Scratch: Lessons for Finch 1 (All)

Finch Basics

The future of Scratch 2.0 offline and ScratchX is uncertain.
We encourage you to use Snap! with your Finch Robot to ensure long-term support.

Lesson Key

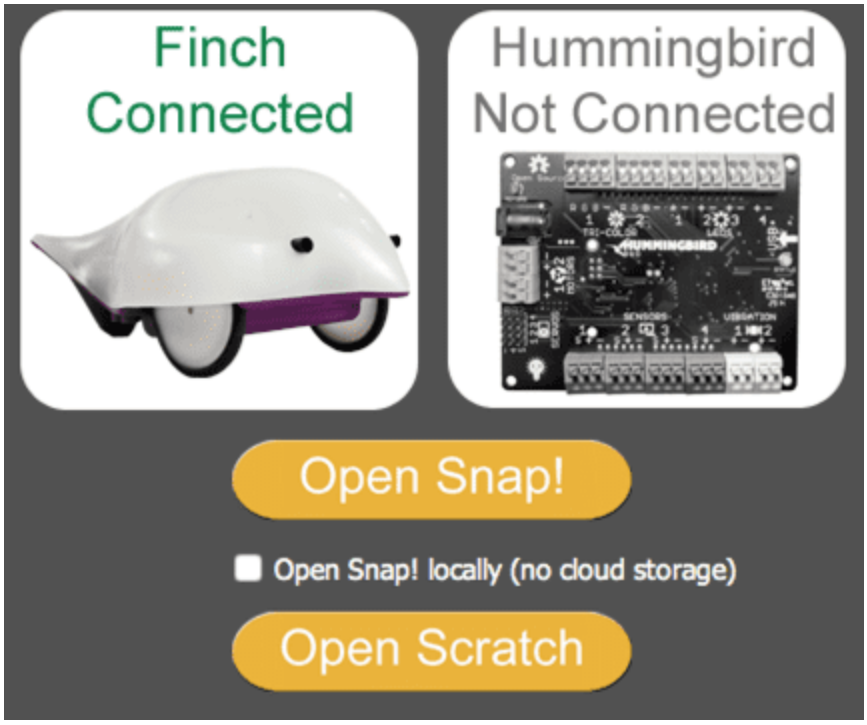
Lesson key available for teachers.

Table of Contents

- Programming in Scratch
- Saving Your Work
- Moving the Finch
- Events
- Creating Longer Scripts
- Turning with the Finch
- Loops
- Color with the Finch
- Sound with the Finch

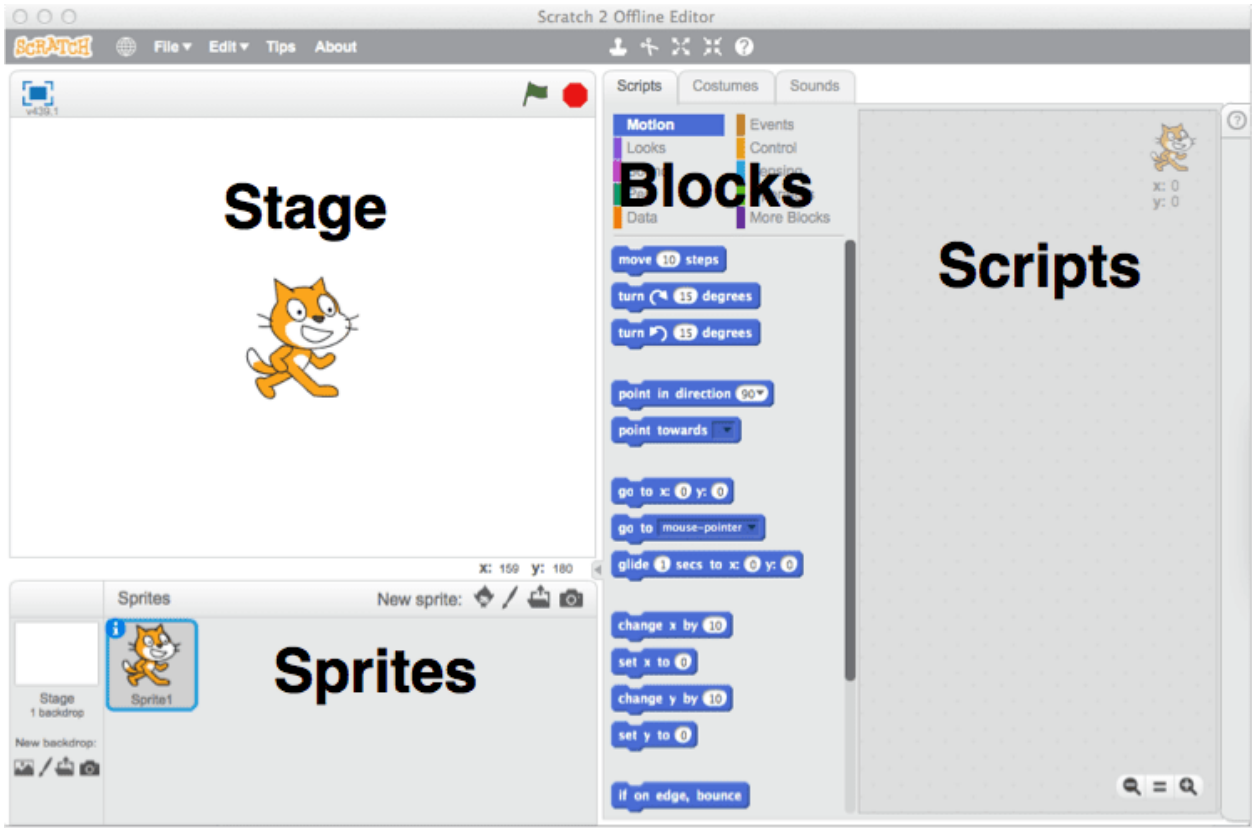
Start by watching the video below to learn how to open Scratch when using the Finch.
Use the USB cord to connect the Finch to the computer. In order for the Finch to run a program, this cord must always be attached to the robot and to the computer.
Then open the Birdbrain Robot Server (or the Finch Connection App, if you are using a Chromebook). This program must remain open the entire time that you are programming in Scratch. It is like a translator between Scratch and your Finch.

The BirdBrain Robot Server should tell you that your Finch is connected. Click Open Scratch.
NOTE: On a Mac, the USB cord for the Finch must be plugged in before you open the BirdBrain Robot Server.

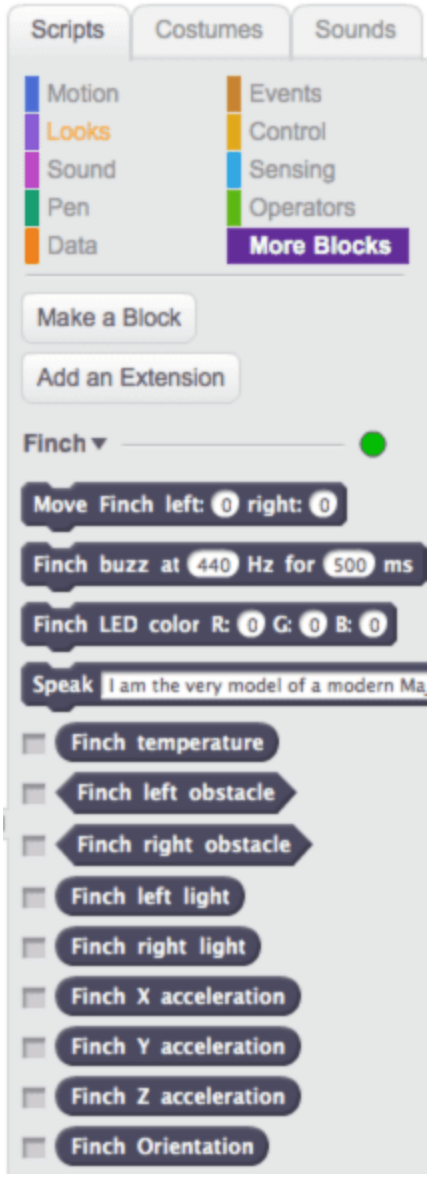


Programming in Scratch

This is the Scratch window. It has a few different parts. To write a program, you will drag blocks from the Blocks area to the Scripts area. In Scratch, programs are also called scripts. You can use Scratch to write scripts that use the Finch or scripts that use cartoon characters called sprites. This lesson will focus on the Finch.



The Blocks area in Scratch contains 10 different menus – Motion, Looks, Sound, etc. Click on the different menus to get an idea of the blocks that they contain. All the Finch blocks are located in the More Blocks menu in Scratch.

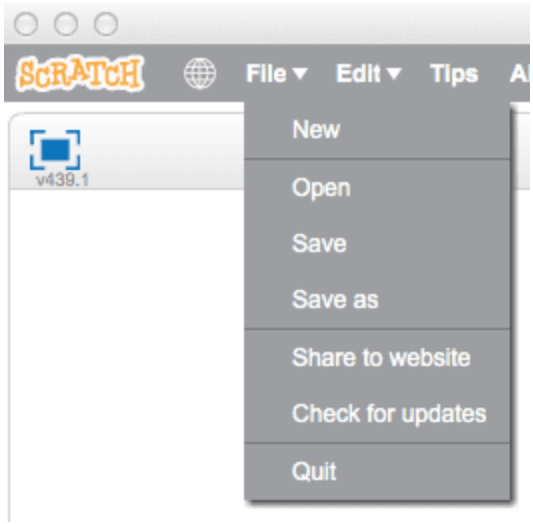


Saving Your Work

It is very important to save your work often! Otherwise, you might lose something important. Watch this video to learn how to save a program.

To save a new project in Scratch, click on File and then Save as (Save Project for Chromebooks using ScratchX). Give your project a name and then click Save. Once your project has a name, you can save it by going to File and then Save.

DO NOT name your Scratch file FinchStart. This will overwrite the blank file that should appear when you open Scratch from the Birdbrain Robot Server.



Moving the Finch

To move and turn the Finch, you will use the Move Finch block. The Finch has two motors, one for each wheel. The Move Finch block enables you to start these motors. The block requires two numbers. These numbers represent the speed of the left and right motors. Each number can be any whole number from -100 to 100.

