

LEGO Education

EV3 Classroom Blocks

Types and Descriptions



Table of Contents	
Word Block Types	2
Word Block Description	4
Motor Blocks	4
Movement Blocks	7
Display Blocks	Error! Bookmark not defined.
Sound Blocks	14
Event Blocks	16
Control Blocks	20
Sensor Blocks	23
Operator Blocks	28
Variable Blocks	32
My Blocks	34
Weather Blocks	Error! Bookmark not defined.
More Motor Blocks	Error! Bookmark not defined.
More Movement Blocks.....	Error! Bookmark not defined.
More Sensors	Error! Bookmark not defined.
Music Blocks	Error! Bookmark not defined.
Log and Visualize Data Over Time	Error! Bookmark not defined.
Display Extension.....	Error! Bookmark not defined.

Word Block Types

The Scratch programming language is made up of different types of blocks, each represented by a different shape.

Hat Blocks



Hat Blocks are used to start a program. They have a rounded top so that other blocks can only be attached under them.

Stack Blocks



Stack Blocks perform the main commands in a program. They're the blocks that can make the motors move and the lights light up!

C Blocks



C Blocks are C-shaped blocks. They're placed between the beginning and the end of the loop or check whether a condition is "true." All of the C Blocks can be found in the *Control* category.

Reporter Blocks



Reporter Blocks hold values, which be a number or a string. Among other things, they can hold a sensor reading or store the value of a variable.

Boolean Blocks



Boolean Blocks are conditions that can either be true or false. They're used together with C Blocks to form the logic of a program.

Cap Blocks

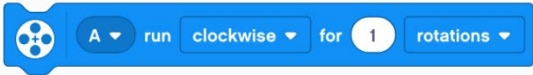

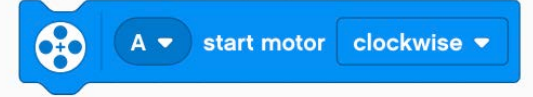





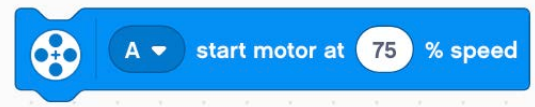
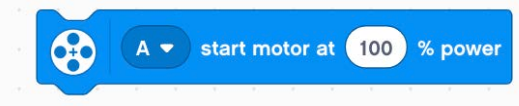
Cap Blocks are used to end scripts. They're shaped with a notch at the top and a flat bottom so that you can't place any blocks below them. There are two Cap Blocks, both which can be found in the *Control* category.




Block Stacks

A *block stack* is a number of blocks that have been put together.

Word Block Description

Motor Blocks	
<p>Motor Blocks either make the motors run or get information from the motors. The Motor Blocks category contains the most common Motor Blocks.</p>	
<h3>Run Motor for Duration</h3> 	<p>Runs one motor clockwise or counterclockwise for a specified number of rotations, seconds, or degrees.</p> <p>The motor speed is set by the Set Speed Block. The default speed is 75%.</p>
<h3>Run Motor for Duration at Speed</h3> 	<p>Runs one motor for a specified number of rotations, seconds, or degrees at a specified speed. A negative speed value runs the motor counterclockwise.</p>
<h3>Start Motor</h3> 	<p>Runs one motor clockwise or counterclockwise forever.</p> <p>The motor speed is set by the Set Speed Block. The default speed is 75%.</p>

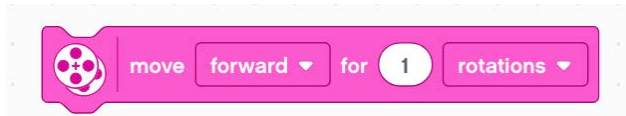
<p>Stop Motor</p> 	<p>Stops one motor from running. The motor will brake so that it quickly comes to a complete stop. The motor will not hold its position once it has stopped.</p>
<p>Set Motor Speed</p> 	<p>Sets the speed of one or more motors. The speed range is -100 to 100. Negative values will reverse the direction of the motor. If the speed hasn't been specified, the default value is 75%.</p>
<p>Set Motor Position at Stop</p> 	<p>Sets the action that the motor will perform when its current command completes. It can be set to either float or actively hold position when the motor stops.</p>
<p>Set Motor Position at Speed</p> 	<p>Starts running a motor at the specified speed until the motor is told to do something else or the program stops. A negative speed value runs the motor counterclockwise.</p>
<p>Start Motor at Power</p> 	<p>Runs a motor at a specified percentage of power until the motor is told to do something else or the program stops. Unlike the blocks that use the speed parameter, which regulates the power of the motor in order to maintain a specified speed, this block will keep the power level constant without regulation. A negative power value runs the motor counterclockwise.</p>

