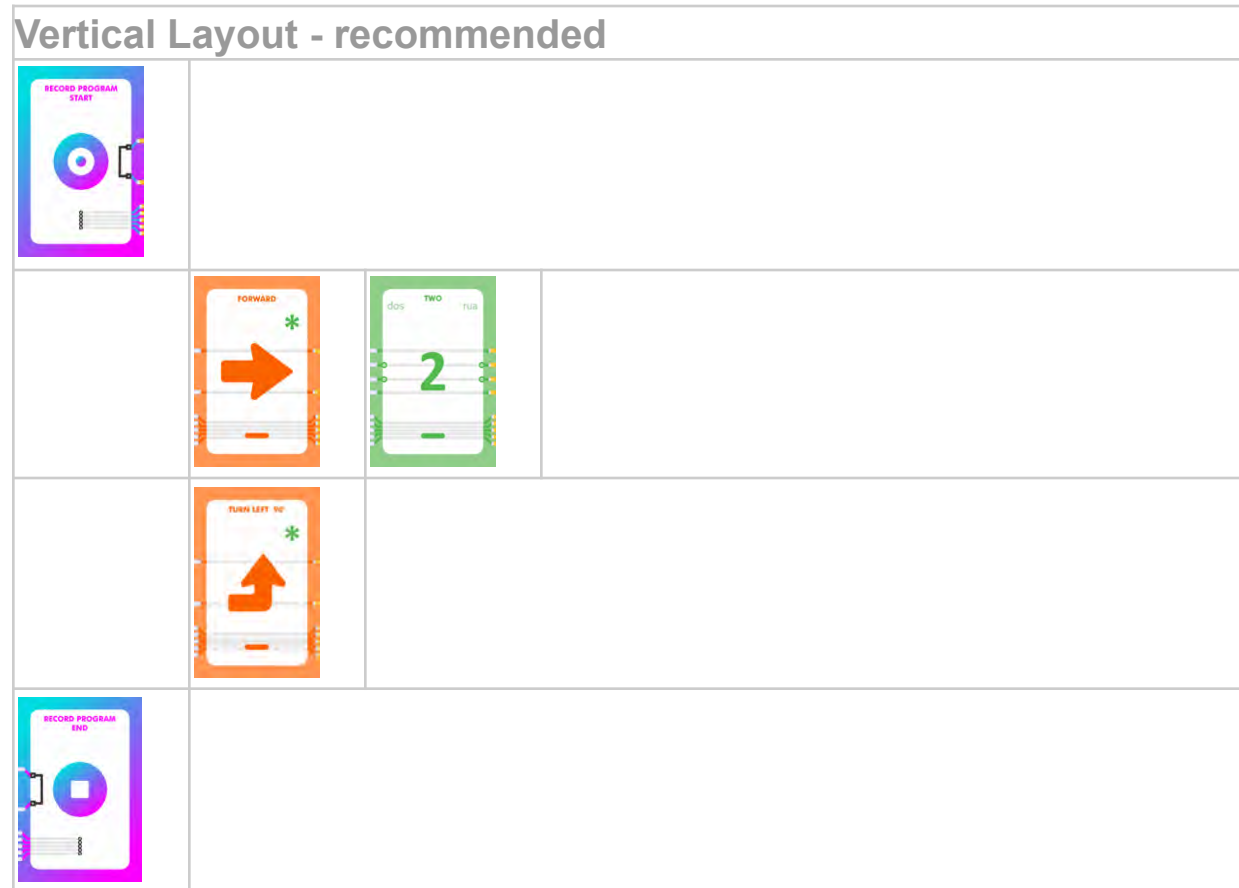
```
Hat = true;
```

How to correctly structure your coding cards

There are two ways that you can layout the coding cards. The easiest way is to lay them out one after the other left to right, horizontally. The benefit of this method is, that it doesn't use much space up but longer programs become difficult to understand. This might be easy for you as you get started. A programmer would normally code in a vertical indented method as shown below.



The best way to layout coding cards is in a vertical fashion, with grouped elements indented. This is much easier for anyone else to be able to read and understand the code, and once the student progresses into text-based coding this then becomes natural for them. It does take up a lot of desk space and helps with the read ability of the code.