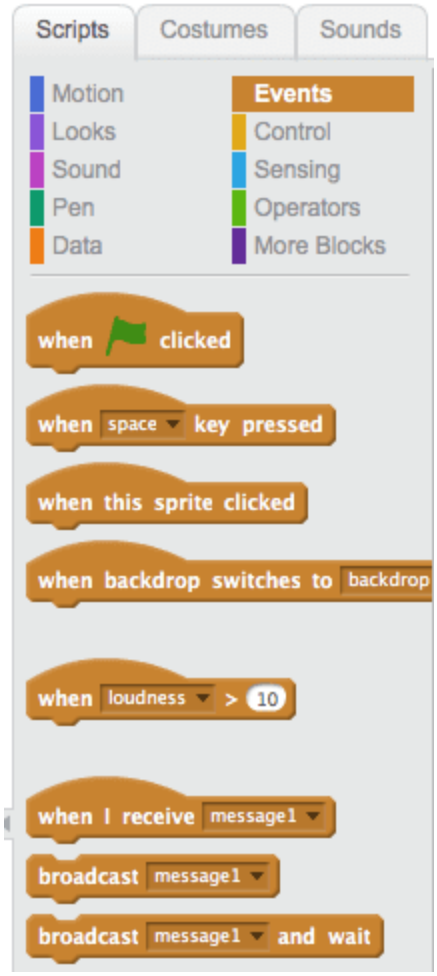
The Move Finch block makes the Finch start moving. Drag a Move Finch block into the Scripts area and set both number to 50. This is your first program! Click on this block to run your program. The Finch will start moving forward. You may need to pick it up to keep it from driving off the table.

Exercise 1:

Change each 50 to 0 and click the Move Finch block again. This should make the Finch stop moving.

Events

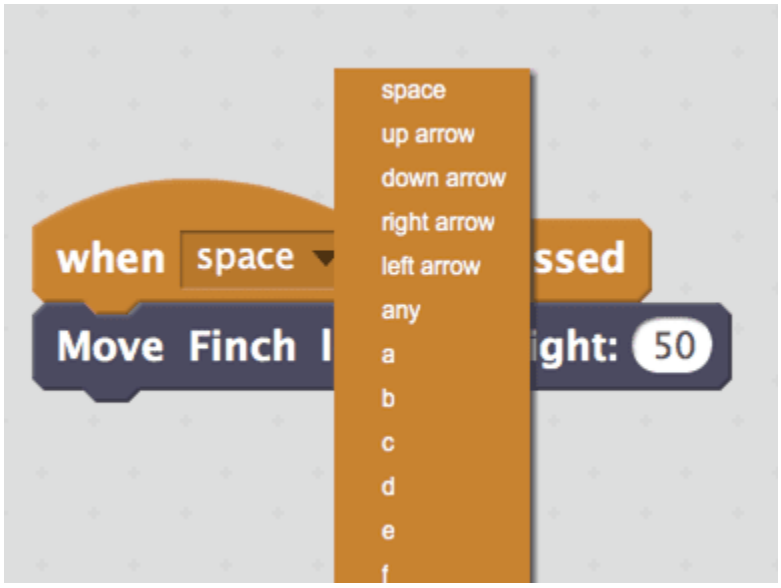
So far, you have been running your program by clicking on it. You can also use an event to start a program. An event is an action that the computer can recognize. For example, you might press a key on the keyboard. Click on the Events menu.



The second block on this menu is when key pressed. Drag this block into your program until it connects to the top of your script. Notice that the shape of this block shows you that it must be at the top of a script. It can't be connected below another block.

Now you can run your program by pressing the spacebar. Try using the spacebar to make the Finch move. You can click the stop sign in the upper left corner of the Stage to stop the Finch.

Click on the black triangle in the when key pressed block. A menu will pop up that will allow you to select other keys. Modify your program so that the Finch moves forward when you press ‘o.’

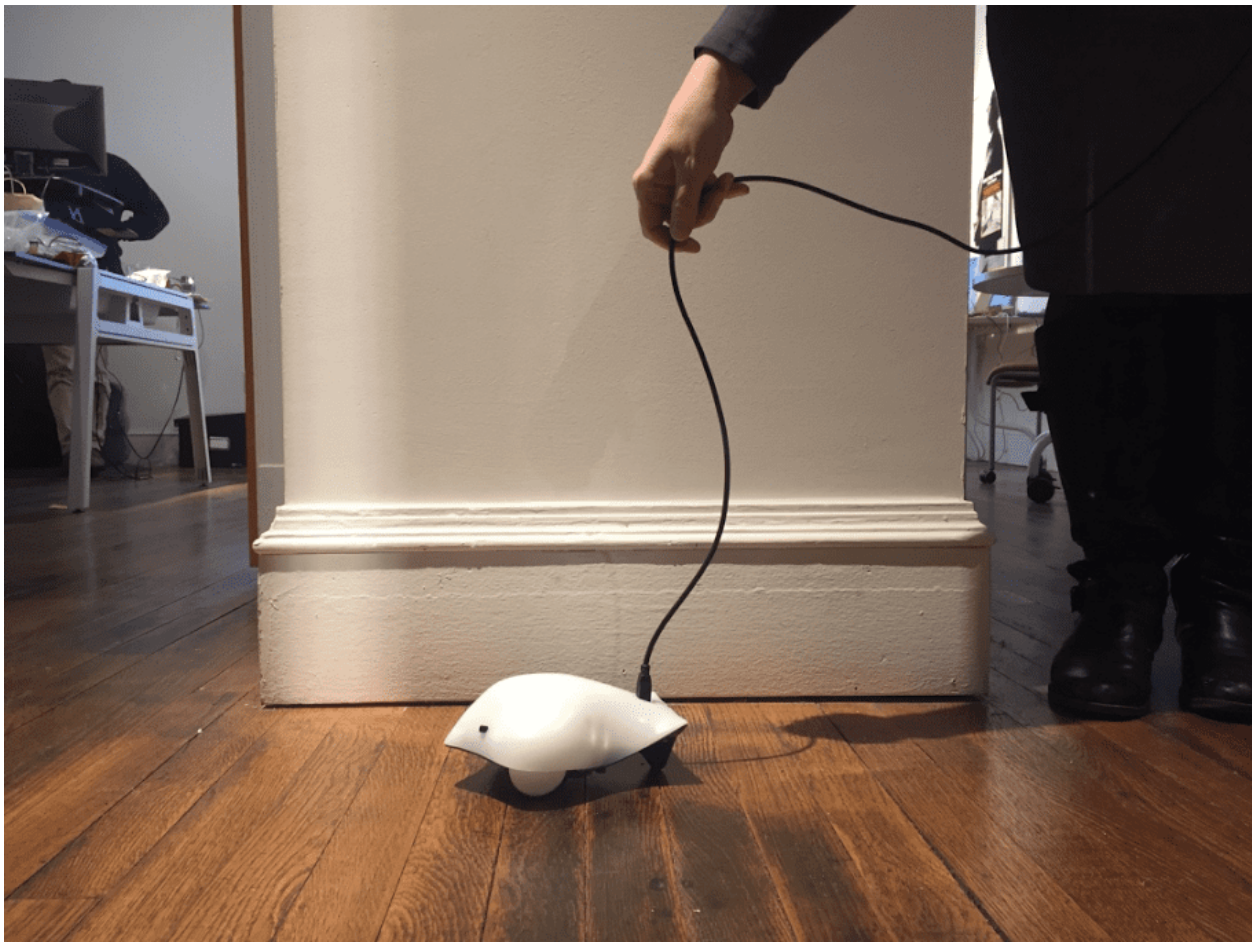


Exercise 2:

You may have already discovered that you can write multiple scripts in the Scripts area. Add a second script that stops the Finch when you press ‘x.’

Tip:

You may need to carry the USB cord as the Finch moves. Otherwise, the cord may keep the Finch from moving and turning freely.



Creating Longer Scripts

The wait block can be found on Control menu. This block pauses the program for the number of seconds shown in the block. This number can be a whole number or a decimal number.

You can use the wait block to move the Finch for a certain period of time. For example, this program will make the Finch drive forward for three seconds and then stop.

Exercise 3:

Try out the program below. Then try several motor speeds between 0 and 100 in the first Move Finch block (keep the speeds of the left and right motors the same). Describe two ways that you can change how far the Finch moves.

Exercise 4:

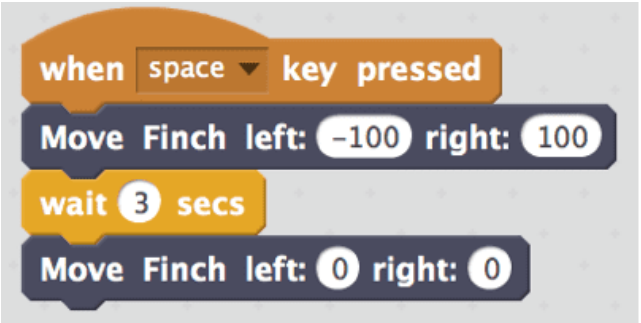
What does it mean for the speed to be negative? Try several motor speeds between 0 and -100 (keep the speeds of the left and right motors the same).

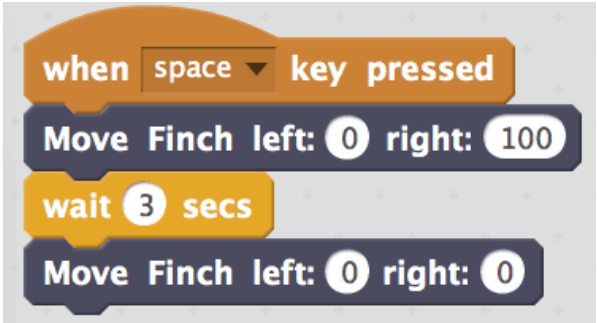
Turning the Finch

So far, the speeds of the left and right motors have been equal. When these speeds are equal, the robot moves in a straight line. When the speeds are not equal, the Finch will turn.

Exercise 5:

Try the two programs shown below. How are these two turns different? How can you make the robot turn in the other direction?



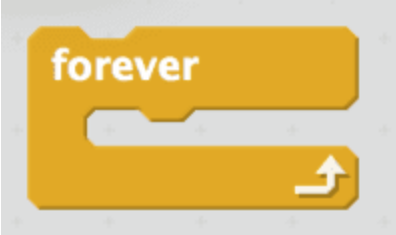


Exercise 6:

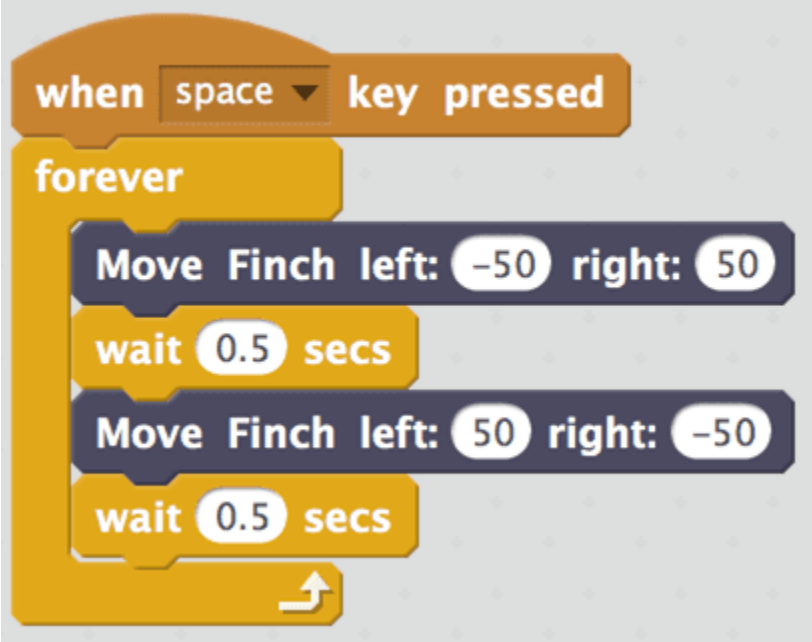
Write a program that makes the robot turn a full circle to the left and then a half circle to the right.