<h3>Reset Motor Degrees Counted</h3> 	<p>Resets the degree count of a motor to “0.” A motor’s degree count is equal to the motor’s relative position from where it started. The degree count is “0” whenever a program starts, or a motor is connected to an EV3 Brick. Turning the motor clockwise increases the count, while turning counterclockwise decreases the count.</p>
<h3>Motor Degrees Counted</h3> 	<p>Returns the number of degrees a motor has turned. The degree count is set to “0” when the program starts or is reset using the Reset Degrees Counted Block. Turning the motor clockwise increases the count, while turning counterclockwise decreases the count.</p>
<h3>Motor Speed</h3> 	<p>Reports the current speed of the motor. The value given is the motor’s actual speed, not the speed set by the Set Speed Block.</p>

Movement Blocks

Movement blocks enable you to run two motors in a synchronized motion. They're primarily used to move Driving Bases around. Only motors of the same type (e.g., two Medium Motors) can be synchronized.

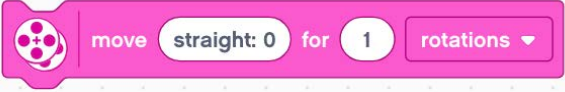
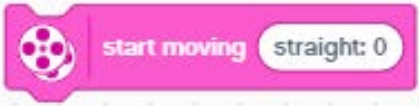
Move for Duration


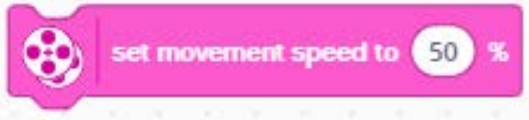
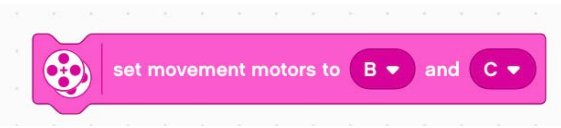
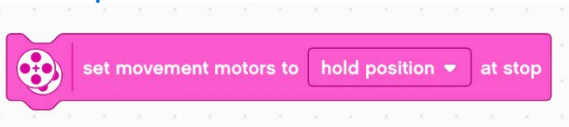


Moves a model forward or backward for a specified number of seconds, degrees or rotations.

Use the Set Movement Motors Block to match the Ports on your model that are connected to Large Motors. The default Ports are B (left) and C (right).

The speed of the motors is set by the Set Movement Speed Block. The default speed is 50%.

<h3>Move with Steering for Duration</h3> 	<p>Moves a model forward the specified number of seconds, degrees, or rotations with the specified steering. the possibility of steering.</p> <p>Higher steering values (i.e. +99 and -99) will make the arc path of the Driving Base sharper.</p> <p>Use a value of “0” to drive in a straight line. Using the values 100 and -100 will make the Driving Base pivot on itself.</p> <p>Use the Set Movement Motors Block to match the Ports on your model that are connected to Large Motors. The default Ports are B (left) and C (right).</p> <p>The speed of the motors is set by the Set Movement Speed Block. The default speed is 50%.</p>
<h3>Start Moving with Steering</h3> 	<p>Starts moving a model forward with the possibility of steering forever. Higher steering values (i.e. +99 and -99) will make the arc path of the Driving Base sharper.</p> <p>Use a value of “0” to drive in a straight line. Using the values 100 and -100 will make the Driving Base pivot on itself.</p>

<p>Stop Moving</p> 	<p>Stops all movement.</p> <p>Use the Set Movement Motors Block to match the Ports on your model that are connected to Large Motors. The default Ports are B (left) and C (right).</p>
<p>Set Movement Speed</p> 	<p>Sets the speed of a moving Driving Base. The speed range is -100 to 100. Negative values change the direction of the movement. The default value is 75%</p>
<p>Set Movement Motors</p> 	<p>Sets the movement motors for the Movement Blocks. The first specified Port sets the left motor, and the second specified Port sets the right motor. The default Ports are B (left) and C (right).</p>
<p>Set Movement Motors to Hold Position at Stop</p> 	<p>Sets the action that the movement motors will perform when their current command completes. It can be set to either float or actively hold the current position when the motors stop.</p>

Move with Steering for Duration at Speed



Moves a model the specified number of seconds, degrees, or rotations at the specified speed, with the specified steering.