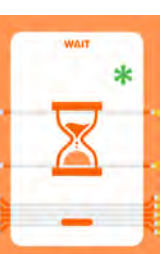


Check out some of the [examples](#) to see how you can improve your code layout.

Basic Coding Card Explainer

The following describes what each card does and how they interact with each other. The below describes the Standard Coding Card Pack.

CONTROL CARDS	
	This is your goto card to start your program! KaiBot must read this card first.
	And like all goods stories, they all must come to an end! Make sure you end your program with this card. If you have KaiBot paired with Kainundrum.com , then your coding card program is converted to both Blockly and text based Python code!
	KaiBot loves this card! Once you tap KaiBot on the Run Program card, you have a 3-second countdown before KaiBot runs your assembled program. Tip: Make sure KaiBot is facing the correct way. The yellow switch is his rear. This card cannot be used inside your program.
	If KaiBot needs to be stopped in his tracks, pick him up and place him on this card to stop your program from running. You can run the same stored program again, by placing KaiBot on the Run Program card again. This card cannot be used inside your program.

	MOVEMENT CARDS
	This card will get KaiBot to move forwards by 15cm or one tile length. You can also add a * number card, a “Read variable” or a “Read Tile” card afterwards, to move forward a set * number of times. E.g. Forward, 12; moves KaiBot forward 12 spaces. 
	Use this card to turn right and once again you'll see the green * to tell KaiBot how many turns. 1 turn = 90 degrees, 2 turns = 180 degrees, 3 turns = 270 degrees. Try also using “read variable” and see what happens if the variable is a negative number?
	The writer got bored explaining right and left and just used a search & replace in the document. Use this card to turn left and once again you'll see the green * to tell KaiBot how many turns. 1 turn = 90 degrees, 2 turns = 180 degrees, 3 turns = 270 degrees. Try also using “read variable” and see what happens if the variable is a negative number? Right?
	If you want to reach for the Moon then you going the wrong way if you use this card. Add a * number card next to this card and you're sure to make the distance between the Moon & KaiBot even greater.
	This card makes KaiBot pause for a set * number of seconds defined by adding a * number card or a “read variable” card.

MOVEMENT CARDS

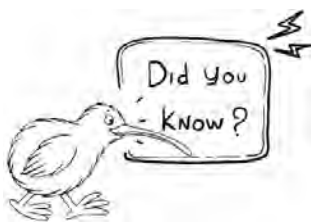
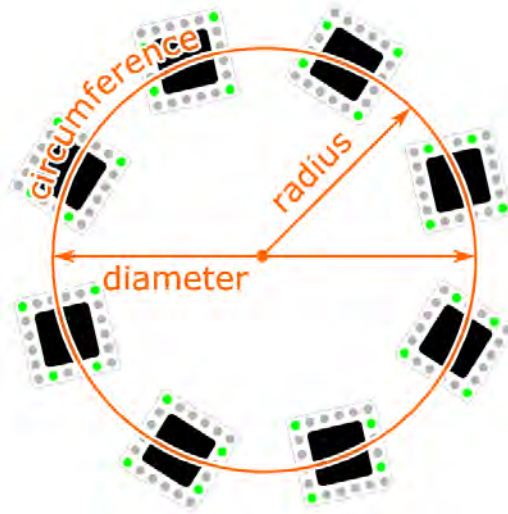


Radius - The radius is the distance from the center point to the edge of the circle.

Diameter - The diameter is a straight line that goes across the circle and through the center. It's twice the length of the radius.

Using this card makes KaiBot drive in a complete circle based on the radius defined by the * **number** card.

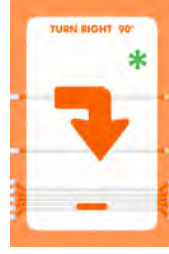
E.g. Adding * **number** 5, will make KaiBot drive in a complete large circle, based on the radius of 5 cm away from the centre point of the circle. Radius 1 would make KaiBot drive in a small circle.



More sheep than people.

The livestock industry is one of the main industries in the country, particularly sheep. Did you know that for every person living in New Zealand there are at least 10 sheep? The population of New Zealand is 5 million, so can you calculate how many sheep we have?