Loops

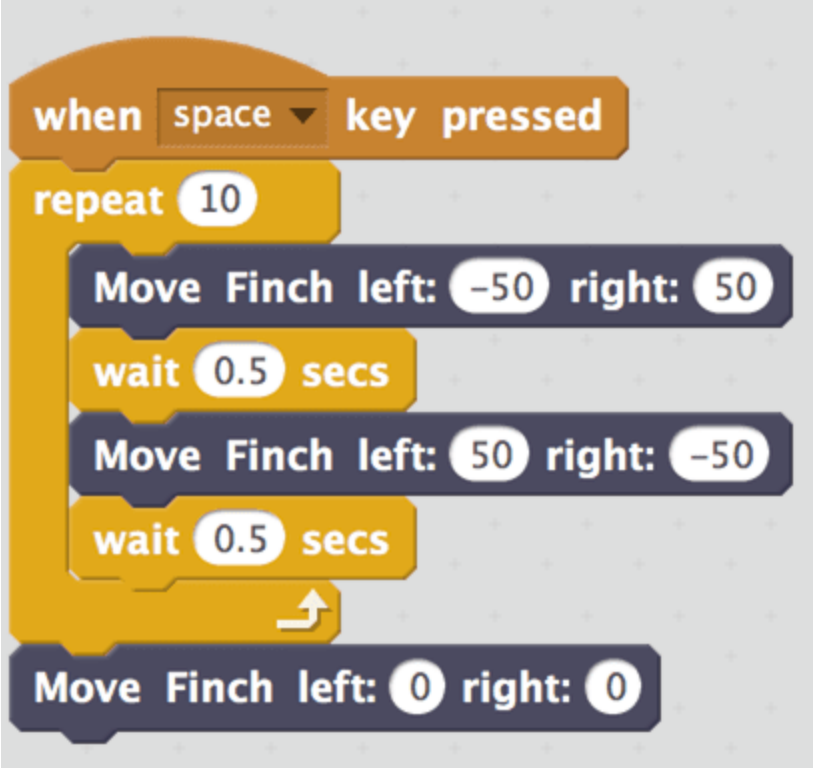
Think about how you might make the Finch turn back and forth repeatedly. One way to do this would be to use a long sequence of commands, but it is much simpler to use the forever block in the Control menu. This block is called a loop. A loop is a programming structure that repeats a portion of a program. Look at the Control menu. Which of the other blocks do you think might be loops?



Other blocks can be placed inside the forever block. The forever block repeats the blocks inside it until you press the stop sign to stop the program. This loop makes the Finch turn left and right. The loop repeats the four blocks inside it over and over. After the second wait block, the program immediately goes back to the first block inside the loop, the block that turns the Finch to the left. What happens if you remove the second wait block?



The repeat block is a loop that repeats the blocks inside it a certain number of times. For example, this program makes the Finch turn left and right ten times. To stop the Finch at the end of the script, you need to add a command below the repeat block.



Exercise 7:

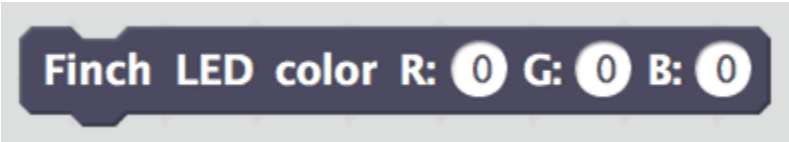
Use a loop to make the Finch drive in a square. What actions does the Finch need to repeat? How many times should it repeat them? Note: It is hard to turn precisely with the Finch; the angles on your square do not need to be perfect.

Programming Tip:

To make your Finch turn more reliably, you should program it to turn slowly (speed 20-40). Also, don't forget to hold the Finch's cord while it is moving!

Color with the Finch

You can use the Finch LED color block to change the color of the Finch's beak. The Finch's beak actually has three tiny light elements inside it. One is red, one is green, and one is blue. This is important for programming the beak. The Finch LED color block requires three numbers, which are labelled R, G, and B. R controls the amount of red light from 0 (none) to 100 (maximum brightness). G and B control the amount of green and blue light, respectively, from 0 to 100.



Exercise 8:

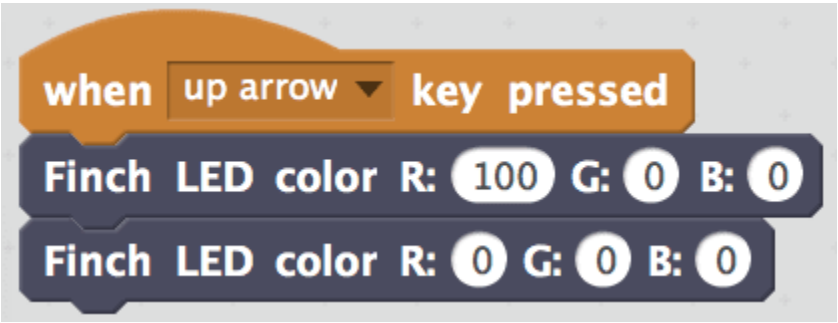
What do you think the program below will do? After you make a hypothesis, try it and find out.

Exercise 9:

Write a program to make the beak blink on and off repeatedly in your favorite color.

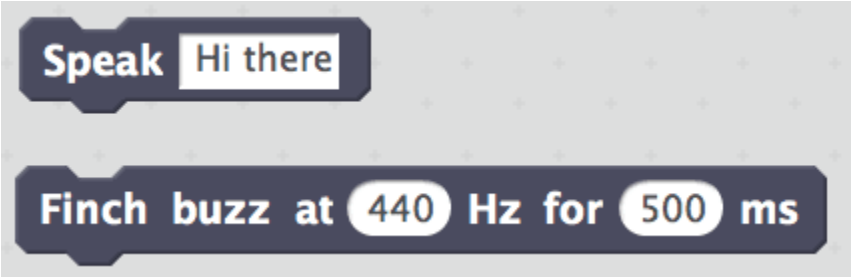
Programming Tip:

When Scratch reaches a Finch LED color block, it sets the beak and moves immediately to the next block. This means that if you do not have a wait block between two Finch LED color blocks, you may not see the effects of the first block. For example, you may not see the LED turn on when you run the script below. The same is true for the Move Finch blocks; a pair of Move Finch blocks should have a wait block between them.



Sound with the Finch

The More Blocks menu for the Finch contains two sound blocks. The Speak block will cause the computer to speak whatever text you type into the block.



The Finch buzz block activates the Finch's buzzer. This block requires two numbers. The number on the left (Hz) gives the frequency of the sound; keep in mind that humans can only hear sounds in the range of 20 to 20,000 Hz. The number on the right (ms) gives the length of the sound in milliseconds. This block will start the buzzer and then move on immediately to the next block.

Exercise 10:

Write a program that uses two Speak blocks. What block do you need to place between your Speak blocks?

Exercise 11:

Use the Finch buzz and wait blocks to make the Finch play a short song.

Conclusion

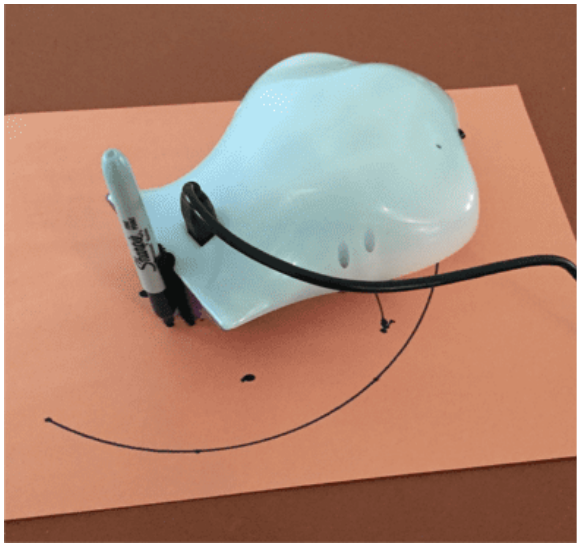
You have learned how to use all the Finch outputs! Watch this video to review before completing the last exercise in this lesson.

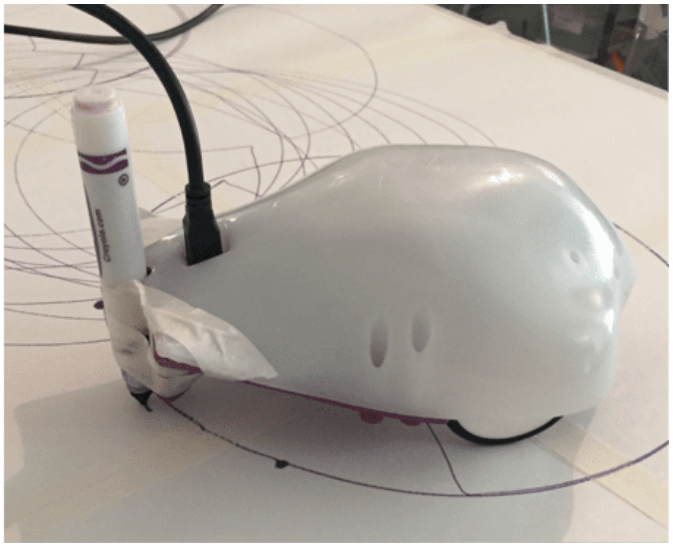
Exercise 12:

Practice all of the things you have learned in this lesson by making the Finch move in a geometric shape, such as a triangle or pentagon. The Finch should buzz as it starts each side of the shape. For an added challenge, make the beak change to a different random color for each side of the shape (Hint: Explore the pick random block in the Operators menu).

Tip:

Use the indentation in the Finch's tail to attach a marker with tape or velcro. Then the Finch will draw your shape! The corners of the shape will be rounded because the marker is attached to the Finch's tail, rather than at its turning point.





Sensing with the Finch

The future of Scratch 2.0 offline and ScratchX is uncertain.
We encourage you to use Snap! with your Finch Robot to ensure long-term support.