Use the Set Movement Motors Block to match the Ports on your model that are connected to Large Motors. The default Ports are B (left) and C (right).

Higher steering values (i.e. +99 and -99) will make the arc path of the Driving Base sharper.

Use a value of "0" to drive in a straight line. Using the values 100 and -100 will make the Driving Base pivot on itself.

Move for Duration at Speed



Moves a model the specified number of seconds, degrees, or rotations at the specified speed for each motor.

The first speed value sets the speed of the left motor, and the second speed value sets the speed of the right motor.

Use the Set Movement Motors Block to match the Ports on your model that are connected to Large Motors. The default Ports are B (left) and C (right).

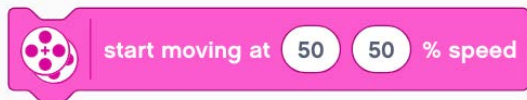
Start Moving with Steering at Speed



Starts moving a model at the specified speed, with the specified steering until the motors are told to do something else or the program stops.

Use the Set Movement Motors Block to match the Ports on your model that are connected to Large Motors. The default Ports are B (left) and C (right).

Start Moving at Speed



Starts moving a model at the specified speed for each motor until the motors are told to do something else or the program stops.

The first speed value sets the speed of the left motor, and the second speed value sets the speed of the right motor.

Use the Set Movement Motors Block to match the Ports on your model that are connected to Large Motors. The default Ports are B (left) and C (right).

Display Blocks

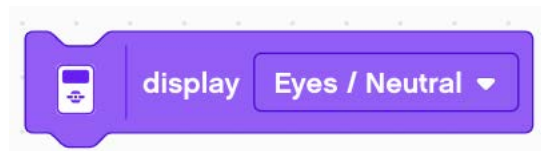
Display blocks enable you to display text or graphics on the EV3 Brick Display.

Display Image for Seconds



Displays the selected image on the EV3 Brick Display for the specified number of seconds.

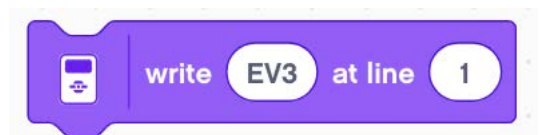
Display Image



Displays the selected image on the EV3 Brick Display. The image remains on the



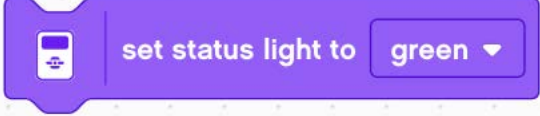
Display until it is overwritten by another block, the Display is cleared, or the program stops.

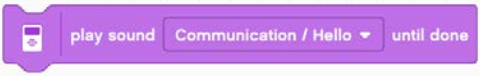
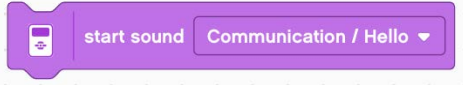
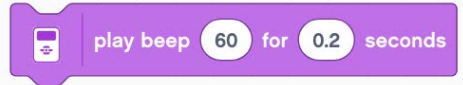
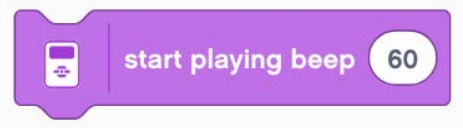
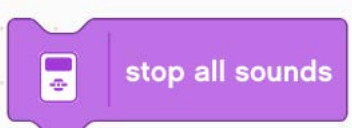
Write at Line



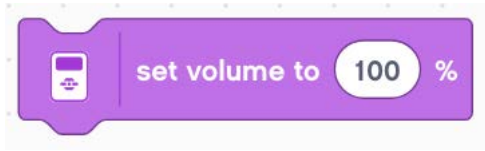
Displays the defined text at the specified line on the EV3 Brick Display. The line height is 10 pixels.