LOOP CARDS	
	Loop repeats the code that is between the “Loop Start” and “Loop End”, not including the * number cards. Adding * number card after this card, defines how many times the loop runs. . You can replace the * number card for a “Read variable” or a “Read Tile” card E.g. Loop Start 12 {Move Forward} Loop End; this moves KaiBot forward 12 spaces. Loop is different to the Conditional Repeat.
	Loop End is used to close the braces {} of the loop.

LOOP EXAMPLE				
				
This example shows (move forward, turn right), 5 times in the loop.				

NUMBER CARDS	
	Number cards can be added directly after any card that displays the green * They can be used along with movement, loops, conditionals, and variables.

NUMBER EXAMPLE



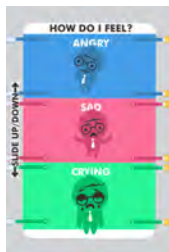
In the above example, adding 2 and then 5 directly after the forward card, makes KaiBot move forward 25 spaces.

SOCIAL EMOTIONAL LEARNING (SEL)



How Do You Feel Today? Some children struggle to communicate their needs and emotions, particularly younger children. Let your kids express their feelings using KaiBot to show their feelings.

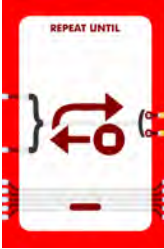
Slide KaiBot up or down the card to program KaiBot to perform that emotion. Only the last emotion selected will be recorded.



Slide KaiBot up or down the card to program KaiBot to perform that emotion. Only the last emotion selected will be recorded. We would rather you don't feel sad, and speak to a teacher or adult if you require help.

Advanced Coding Card Pack / Braille (optional)

A star below refers to the “★Advanced Coding Card Pack”. This pack also includes the basic cards.

★Advanced Coding Card Pack	REPEAT CARDS
	Repeat is different to a Loop. A loop loops the code a certain number of times, and you cannot exit the loop. A repeat will continue to repeat forever, or until a true condition is met. The code that you want to be repeated goes in between the braces {}
	This card is used to end the repeat, it represents the end brace } After this card, you need to add the condition that needs to be met. Note the small open bracket on the right (
	This card is used to close the bracket) and close the condition.

REPEAT EXAMPLE						
						

REPEAT EXAMPLE

In the above example, because 2 will never be greater than 5, KaiBot keeps moving forward. You can replace one of the numbers with a variable card.

★ Advanced
Coding Card
Pack

COLOR MIXING CARDS



Computer coding involves many different knowledge and skills. Have you ever tried color mixing with coding? On computers, a very common color-coding system is the RGB color model. The RGB color model is an additive color model in which red, green and blue light are added together in various ways to reproduce a broad array of colors. The name of the model comes from the initials of the three additive primary colors, red, green, and blue.

When you run your code, the KaiBots screen will display the resulting color, as well as the RGB, LED lights will also display the color. If you have KaiBot paired with Kainundrum.com and run your code, the virtual KaiBot can paint with your newly mixed color in Picasso mode.



Place KaiBot on this card to select the percentage of red you want to display or mix with the other colors. If you only want to display a percentage of red, then don't add the other color mixing cards.



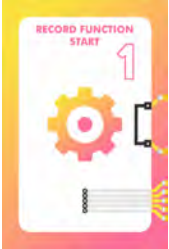
Place KaiBot on this card to select the percentage of green you want to display or mix with the other colors. If you only want to display a percentage of green, then don't add the other color mixing cards.



Place KaiBot on this card to select the percentage of blue you want to display or mix with the other colors. If you only want to display a percentage of blue, then don't add the other color mixing cards.

★ Advanced Coding Card Pack	COLOR MIXING CARDS
	Always close with the Color Mixer End card

COLOR MIXING EXAMPLE	
	
In the above example, adding 100% of Red and 100% of Blue will result in yellow	

★ Advanced Coding Card Pack	FUNCTION CARDS
	There are two sets of function cards. Set 1 and Set 2. Functions are sets of code that contain one or more steps to complete a specific action. Functions can be small, simple, straightforward or long, complicated, and multifaceted. Functions can contain all the regular coding cards within them, excluding the Start, End & Run Program cards. The [bracket opens the function and any cards after this must then be closed with a] Record Function End. Think of functions as a different program or sub routine that can be called. Always record functions, before recording your main code.