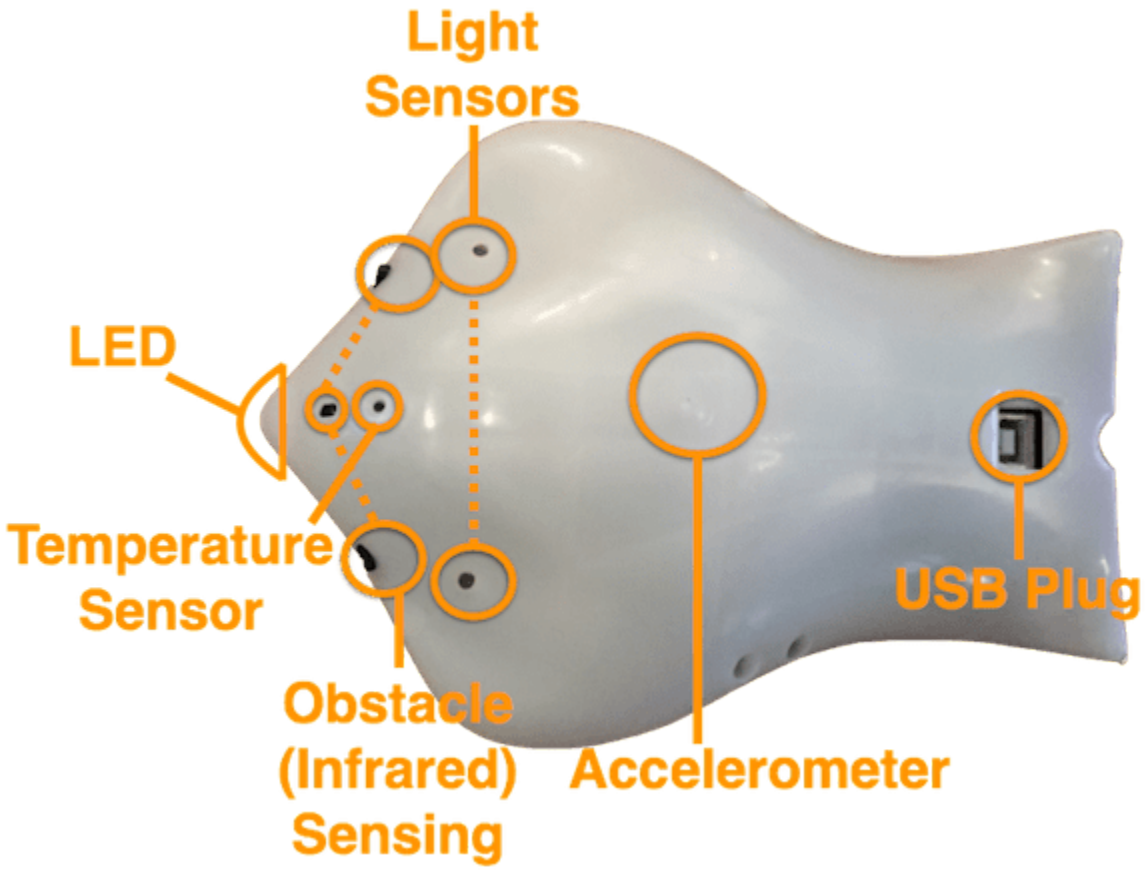
Lesson Key

Lesson key available for teachers. GET ACCESS

Table of Contents

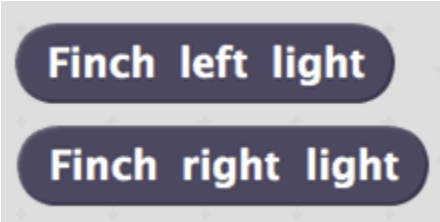
- Creating a Variable
- Following the Light
- Using Math with the Temperature Sensor
- Moving Sprites with the Accelerometer

In the first lesson, you learned to use the Finch outputs: the motors, lights, and buzzer on the Finch. The Finch also has sensors that provide input to the robot. A sensor makes measurements and sends them to the program that the Finch is running. The program can use the sensor information to set outputs or make a decision. The Finch has two light sensors, an acceleration sensor (accelerometer), two obstacle sensors, and a temperature sensor.



The sensor blocks for the Finch are at the bottom of the More Blocks menu in Scratch. For example, the Finch left light and the Finch right light blocks can be used to measure the values

of the two light sensors. Each of these blocks has a value from 0 (no light) to 100 (maximum light).

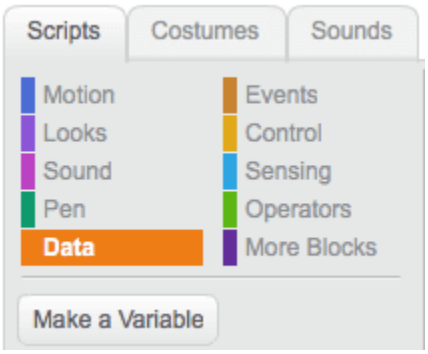


Exercise 1:

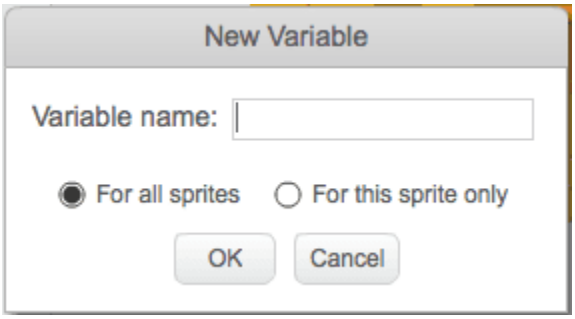
Drag a light sensor block into the Scripts area. Click on the block to see the current value of the sensor. Next, cover the light sensor with your hand and click on the block again.
You should see the value of the light sensor change. What is the value of the block when you shine a flashlight on the sensor?

Creating a Variable

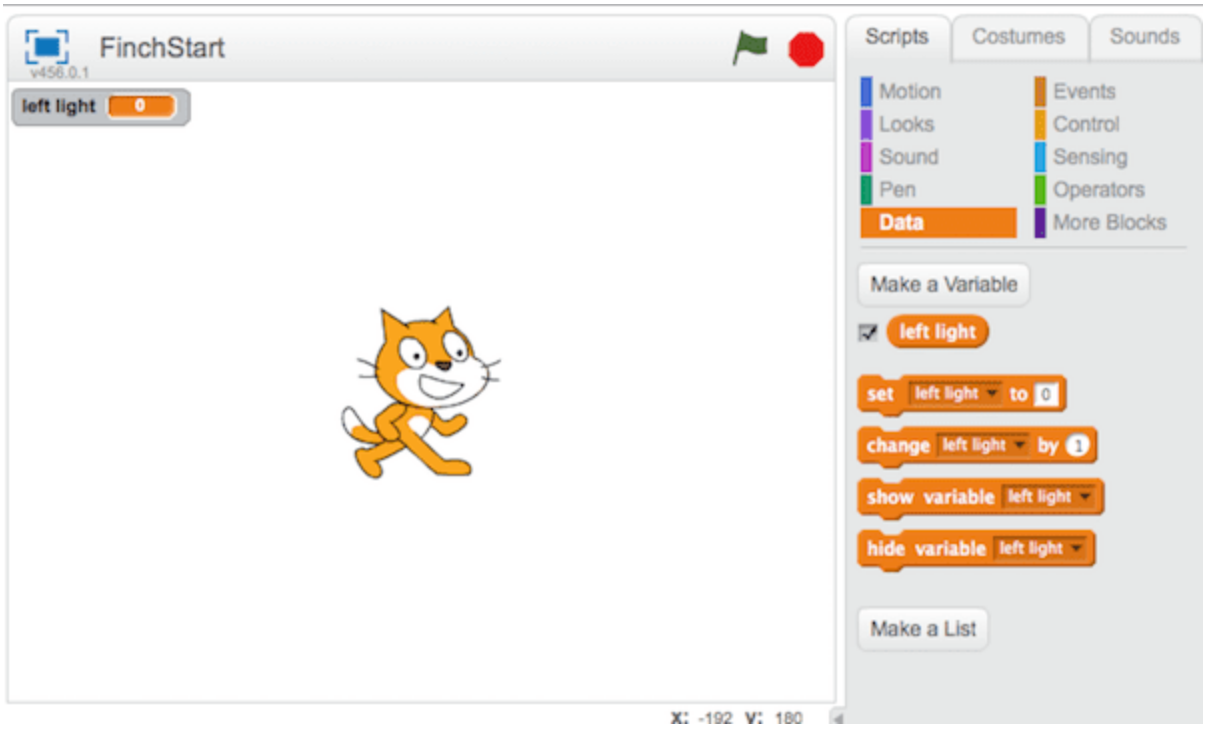
To see the value of a sensor change, it is helpful to create a variable. A variable is just a name that represents a value. You will now create a variable named “left light” that will hold the value of the left light sensor and display that value on the screen.
First, go to the Data menu. Then click Make a Variable.



The New Variable window will appear. Call the variable “left light” and click OK.



You will notice that the Data menu looks different now. It contains some blocks that you can use to change the value of your variable. Also, the value of the variable is shown in the top-left corner of the stage.



We want to set the value of our variable equal to the value of the left light sensor. To do this, we can use the set to block. We will place this block inside a forever loop to continually set the variable to the value of the sensor.

Exercise 2:

Run the script below and watch the value of the variable change as you cover and uncover the light sensor. Add a variable for the right light sensor and modify your program so that both variables change as you cover and uncover the Finch.

Following the Light

You can use the sensor blocks to make the robot respond to its environment. For example, you can make the robot follow a flashlight using the script shown below. The Move Finch block set

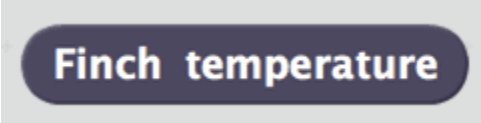
the speed of the robot using the values of the light sensors, and the forever loop repeats this action over and over.

Exercise 3:

Use a flashlight to try out the program shown below. Why is the speed of the left motor controlled by the right sensor? What happens when the speed of the motor is controlled by the sensor on the same side?

Using Math with the Temperature Sensor

There is also a Finch block that can be used for measuring temperature in Celsius.



Exercise 4:

Create a variable named “temperature” that is equal to the value of the temperature sensor. What is the temperature in your room? Can you raise the temperature reading of the sensor? But what if you want to know the temperature in Fahrenheit instead? You can use this formula, but you need to do some math in Scratch.

$$F = \frac{9}{5}C + 32$$

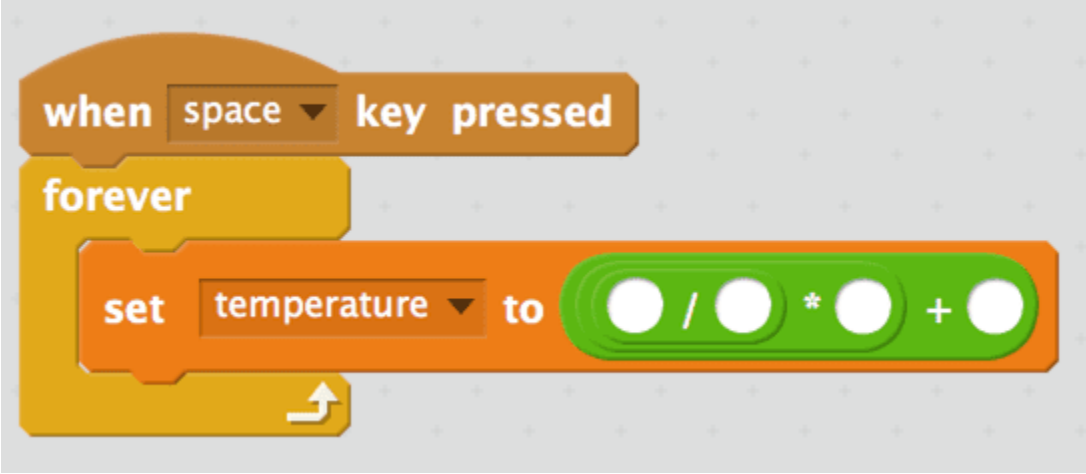
In the Operators menu, Scratch contains blocks that you can use to do arithmetic.