The text remains on the Display until it is overwritten by another block, the Display is cleared, or the program stops.

<h3>Write at Coordinates with Font</h3> 	<p>Displays the defined text in the specified font, at the specified XY coordinates on the EV3 Brick Display. The line height is 10 pixels.</p> <p>The text remains on the Display until it is overwritten by another block, the Display is cleared, or the program stops.</p>
<h3>Clear Display</h3> 	<p>Clears the EV3 Brick Display.</p>
<h3>Set Status Lights to Color</h3> 	<p>Sets the Brick Status Lights to the specified color and pulse option.</p> <p>Possible options are green, red, orange, green pulse, red pulse, and orange pulse.</p>

<h2>Sound Blocks</h2> <p>Sound blocks enable you to play sounds using the speaker inside the EV3 Brick.</p>	
<h3>Play Sound Until Done</h3> 	Plays a selected sound on EV3 Brick and waits until the sound is finished.
<h3>Start Sound</h3> 	Starts playing a selected sound on the EV3 Brick and immediately continues the programming stack.
<h3>Play Beep for Seconds</h3> 	Plays a beep tone on the EV3 Brick for a specified number of seconds.
<h3>Start Playing Beep</h3> 	Plays a beep tone on the EV3 Brick. The tone will keep playing until the speaker is told to do something else or the program stops.
<h3>Stop All Sounds</h3> 	This block stops all sounds that are currently playing on the EV3 Brick.

Set Volume



This block sets the volume of the sound. The default volume is 100%.

Event Blocks

Event blocks are comprised entirely of Hat Blocks, meaning that they're always the first block in a programming stack and other blocks can only be attached under them. Hat Blocks are necessary to start a programming stack and will be triggered when a specified event occurs.

When Program Starts



Runs all of the blocks attached to it, sequentially from top to bottom, when the program starts. The program can be started by using the Download and Run Button, or by selecting and running the program from the EV3 Brick.


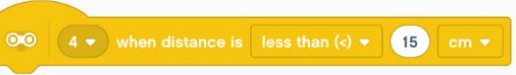

When Color Is









Runs the blocks attached to it when the Color Sensor detects a specified color. The sensor can detect:

- Black
- Brown
- Blue
- (4) Light Blue
- Green
- Yellow
- Red
- White
- No color
- Any change in color

This block will only trigger in the event of detecting the specified color. This means that the block will not re-trigger as long as the detected color remains unchanged.

<h3>When Pressed</h3>  <p>The image shows a yellow 'When Pressed' block from the LEGO Mindstorms software. It features a touch sensor icon on the left, a dropdown menu with the number '1', the text 'when', and another dropdown menu with the word 'pressed'.</p>	<p>Runs the blocks attached to it when the Touch Sensor is pressed or released.</p> <p>This block will only trigger at the specified event. This means that the block will not re-trigger as long as the state of the Touch Sensor remains unchanged.</p>
<h3>When Distance Is</h3>  <p>The image shows a yellow 'When Distance Is' block. It includes an ultrasonic sensor icon, a dropdown menu with the number '4', the text 'when distance is', a dropdown menu with 'less than (<)', a text input field with the number '15', and a dropdown menu with 'cm'.</p>	<p>Runs the blocks attached to it when the Ultrasonic Sensor's distance to an object is less than, greater than, equal to, or changes more than the specified distance in centimeters or inches.</p> <p>This block will only trigger at the specified event. This means that the block will not re-trigger if the distance remains unchanged.</p>
<h3>When Angle Is</h3>  <p>The image shows a yellow 'When Angle Is' block. It features a gyro sensor icon, a dropdown menu with the number '2', the text 'when angle is', a dropdown menu with 'less than (<)', a text input field with the number '45', and a degree symbol.</p>	<p>Runs the blocks attached to it when the Gyro Sensor's angle is less than, greater than, equal to, or changed more than the specified angle.</p> <p>This block will only trigger at the specified event. This means that the block will not re-trigger if the angle remains unchanged.</p>