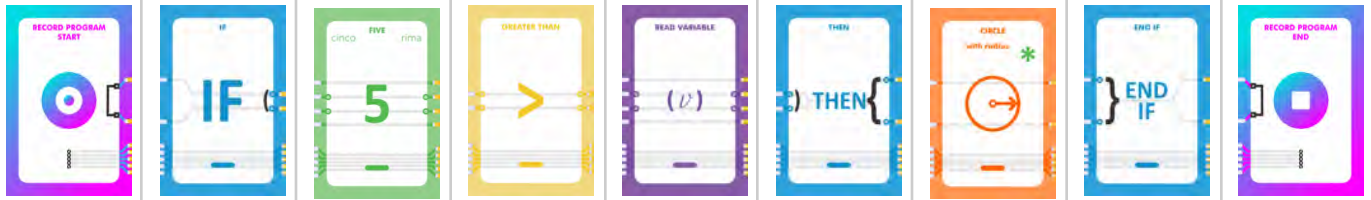
★ Advanced Coding Card Pack	FUNCTION CARDS
 A yellow and orange card with a gear icon and the text 'RECORD FUNCTION END' and a large '1'.	Place your code between the Record Function Start [and the] Record Function End.
 A yellow and orange card with a play button icon and the text 'RUN FUNCTION' and a large '1'.	Place the Run Function card in your main code. Refer to the below example. Placing KaiBot on the Run Function card will not make him immediately perform the function. The function is treated as a subset program of your main code.
 A blue and green card with a gear icon and the text 'RECORD FUNCTION START' and a large '2'.	There are two sets of function cards. Set 1 and Set 2. Functions are sets of code that contain one or more steps to complete a specific action. Functions can be small, simple, straightforward or long, complicated, and multifaceted. Functions can contain all the regular coding cards within them, excluding the Start, End & Run Program cards. The [bracket opens the function and any cards after this must then be closed with a] Record Function End. Always record functions, before recording your main code.
 A blue and green card with a gear icon and the text 'RECORD FUNCTION END' and a large '2'.	Place your code between the Record Function Start [and the] Record Function End.
 A blue and green card with a play button icon and the text 'RUN FUNCTION' and a large '2'.	Place the Run Function card in your main code. Refer to the below example. Placing KaiBot on the Run Function card will not make him immediately perform the function. The function is treated as a subset program of your main code.

FUNCTION EXAMPLE

In the above example, the function is first recorded with KaiBot, and then the main program is recorded. When KaiBot runs the main program, the main program will also run the function. The example would have moved the robot forward 10 spaces. Functions can be used to save space in your code.

★ Advanced Coding Card Pack	COMPARISON OPERATORS
 The image shows a yellow Scratch 'Equal To' comparison block. It has two input fields on the left and right, and a single output field at the bottom. The text 'EQUAL TO' is at the top, and a yellow equals sign is in the center.	Are used to compare two values to each other and can also be used to control the flow of a program based on the value of a number or variable. The 'equal to' comparator is also referred to in coding as two equal signs back-to-back '=='. It is not setting two things equal but is returning 'true' if two things are equal. You will often want to compare two things and do something if they are equal, like checking if the current time is equal to the alarm you have set.
 The image shows a yellow Scratch 'Not Equal To' comparison block. It has two input fields on the left and right, and a single output field at the bottom. The text 'NOT EQUAL TO' is at the top, and a yellow not-equal sign is in the center.	The not-equal-to operator (!=) returns true if the operands don't have the same value; otherwise, it returns false.
 The image shows a yellow Scratch 'Less Than' comparison block. It has two input fields on the left and right, and a single output field at the bottom. The text 'LESS THAN' is at the top, and a yellow less-than sign is in the center.	The less-than and greater-than comparators return 'true' when the comparison is true and false when the condition is not-true.
 The image shows a yellow Scratch 'Greater Than' comparison block. It has two input fields on the left and right, and a single output field at the bottom. The text 'GREATER THAN' is at the top, and a yellow greater-than sign is in the center.	The less-than and greater-than comparators return 'true' when the comparison is true and false when the condition is not-true.

COMPARISON EXAMPLE



In the above example, IF 5 is greater than the value stored in the variable, then the KaiBot turns in a circle?
Remember to always set your variable to a value first.