Exercise 5:

You can place arithmetic blocks inside one another. When you do this, the operation in the innermost block will be performed first. In the expression below, the 2 and 4 will be added together first, and then the result will be multiplied by 10. What will be value of this expression be? Make a hypothesis and then test it in Scratch. Remember, you can click on a block to find the value of the block.

Exercise 6:

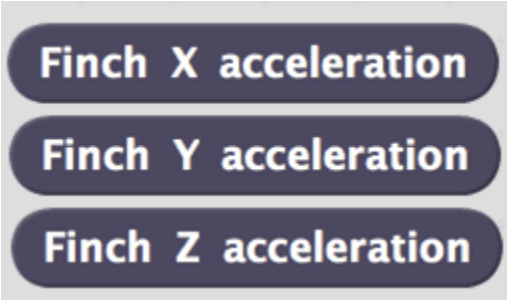
Fill in the blanks below to make your temperature variable hold the temperature in Fahrenheit measured by the Finch.



Moving Sprites with the Accelerometer

The last Finch blocks that we will use in this lesson measure acceleration. These blocks measure how much the Finch is tilted (or changing its speed). When the Finch is sitting on a flat surface, these blocks will not change. There are three different blocks because the Finch can

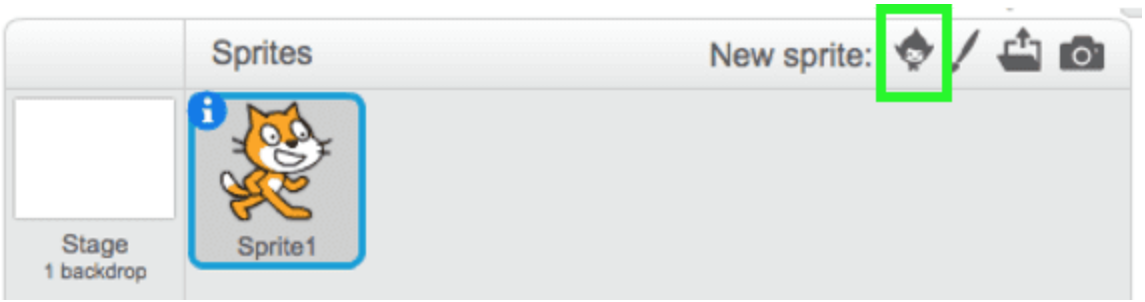
measure tilt in three different directions. The value of each block is a decimal number between -1.5 and 1.5.



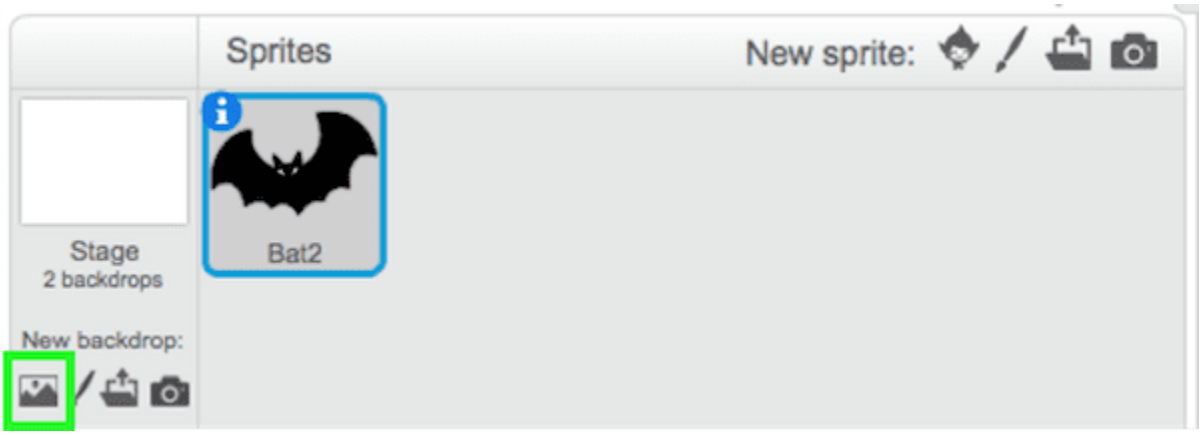
Exercise 7:

Create three variables. Use the program below to continually change these variables to the values of the acceleration blocks. Tilt the Finch in different directions. What type of tilt makes each variable change?

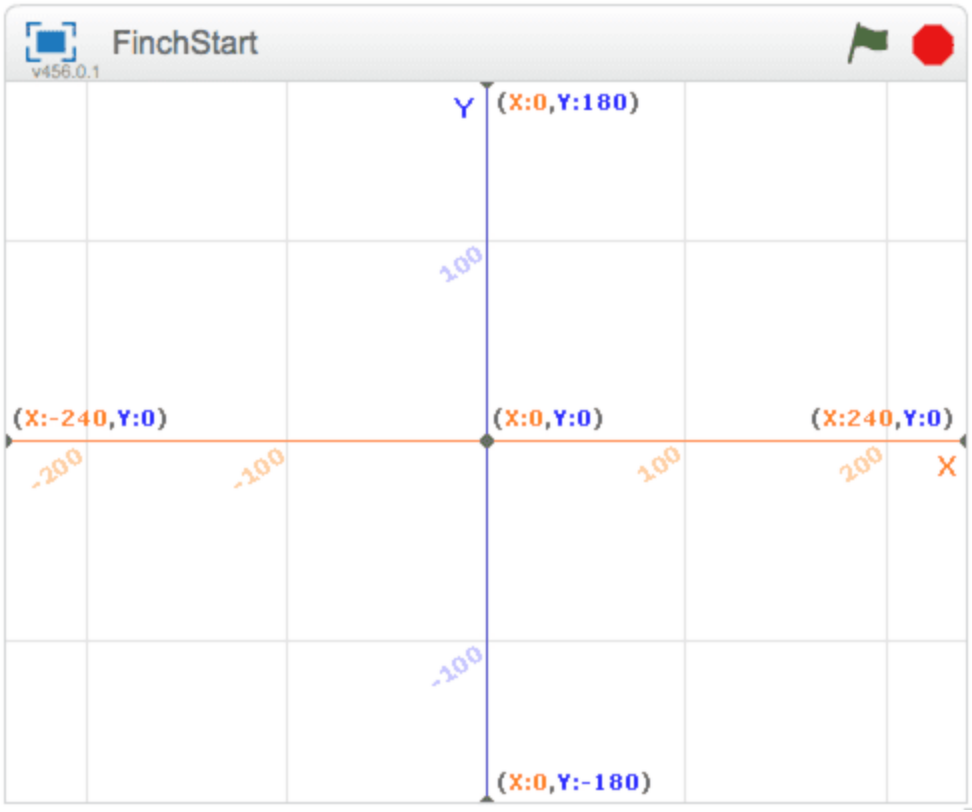
You can use the accelerometer to move a sprite around the stage in Scratch. This will enable you to make your own video games with the Finch as the controller!
First, click on the small picture beside the text “New Sprite.” This will open Scratch’s Sprite Library window. Select a sprite you like and click OK. If you want to get rid of the cat, right-click on it and select delete.



Sprite in Scratch move on a coordinate grid. You can see this grid by clicking the picture under “New backdrop:” by the Sprites area. All the way at the bottom is a background called “xy-grid.” Select it and click OK.



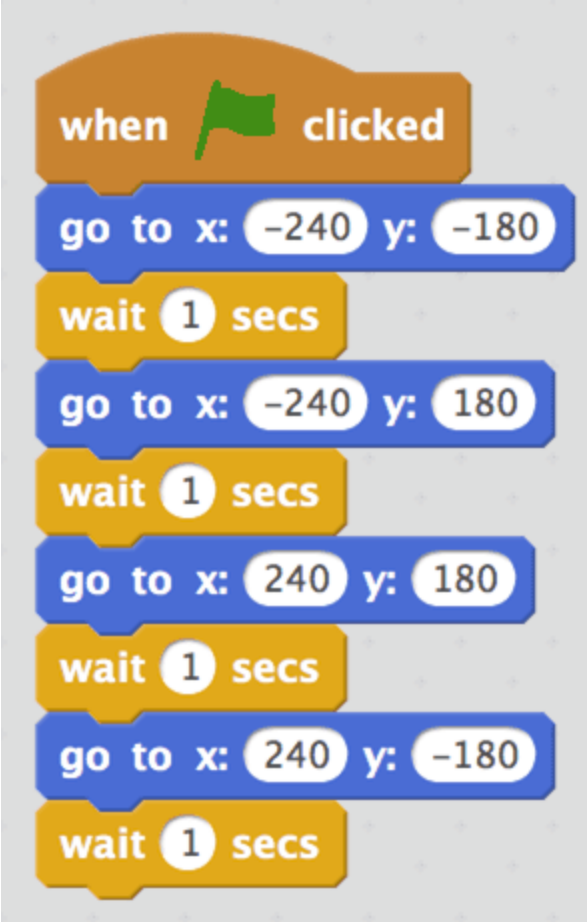
You should see a background that looks like this. The x -value of a sprite can be between -240 and 240 in Scratch, and the y -value can be between -180 and 180.



You can use the go to block in the Motion menu to send a sprite to a particular point.

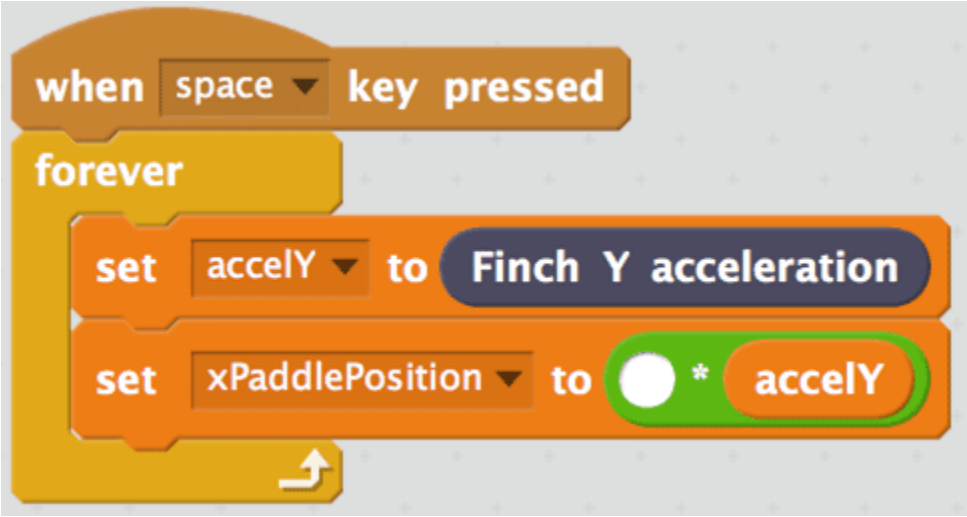
Exercise 8:

What will your sprite do when you run this program? Make a hypothesis, and then try it out.