<h3>When Brick Button Pressed</h3> 	<p>Runs the blocks attached to it when the specified Brick Button is pressed or released.</p> <p>This block will only trigger at the specified event. This means that the block will not re-trigger as long as the Brick Button remains in its current state.</p>
<h3>When</h3> 	<p>Runs the blocks attached to it when its Boolean condition is true.</p> <p>This block will only trigger in the event of the condition becoming true. This means that the block will not re-trigger if the condition remains true.</p>
<h3>When I Receive Message</h3> 	<p>Runs the blocks attached to it when the specified message is broadcasted by either the Broadcast Message or the Broadcast Message and Wait Block.</p>
<h3>Broadcast Message</h3> 	<p>Broadcasts the specified message. All of the When I Receive Message Blocks that have been set to the specified message will activate.</p> <p>This Broadcast Block sends the specified message and immediately proceeds to the next block, unlike the Broadcast Message and Wait Block, which will wait until all of the programming stacks “listening” for this message have finished before proceeding.</p>

<p>Broadcast Message and Wait</p> 	<p>Broadcasts the specified message. All of the When I Receive Message Blocks that have been set to the specified message will activate.</p> <p>This Broadcast Block sends the specified message and waits until all of the programming stacks “listening” for this message have finished before proceeding to the next block in its own stack. This is unlike the Broadcast Message Block, which will proceed without waiting.</p>
<p>When Timer</p> 	<p>Runs the blocks attached to it when the timer exceeds the specified value.</p> <p>This block will only trigger in the event of the timer exceeding the specified value.. This means that the block will not re-trigger as long as the timer is above the specified value.</p>

Control Blocks

The Control Blocks category contains all of the blocks that can modify the linear flow of block execution, such as “wait for” structures, loops and conditions.

Wait for Seconds



This block pauses the programming stack for a specified number of seconds – it supports whole numbers and decimals.

Wait Until



This block pauses the programming stack until a specified Boolean condition is true.

Repeat Loop


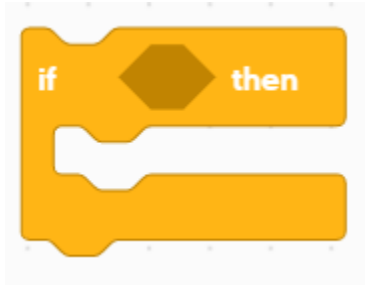
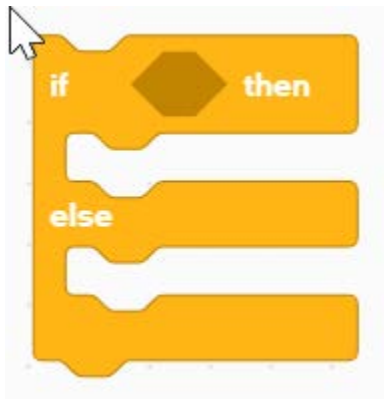




This block pauses the programming stack until a specified Boolean condition is true.

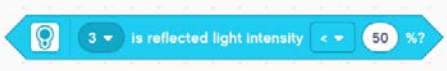

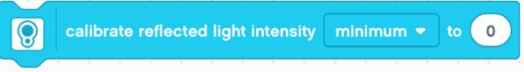

Forever Loop









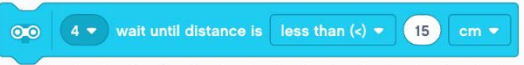
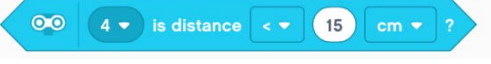

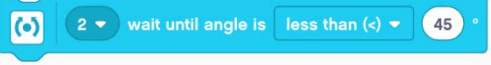


All of the blocks held inside this block will loop forever. The only way to stop the loop is to interrupt the program by pressing the Stop button, or by using the Stop All Block.