★ Advanced Coding Card Pack	CONDITIONAL STATEMENTS
 The IF coding card is a blue-bordered card with a white background. It features the word 'IF' in large blue letters. Above it, in smaller blue letters, is 'IF'. To the right of 'IF' is a small blue bracket. The card has a blue header and footer.	A statement helps a computer decide what to do next. A condition statement has an If/Then format. For example, If $v = 1$, then turn left <code>if(condition here is true){ run all of these code statements } else{ run these statements }</code> An “if” statement creates a branch within your program so that your code 'makes decisions'. It tests a condition and, if that condition is true, some code statements will run. If the condition is false, the 'if' statement is passed over.
 The THEN coding card is a blue-bordered card with a white background. It features the word 'THEN' in large blue letters. Above it, in smaller blue letters, is 'THEN'. To the right of 'THEN' is a small blue bracket. The card has a blue header and footer.	The “then” statement performs the following code if the condition is true.
 The ELSE coding card is a blue-bordered card with a white background. It features the word 'ELSE' in large blue letters. Above it, in smaller blue letters, is 'ELSE'. To the right of 'ELSE' is a small blue bracket. The card has a blue header and footer.	The optional “else” statement performs the following code if the condition is false.
 The END IF coding card is a blue-bordered card with a white background. It features the words 'END IF' in large blue letters. Above them, in smaller blue letters, is 'END IF'. To the right of 'END IF' is a small blue bracket. The card has a blue header and footer.	This ends and closes the entire IF statement.

CONDITIONAL EXAMPLE

In the above example, IF 5 is greater than the value stored in the variable, then the KaiBot turns left, else KaiBot turns right. Remember, the “Else” card is optional.

★ Advanced Coding Card Pack	VARIABLES
	A variable is a container that holds a value, such as a piece of text or a number. The value can change, which is why it's stored in a variable. The Coding Cards only work with integers, whole numbers that are positive or negative. This card is used to set v to a * number card, placed directly afterwards.
	The Read Variable card is used to load the current value of the variable.
	This card returns the current Kai-Tile number that KaiBot is sitting on. Note: When your program runs the “Read Tile Number”, then whatever tile KaiBot is sitting on, that’s the tile number it will read.
	Use this to increase the variable value, by the * number placed after this card.
	Use this to decrease the variable value, by the * number placed after this card.



KaiBot in Kainundrum Cards

The following cards can be used alongside Kainundrum.com when you [pair KaiBot](#), then your virtual KaiBot will mirror the programmed actions from your physical KaiBot.

Requirements:

KaiBot

PC, MAC or Chromebook with Bluetooth

Magnetic [Kai-Tiles](#) that are laid out to match the virtual world in Kainundrum.

Tip: You can also create your own layouts in Kainundrum using the Game Builder.

KAINUNDRUM CARDS		
	Add this card alongside your code to turn on or off your virtual KaiBot Electromagnet. This card acts as a switch, having KaiBot scan the top of the card is on, and the bottom of the card is off. The Electromagnet can be used to pick up cubes in the game.	
	Add this card alongside your code to turn on or off Picasso mode. This card acts as a switch, having KaiBot scan the top of the card is on, and the bottom of the card is off. Picasso mode allows you to paint the path your virtual robot drives. If you want to move the brush to the centre of the robot in order to draw shapes, then use Blockly under tags: <pre>picasso pen centered on robot true</pre>	
	Add this card alongside your code to turn on or off the virtual Bumper. This card acts as a switch, having KaiBot scan the top of the card is on, and the bottom of the card is off. The Bumper allows you to use Blockly or Python to sense if the virtual KaiBot has collided with an object.	
	The engineers who programmed Kainundrum thought it would be fun to hide secret Easter Eggs inside the program. We asked them what they are, but they wouldn't tell us, after all, it is a secret. If you find out what it is, please do let us know, the writer is still looking. <i>Why did the programmer take a break? Because he wanted a byte to eat.</i>	

Debugging Code

KaiBots built-in interpreter will record each scanned card, alongside this process, is KaiBots bug identifier.

The debugger is in charge of constantly checking the scanned code and ensuring that it's valid.

When it comes to teaching children problem-solving, few methods are as powerful as learning to debug code. The process of finding bugs and implementing debugging strategies is great practice for problem-solving as well as helping children understand how to solve problems they come across in the real world.