Exercise 9:

Declare another variable named xPaddlePosition . You will use this variable to determine the x -position of the paddle on the stage. As you tilt the Finch, the value of the y -acceleration is between -1 and 1. What number will you need to put in the blank below to make the paddle move from -240 to 240?



Exercise 10:

Place a go to block at the bottom of your forever loop. Use this block to continually set the x -position of the paddle sprite to xPaddlePosition . The y -coordinate for the sprite can be -180; this is the bottom of the stage. When you run your script, you should be able to move your sprite by tilting the Hummingbird!

Exercise 11:

You will notice that when you tilt the Finch to the left, the paddle moves to the right. When you tilt the Finch to the right, the paddle moves to the left. How can you adjust your code to fix this problem?
Now that you can control a sprite with the Finch, you can add other elements to create your own video game! For more details, see the Finch Pong I activity; you have already done the first part of this activity.

Decisions with the Finch

The future of Scratch 2.0 offline and ScratchX is uncertain.
We encourage you to use Snap! with your Finch Robot to ensure long-term support.

Lesson Key

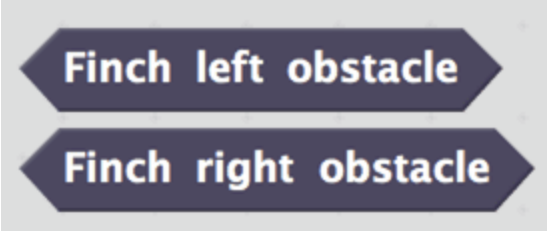
Lesson key available for teachers. GET ACCESS

Table of Contents

- Finding Obstacles
- Making Comparisons
- Orientation of the Finch

Finding Obstacles

In the last lesson, you were introduced to some of the sensor blocks for the Finch. In this lesson, you will learn to use those blocks to help the Finch make decisions about what it should do.
First, consider the Finch obstacle blocks, Finch left obstacle and Finch right obstacle. These blocks are Boolean blocks. That means that the value of the block is either true or false. If an object is in front of the right obstacle sensor, then the Finch right obstacle block is true. Otherwise, it is false. Each obstacle sensor can detect objects about 1-4 inches from the sensor.

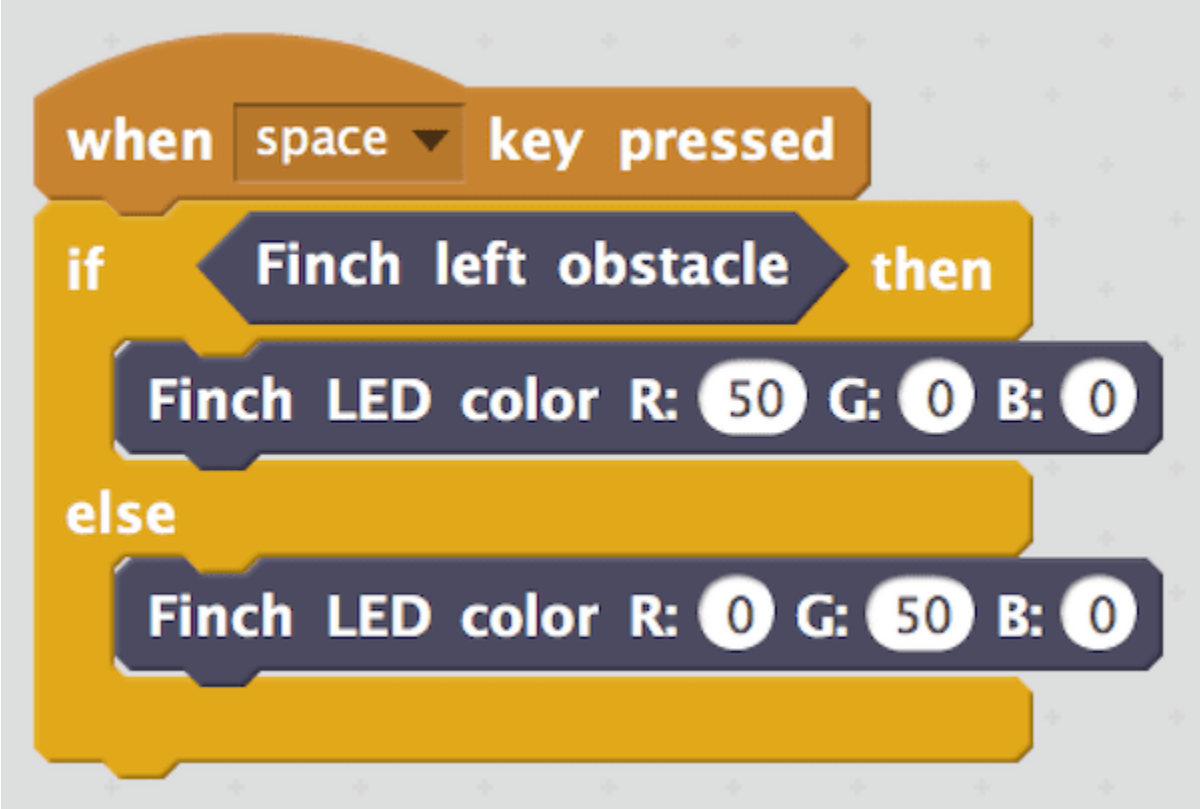


Exercise 1:

Drag both obstacle blocks into the Scripts area. Click on each block when nothing is near the Finch. Both blocks should be false. Then try placing a cardboard box close to the Finch, first on the right side and then on the left. Verify that each obstacle block is true when that sensor is about 1-4 inches from the box.

You can use the obstacle blocks in an if then else block. The if then else block in Scratch is a decision block. It is found on the Control menu.

The if then else block has spaces in the middle that can hold other Scratch blocks. In addition, the if then else block contains a hexagonal space. This space requires a Boolean block. If the Boolean block is true, the program runs the blocks in the top of the if then else. Otherwise, the program runs the blocks in the bottom of the if then else. For example, the program below turns the Finch's beak red when an object is close to the left sensor. Otherwise, the beak is green.



Exercise 2:

Try out the example script shown above. When you press the spacebar, it will check the left obstacle sensor and turn the beak red or green.

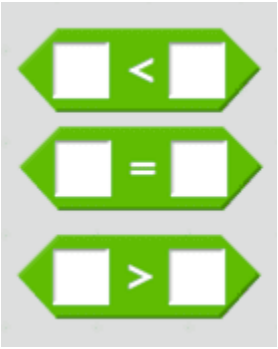
The script above makes a decision each time you press the space bar. You may also want to repeat this decision over and over. Then the beak will turn red whenever something gets too close to the sensor. To do this, place the if then else block inside a forever loop.

Exercise 3:

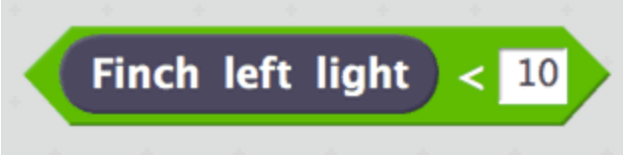
Write a program that will make the Finch turn right when the left obstacle sensor detects an object. Otherwise, the Finch should move straight ahead. Large, lightly-colored objects (like cardboard boxes) make the best obstacles.

Making Comparisons

The obstacle blocks are the only Finch blocks that are Boolean blocks. However, you can use the other sensor blocks to create Boolean blocks. You can do this using the three blocks shown below, which are found on the Operators menu. These are called comparison operators, or comparison blocks.



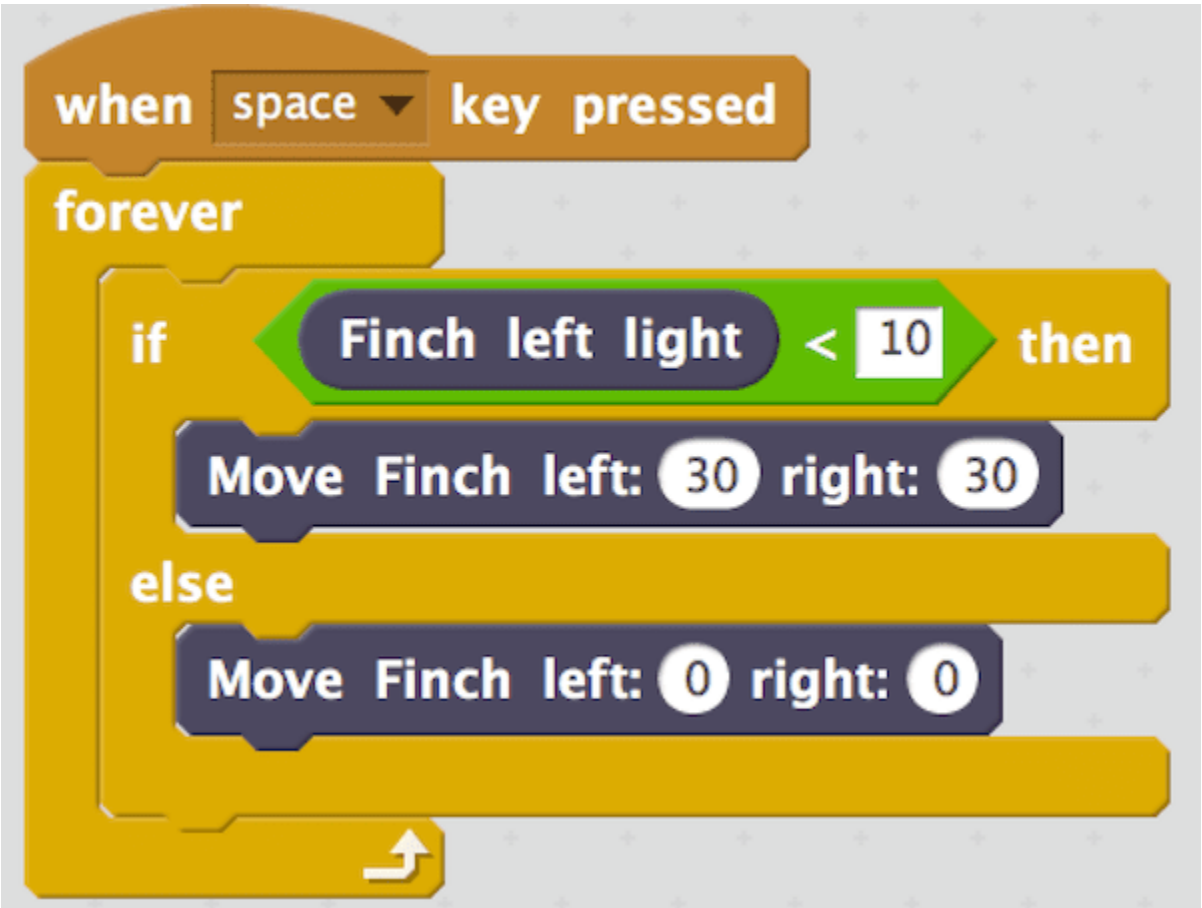
Create the Boolean block shown below. This block is true if the value of the left light sensor is less than 10, and false otherwise. The value that a Boolean block uses to make a decision is called the threshold. In this case, the threshold is 10.