<h3>Repeat Until Loop</h3>  <p>The image shows a yellow 'repeat until' block. It has a diamond-shaped slot for a Boolean condition on the right side. Below the condition slot is a large white arrow pointing to the right, indicating the direction of the loop. The block has a notch on the left side for interlocking with other blocks.</p>	<p>All of the blocks held inside the block will loop until the specified Boolean condition is true. When the specified condition becomes true, blocks beneath it will play.</p>
<h3>If Then</h3>  <p>The image shows a yellow 'if then' block. It features a diamond-shaped slot for a Boolean condition on the left side, followed by the word 'then'. Below this is a large white arrow pointing to the right. The block has a notch on the left side.</p>	<p>This block will check whether the specified Boolean condition is true.</p> <p>If the condition is true, all of the blocks held inside it will play. If the condition is false, the blocks will be ignored.</p>
<h3>If Then Else</h3>  <p>The image shows a yellow 'if then else' block. It has a diamond-shaped slot for a Boolean condition on the left side, followed by the words 'if then' and 'else'. Below these are two large white arrows pointing to the right, representing the two execution paths. The block has a notch on the left side.</p>	<p>This block will check whether the specified Boolean condition is true.</p> <p>If the condition is true, the blocks held inside the first space will play and the stack will continue. If the condition is false, the blocks inside the second space will play.</p>
<h3>Stop</h3>  <p>The image shows a yellow 'stop and exit program' block. It is a simple rectangular block with the text 'stop and exit program' and a small downward-pointing triangle on the right side. It has a notch on the left side.</p>	<p>Stops all running programming stacks, its own programming stack, or stops and exits the program.</p>


<p>Stop Other Stacks</p>  An orange rectangular block with a notch on the left side and a bump on the right side. The text "stop other stacks" is written in white lowercase letters in the center.	<p>This block stops all programming stacks expect its own.</p>
--	--

Sensor Blocks	
Sensor Blocks receive information from the sensors.	
<p>Is Reflected Light Intensity?</p> 	<p>Returns “true” if the Color Sensor’s reflected light intensity is greater than, equal to, or less than the specified percentage.</p>
<p>Reflected Light Intensity</p> 	<p>Returns the Color Sensor’s reflected light intensity, as a percentage. The returned value is scaled according to the Color Sensor’s reflected light intensity calibration, which can be changed using the Calibrate Reflected Light Intensity Block.</p>
<p>Set Reflected Light Intensity Calibration</p> 	<p>Calibrate the Color Sensor’s sensitivity to reflected light intensity. Minimum and maximum thresholds can be calibrated.</p>
<p>Wait Until Color Is</p> 	<p>Pauses the programming state until the Color Sensor detects the specified color. The sensor can detect</p> <ul style="list-style-type: none"> – Black – White – Blue – Brown – Green – Yellow – Red – No color

<p>Is Color?</p> 	<p>Returns true if the Color Sensor detects the specified color. The detectable colors are:</p> <ul style="list-style-type: none"> – Black – White – Blue – Brown – Green – Yellow – Red – No color
<p>Color</p> 	<p>Returns the Color Sensor's detected color, as a number. Possible values are:</p> <ul style="list-style-type: none"> 0 – No color 1 – Black 2 – Blue 3 – Green 4 – Yellow 5 – Red 6 – White 7 – Brown
<p>Is Ambient Light Intensity</p> 	<p>Returns true if the Color Sensor's ambient light intensity is greater than, equal to, or less than the specified percentage.</p>
<p>Ambient Light Intensity</p> 	<p>Returns the Color Sensor's ambient light intensity, as a percentage.</p>
<p>Wait Until Pressed</p> 	<p>Pauses the programming stack until the Touch Sensor is pressed or released.</p>

<p>Is Pressed?</p> 	<p>Returns true if the Touch Sensor is pressed.</p>
<p>Wait Until Distance Is</p> 	<p>Pauses the programming stack until the Ultrasonic Sensor's distance to an object is less than, greater than, equal to, or changed more than the specified distance in centimeters or inches.</p>
<p>Is Distance?</p> 	<p>Returns true is the Ultrasonic Sensor's distance to an object is greater than, equal to, or less than the specified distance in centimeters or inches.</p>
<p>Distance?</p> 	<p>Returns the Ultrasonic Sensor's distance to an object in centimeters or inches. The sensor's range is 0-255 centimeters or 0-100 inches.</p>
<p>Wait Until Angle Is</p> 	<p>Pauses the programming stack until the Gyro Sensor's angle is less than, greater than, equal to, or changed more than the specified angle.</p>
<p>Is Angle?</p> 	<p>Returns true if the Gyro Sensor's angle is greater than, equal to, or less than the specified angle.</p>
<p>Angle</p> 	<p>Returns the Gyro Sensor's angle in degrees.</p>