When working with computers and writing code, something can always go wrong. It's a major part of computational thinking and problem-solving. When things don't work as you wanted them, it's okay. "Just keep on swimming".

On 9 September 1947 the word "bug" was used by Grace Hopper in her diary, where she referred to a moth as a bug. As this bug was found in the computer, the term caught on and is now used to define errors or faults found in our computers.



Incremental Testing

No programmer wants to deal with a dozen or more bugs all at the same time. Moreover, they don't want to have to search hundreds of lines of code to find the source of a bug. This is why — instead of finishing a large project and then testing — programmers test the code as they go, breaking the code into smaller sections and testing each small section of code.

Even the most complicated computer program, video game, or algorithm can be decomposed into tiny steps. Coders take these individual steps, create them one at a time, and test each as they go. By following this best practice, even beginner coders can manage to debug with ease.

Once KaiBot finds a bug in your code, get the student to try and figure out why it's occurring. The best way to start on this is just to take a look at what the code is doing and get the student to try and perform and walk out the steps and sequences to understand what the code is supposed to do.

Rubber Duck Debugging

In software engineering, rubber duck debugging is a method of debugging code by articulating a problem. This method helps your students find bugs in their code. The term rubber duck refers to an entity that has less than half a brain and understands almost nothing about a given problem.



Rubber Duck Debugging is a programming methodology where you explain the task you are performing with all the details you are trying to achieve, line-by-line. By describing the problem the programmers force themselves to express their ideas in a clear way and step by step sequence. This explanation is called “telling the duck your problem”.

When a coder does so, the bug will be easily spotted as you will be forced to face the logic problems you might have ignored previously. All with the help of rubber duck programming.

Once you understand debugging you are on a path to becoming a better coder and problem solver. 🦆

Computer Programming Jargon

Syntax is the rules of how a programming language works. To put it simply, the syntax of a programming language defines how you should write the statements in a program. It is a set of rules that tell you what combinations of symbols are valid or invalid.

Here is an example of the construct of an English sentence. E.g. - What is your name?
Here we put the pronoun first, then the verb and so on. We don't say “Name your is what?” or you might sound like Yoda.

Forward 10; is valid
10 Forward; is invalid

A Syntax error is caused by how you wrote your code that breaks the rules of the syntax to make the code invalid.

Logical errors are problems with the logic where the program cannot make sense of what it was asked to do. If a program works differently than you expected it to work, there is a good chance that there's a logical error somewhere.

Magnetic snap together Kai-Tiles

All robots and even cars suffer from their wheels slipping on the ground, this slipping can cause the direction of the robot to go off course. Even a small amount of error can have a large effect when moving over a greater distance.

Screen-free without Kai-Tiles

KaiBot can be used without tiles, however, the robot might not always turn to the correct angle or move to the desired position.

Screen-free with Kai-Tiles

To get KaiBot to follow instructions, it's best that you use KaiBot with Kai-Tiles. **There is a plastic protective sheet on each tile; please remove it before you start using the tiles.** When using KaiBot with Kai-Tiles, KaiBot always knows where it is as KaiBot can read the tile number as well as the x & y position on the tile.



KaiBot without Kai-Tiles

When KaiBot is moving around just on a flat surface without KaiTiles, his wheels can slip and go off course. When you code him to turn left 90 degrees and he turns less or more than that, his wheels have slipped. Try using KaiBot on different surfaces and you'll see the effects of slipping more or less. Just like a car can lose traction on an icy road. If you have Kai-Tiles then you are in for a treat as KaiBot corrects this slippage.



Tip: Without Kai-Tiles, you won't be able to mirror KaiBot in Kainundrum.



KaiBot with Kai-Tiles

Did you know? These magnetic Kai-Tiles are coated with magic dust all the way from New Zealand (only kidding). However, there is a plastic protective sheet on each tile; please remove it before you start using the tiles.

The tiles and coding cards are actually specially printed, so KaiBot can read his x and y positions when placed on them.

KaiBot wheels can still slip on these tiles, however, after every movement, KaiBot compares the destination angle less the angle he is facing and then makes a slight correction.

Each tile is uniquely numbered and uses a mathematical grid. It uses a Cartesian coordinate system with a two-dimensional plane with x & y