Use the Finch left light block to measure the amount of light in your room. Then measure the value of the light sensor when you cover it with your hand. The average of these two values is a good threshold for the light sensor.

Exercise 4:

What do you think the script below will do? After you have made a hypothesis, test it out.



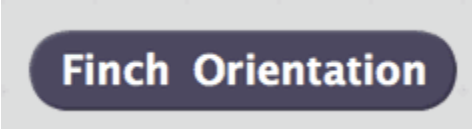
You should always use the < block or the > block with the Finch light, acceleration, and temperature blocks. The = block does not work well with these sensor blocks because the value of a sensor is rarely equal to a particular number. For example, the value of the light sensor might change from 52 to 49 to 51. A block checking whether the value is equal to 50 would be false for all of these values, but a block checking whether the value is greater than 50 would detect the changes.

Exercise 5:

Write a program that makes the Finch buzz when the user turns the Finch upside down. Otherwise, the Finch should do nothing. This program should use one of the following blocks: Finch X acceleration, Finch Y acceleration, or Finch Z acceleration.

Orientation of the Finch

In the last exercise, you had to use one of the acceleration blocks. There is also a Finch block called Finch Orientation. This block has one of seven values: Level, Beak_Up, Beak_Down, Left_Wing_Down, Right_Wing_Down, Upside_Down, and In_Between. Tilt the Finch in different directions and click on the Finch Orientation block to see the different values.



You can use the = block to check if the Finch Orientation block has a particular value. In fact, you must use the = block because the other blocks don't make sense. What would it mean to be greater than "Beak_Up"? The Boolean block below is true when the Finch's left wing is down and false otherwise. You must include the underscores for the code to work.

Exercise 6:

Modify your program from the last exercise to use a Finch Orientation block to detect when the Finch is upside down. What changes do you need to make?
So far, you have only used Boolean blocks inside if and if then else blocks. You can also use Boolean blocks inside a wait until block. This block stops the program until the Boolean statement in the wait until block is true. For example, the program below turns the Finch's beak red and then waits until the left obstacle sensor detects an object. Then it turns the beak off.

Exercise 7:

What will happen if you run this script when there is already an object next to the left obstacle sensor? Try it and find out!

Exercise 8:

Write a program that makes the robot move in a large circle until it finds an area that is dark. As you start to write programs that make decisions, you will notice that you need to think carefully before you begin to write a program. Planning a program carefully can save you from spending a lot of time debugging. Experienced programmers often draw a diagram (called a flowchart) of how a program will work or write out the steps that the robot will follow (this is called pseudocode).

More Decisions with the Finch

The future of Scratch 2.0 offline and ScratchX is uncertain.
We encourage you to use Snap! with your Finch Robot to ensure long-term support.

Lesson Key

Lesson key available for teachers. GET ACCESS

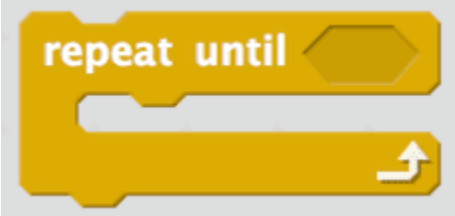
Table of Contents

- Repeat Until
- Logic Operators
- Nested Blocks

Repeat Until

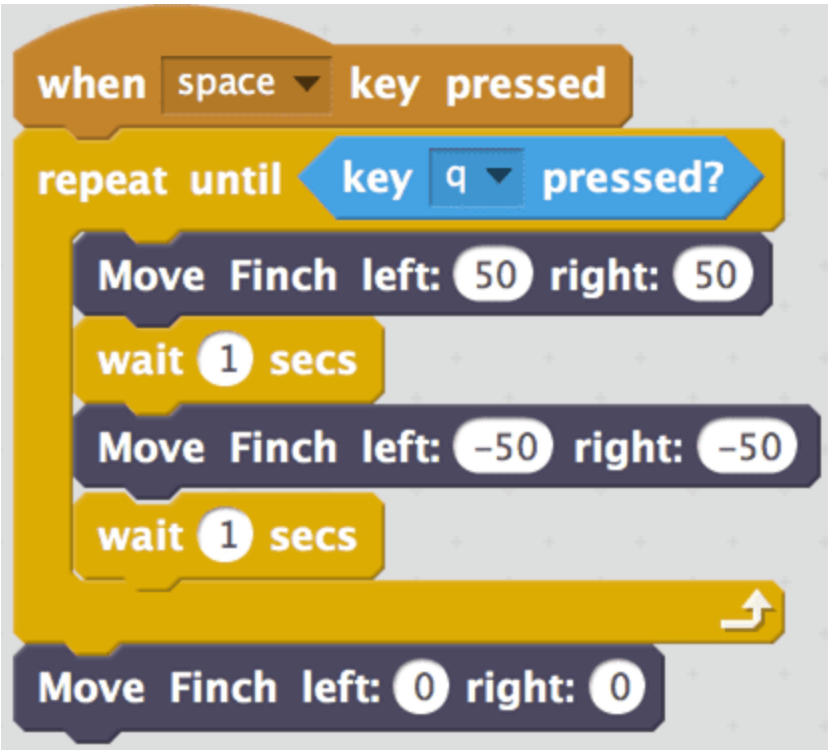
In the last lesson, you learned to create Boolean blocks with the Finch sensors. You used these blocks in the if, if else, and wait until blocks. In this lesson, you will learn to use one more decision block and to write programs that make more complicated decisions. As your programs become more complex, remember that you will need to spend time carefully planning what a program should do and how it should do it.

You may have used a repeat block to repeat a series of blocks some number of times. Scratch also includes the repeat until block. This block contains a space for a Boolean block. The actions inside the repeat until block will be repeated until the Boolean block is true. The repeat until block is a loop, just like the forever block. Each time through the loop, the program checks the Boolean block. If the block is false, the program repeats the blocks inside the loop. If the block is true, the program skips the blocks inside the loop and moves on to the next statement in the program.



Exercise 1:

The example program shown below will make the Finch move forward and back until you pressed the ‘q’ key. Then the Finch will stop. How is this program different from a program that uses a forever loop? What happens if you quickly press and release the ‘q’ key while the robot is moving forward? Explain why this occurs.

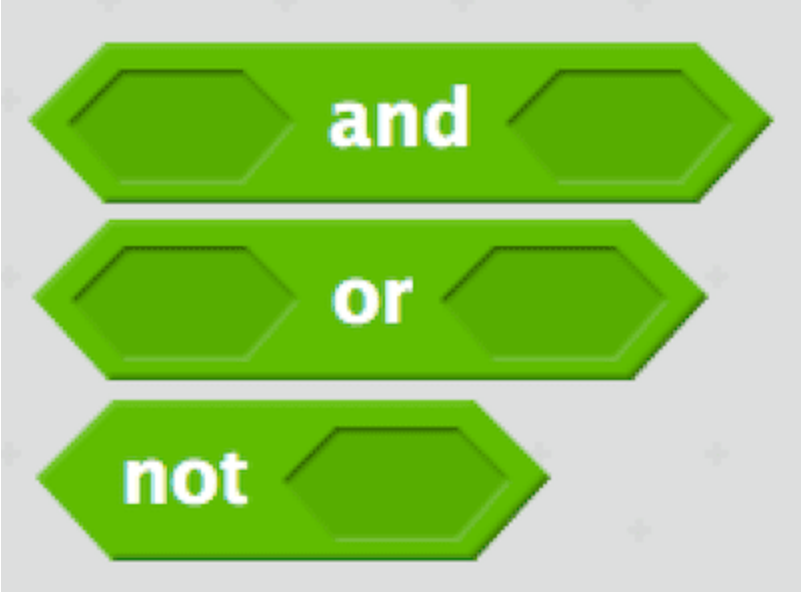


Exercise 2:

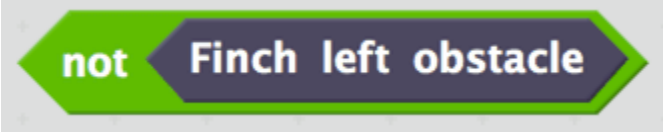
Modify one of your programs from the last lesson to use a repeat until loop instead of a forever loop. Describe one advantage of using a repeat until loop.

Logic Operators

So far, you have used comparison operators (>, <, and =) to create Boolean blocks. Sometimes, you want to make more complicated decisions. For example, if you want the robot to stop when it reaches an obstacle, you probably want to check both the right and the left obstacle sensors. You can use the logic operators and, or, and not to make complex decisions. Each of these operators is a Boolean block that contains space for one or two other Boolean blocks.



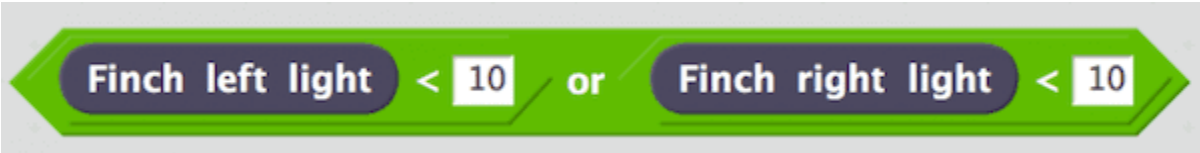
The not block transforms the value of a Boolean block into its opposite. For example, if Finch left obstacle is true, then the value of the block below is false.



Exercise 3:

Make the Finch beep while it is level. When the Finch is not level, it should stop beeping.
The and block is true only when both Boolean blocks inside it are true. If either Boolean block is false, then the and block is false. For example, the block below is true only if both light sensors detect that it is dark.

The or block is true if either of the Boolean blocks inside it is true. The or block is also true if both Boolean blocks inside it are true. The or block is only false if both Boolean blocks inside it are false. The block below is true if either light sensor detects that it is dark.