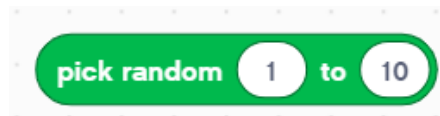
<p>Reset Angle</p> 	<p>Returns the Gyro Sensor's angle to "0."</p>
<p>Angular Velocity</p> 	<p>Returns the Gyro Sensor's angular velocity, in degrees per second.</p>
<p>Wait Until Brick Button is Pressed</p> 	<p>Pauses the programming stack until the specified Brick Button is pressed or released.</p>
<p>Is Brick Button Pressed?</p> 	<p>Returns true if the specified Brick Button is pressed or released.</p>
<p>Brick Button</p> 	<p>Returns the Brick Button pressed, expressed as a number. Possible values are: 0 = No Button 1 = Left Button 2 = Center Button 3 = Right Button 4 = Up Button 5 = Down Button</p>
<p>Timer</p> 	<p>This block reports the time, in seconds, since the program started. The timer restarts every time the program restarts. Use the Reset Timer Block to manually restart the timer.</p>

<p>Reset Timer</p>  A blue block with a notch on the left side and a bump on the right side, containing the text "reset timer" in white.	<p>This block resets the timer.</p>
---	-------------------------------------

Operator Blocks

Operator blocks perform all of the mathematical operations that can be done using numerical values.

Pick Random Number



Picks a random number within the specified range, including both endpoints. For example, with a specified range of 1 to 3, the block could return a 1,2 or 3.

Plus



Adds two values and returns the result.

Minus



Subtracts the second value from the first value and returns the result.

Multiply













Multiplies two values and returns the result.

Divide

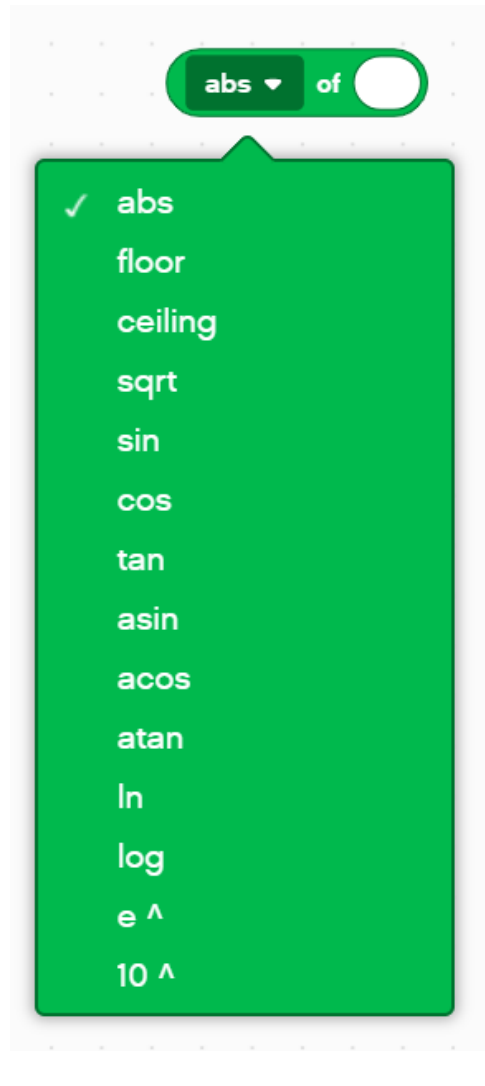


Divides the second value by the first value and returns the result.

<p>Less Than</p>  <p>The image shows a green 'Less Than' block with a white circle on the left and the number '100' on the right, with a less-than sign (<) in between.</p>	<p>Returns true if the first value is less than the second value.</p>
<p>Equal</p>  <p>The image shows a green 'Equal' block with a white circle on the left and the number '100' on the right, with an equals sign (=) in between.</p>	<p>Returns true if the first value is equal to the second value.</p>
<p>Greater Than</p>  <p>The image shows a green 'Greater Than' block with a white circle on the left and the number '100' on the right, with a greater-than sign (>) in between.</p>	<p>Returns true if the first value is greater than the second value.</p>
<p>And</p>  <p>The image shows a green 'And' block with two dark green hexagonal ports on either side and the word 'and' in the center.</p>	<p>Returns true if both Boolean values are true.</p>
<p>Or</p>  <p>The image shows a green 'Or' block with two dark green hexagonal ports on either side and the word 'or' in the center.</p>	<p>Returns true if at least one of the Boolean values are true.</p>
<p>Not</p>  <p>The image shows a green 'Not' block with a dark green hexagonal port on the right and the word 'not' on the left.</p>	<p>Returns true if the Boolean value is false.</p>