Exercise 4:

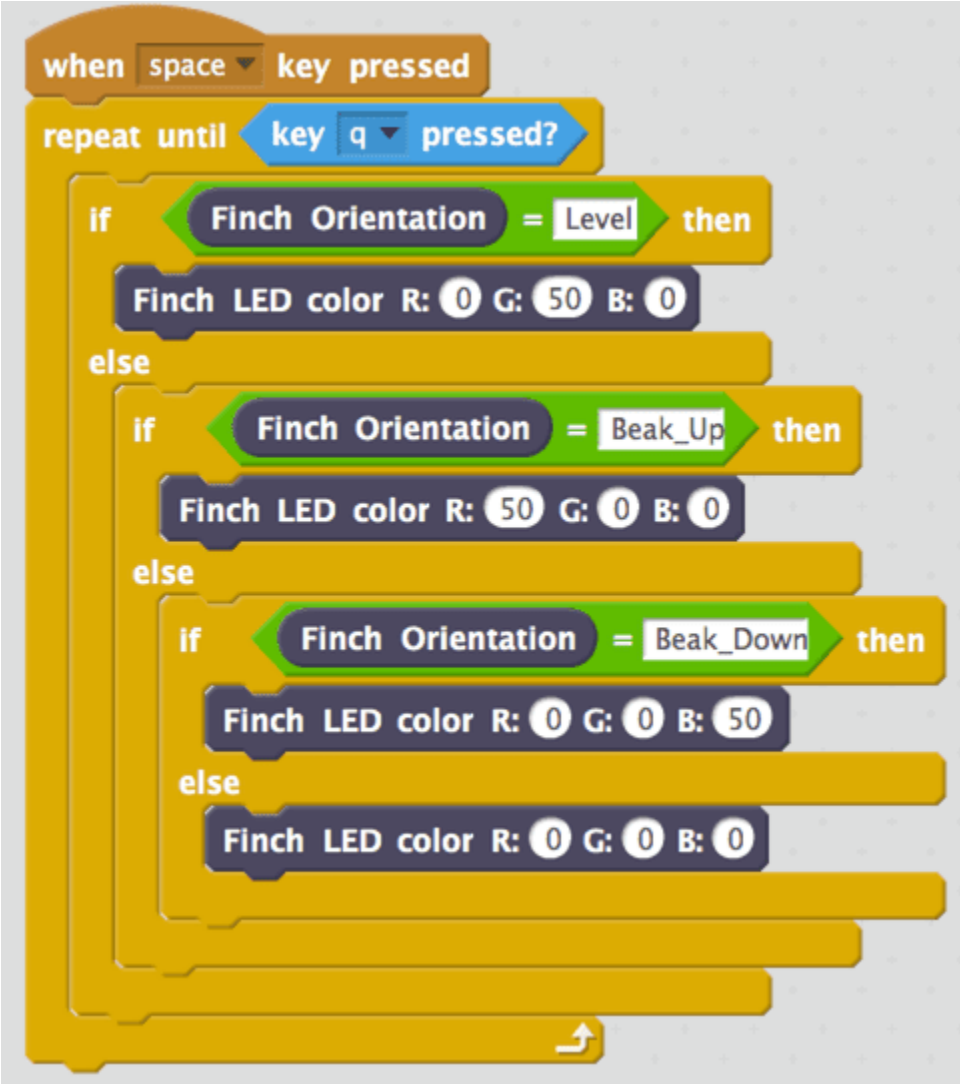
Write a program that makes the Finch move forward if there is nothing in its way. If the robot detects an obstacle, it should move backward. The robot should stop when you press the ‘q’ key.

Nested Blocks

The logic operators can be used to program the Finch for more complex behaviors. You have also created complex behaviors by placing if else blocks inside loops. This is called nesting. You can take nesting even further.

Exercise 5:

Look at the code below, and make a hypothesis about what the robot will do when this program runs. Then run the code to test your hypothesis. What other situations might require several nested if else blocks?



Lists with the Finch

[The future of Scratch 2.0 offline and ScratchX is uncertain.]

[We encourage you to use Snap! with your Finch Robot to ensure long-term support.]

Lesson Key

Lesson key available for teachers. GET ACCESS

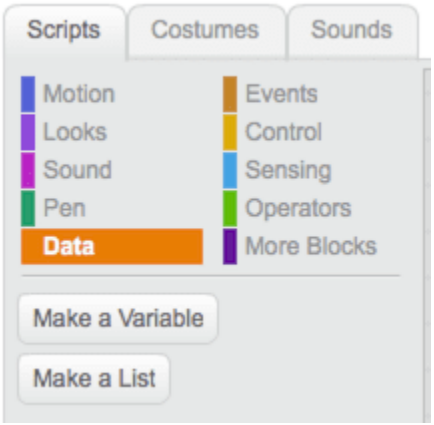
Table of Contents

- Creating a List
- Using Items in a List

In the previous lessons, you have used the Finch sensors to make measurements, and you have stored these measurements in variables. In this lesson, you will learn to use a new type of variable called a list. A list can hold many values, instead of just one.

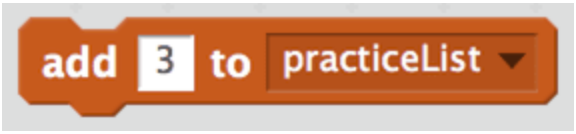
Creating a List

To create a list in Scratch, go to the Data menu and click Make a List. For now, name your variable practiceList. The default option For all sprites is fine.



At the beginning of a program, you need to make sure that the list is empty. To do this, use the delete of block on the Data menu. Click on the black triangle to select “all,” and make sure that practiceList is selected.

To add items to a list, you can use the add to block in the Data menu. For example, the block below adds the number 3 to practiceList. Try adding a few other values to practiceList. Notice that each value is added to the bottom of the list. By default, the list is shown in the stage area, although you can deselect the list to hide it.



You can also add items to the beginning of the list using the insert at of block. For example, the block below inserts the number 2 at the top of practiceList.

You can use the repeat block (Control menu) to add a large number of items to a list. The repeat block is a loop that repeats the blocks inside it a certain number of times. For example, the loop below adds five accelerometer measurements to practiceList.

Exercise 1:

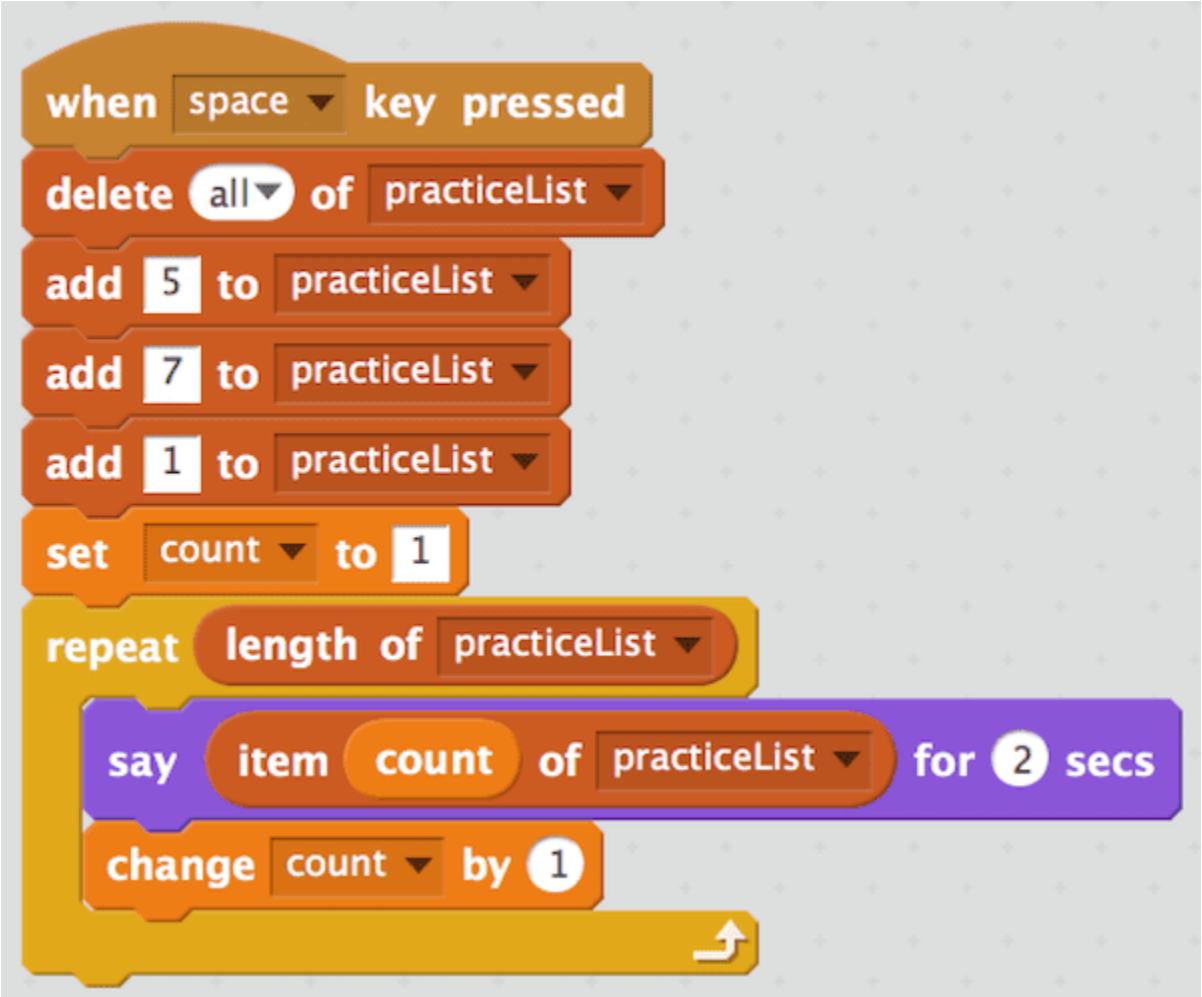
Write a program that records 60 measurements with the light sensor. The program should wait 0.5 seconds between measurements. Be sure to save this code, because you will need it later.

Using Items in a List

Each item in the list is assigned a number from 1 to the number of items in the list. You can access an item in the list using the item of block. For example, the block below will display the first item of practiceList.

Often, you will want to use a loop to move through all of the items in a list. You can do this using a counter variable and a repeat block. An example is shown below. At the beginning of the program, practiceList is set to contain three items. Before the repeat block, the variable count is

set to 1, corresponding to the first item in the list. In order to repeat the contents of the loop for each item in the list, the number of repetitions in the repeat block is set to the length of the list using the length of block (Data menu). Each time through the loop, the item in the list that corresponds to count is displayed, and then the value of count is increased by 1. Try out this script to make sure you understand how it works.



Exercise 2:

Return to your program from the last exercise. After you collect the accelerometer measurements, use a loop to move through the list. Use each item in the list to determine the position of the sprite on the screen. Remember, you may need to scale the accelerometer value so that you can see the sprite move.

Exercise 3:

Search through your list to find the largest value.

Exercise 4:

Finally, modify your program so that it calculates the sum of the items in your list in a variable called sum. Then calculate the mean acceleration.