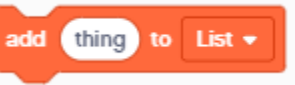
<p>Join Strings</p> 	<p>Concatenates, or “joins” the values together and returns the result. For example if “hello” and “world” were input into the block, it would return “helloworld”). To return “hello world” input “hello ” and “world” or “hello” and “ world” including a space after hello or before world.</p>
<p>Length of String</p> 	<p>Returns the number of characters contained in the given string. For example, if the input is “LEGO,” the block returns “4.”</p>
<p>Mod</p> 	<p>Returns the remainder of the division when the first value is divided by the second value.</p> <p>For example, when <i>10</i> is the first input and <i>3</i> is the second, the block will report <i>1</i> because <i>10</i> divided by <i>3</i> gives a remainder of <i>1</i>.</p> <p>Negative numbers behave a little differently because a remainder must always be positive (e.g., $-10 \text{ mod } 3$ equals <i>2</i>, not <i>-1</i> because you have to multiply <i>3</i> by <i>-4</i> to have any remainder at all).</p>
<p>Round</p> 	<p>Rounds the given number to the nearest integer. It follows the standard rules of .5 or higher are rounded up, whereas decimals of less than .5 are rounded down.</p>

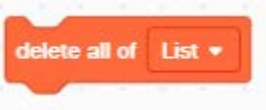
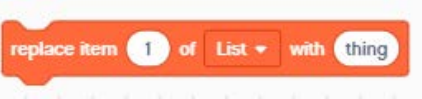

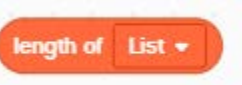
Math Functions





Performs the specified math function on a given number and reports the result.

Abs – absolute value
Floor – lowest value
Ceiling – highest value
Sqrt – square root
Sin – sine
Cos – cosine
Tan – tangent
Asin – inverse sine
Acos – inverse cosine
Atan – inverse tangent
Ln – natural logarithm
Log – logarithm
E[^] - exponential (base e)
(inverse functions of Ln)
10[^] - exponential (base 10)
(inverse function of log)

Variable Blocks	
<p>The Variable Blocks category contains all of the blocks linked to variables and lists (arrays).</p>	
<p>Variable</p> 	<p>Returns the value of a variable. Whenever a variable is created, a version of this block appears with the variable's given name on it, resulting in a version of this block for every variable. Each version of the block holds its "assigned" value.</p>
<p>Set Variable To</p> 	<p>Sets the variable to the specified value. The variable can be either a string or a number.</p>
<p>Change Variable By</p> 	<p>Changes the variable by a specified value. If the variable is a text string and not a number, it is set to the value the variable was to be changed by (casting the string to a "0").</p>
<p>List</p> 	<p>Returns the value of a list. Whenever a list is created, a version of this block appears with the list's given name on it, resulting in a version of this block for every list.</p>
<p>Add Item to List</p> 	<p>Adds an item containing the specified text or number to a list.</p>

<p>Delete All Items in List</p> 	<p>The block deletes all of the items in the specified list.</p>
<p>Replace Item at Index in List with Another Item</p> 	<p>Replaces the specified item's content with a specified value.</p>
<p>Value of Item at Index in List</p> 	<p>Returns the value of the specified item in a specified list.</p>
<p>Length of List</p> 	<p>Returns the number of items contained in the specified list.</p>

<h2>My Blocks</h2> <p>My Blocks are part of the Variable Blocks category. You can create your own definition of a My Block.</p>	
<h3>Define Block</h3> 	<p>This block allows you to create your own block. The <i>My Block</i> is the group of blocks that is attached to the <i>Define</i> Block.</p>
<h3>My Block</h3> 	<p>Runs a user-defined My Block. The block is defined by the Define Block.</p>