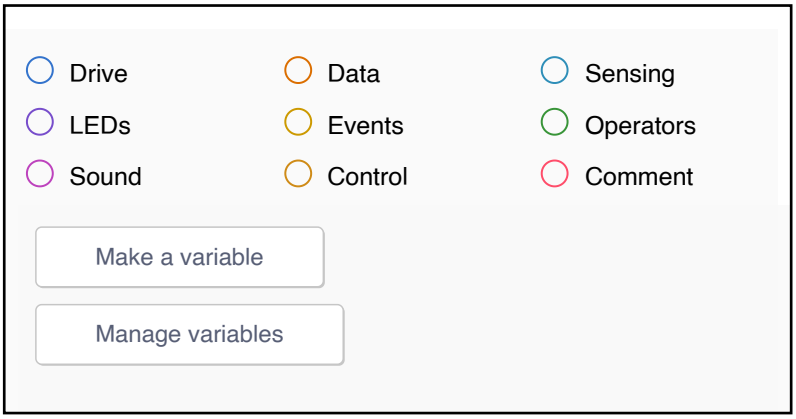
Name _____

Let's explore variables

In EdScratch, there is one category of blocks that looks very different to the other categories when you first open it: the **Data** category.



When you first click on the **Data** category, there are no blocks available for you to use in an EdScratch program. Instead, there are two buttons: the **Make a variable** button and the **Manage variables** button. By using the **Data** category, we can create **variables** to use in our EdScratch programs.

Jargon buster

A **variable** is a bit of memory that is used to store a value in a program. You can think of a variable like a container that you can use to store some bit of information in a way that will make sense to a computer.

In computer programming, we often use the same bit of information multiple times in a single program. Variables make it a lot easier to do this. Using variables in programs lets us tell the computer to store a specific bit of information inside a variable. We can then use that variable in different places in the program. Any time the computer sees the variable, it will recall whatever information is stored inside the variable.

In EdScratch, when you click the **Make a variable** button, a pop-up window will open up asking you to give your variable a name. Giving variables good names is important: you want the name of your variable to make sense to you and tell you what bit of information is stored inside.

Why is that?

Variable names need to make sense to you and to the computer. Computer languages often have rules about what characters can be used in variable names. The computer can only understand variables with names that follow these rules.

Your EdScratch variable names can only contain lowercase English letters, uppercase English letters, numbers, and underscores (_). Other symbols, like exclamation marks (!) or spaces, are not allowed.

Give your variable a name that will help you identify what type of information is going to be stored inside the variable. If you want to change the name of a variable after you make it, you can do that using the **Manage variables** button.

Once you make and name a variable, it will appear in the **Data** category along with some other special blocks including the **set** block, the **increment** block and the **decrement** block. You can then use these blocks along with your variable in an EdScratch program.

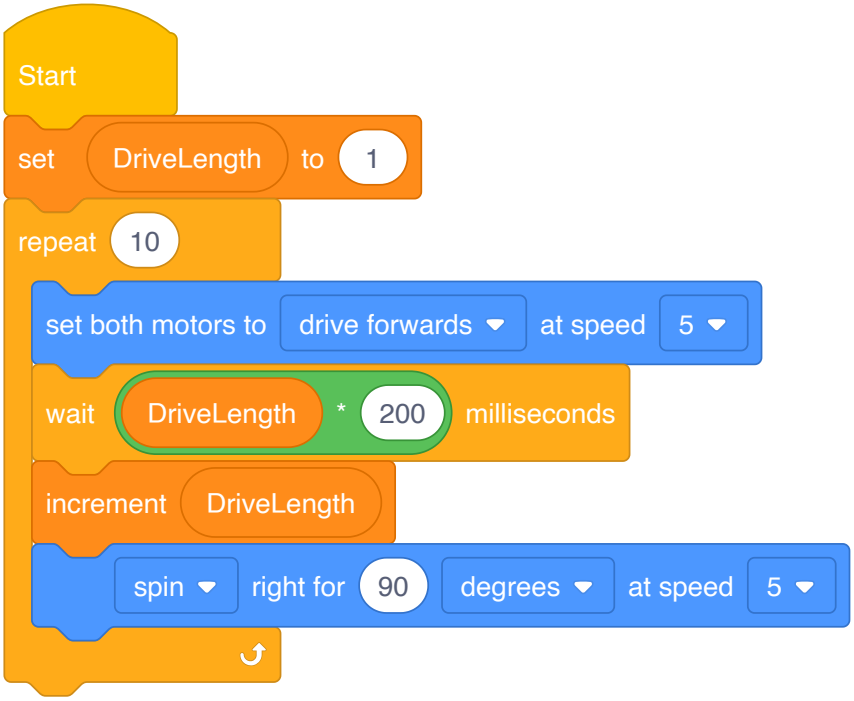
Jargon buster

In computer programming**, increment** means to increase by 1 and **decrement** means to decrease by 1.

Trace the code

In EdScratch, we can use variables to store different values. Because these values are numbers, we can then do different things with these numbers, like compare them to other values or do computations with them.

Look at the following program:



This program uses a variable named **DriveLength.**

It is important to note that a variable represents a value that is set somewhere in your program. That's why you always need to use code in your program to tell the computer what to set the variable to be.

This is what the **set** block in this program is doing. At the beginning of the program, the **set** block sets the variable **DriveLength** to be 1. The value of **DriveLength** isn't going to stay set to 1 forever, however. That's because this program uses another block from the **Data** category, the **increment** block:

The **increment** block is inside the **repeat** loop and changes the variable **DriveLength** to a new value. What is this block changing the variable **DriveLength** to be? That will depend on where in the program you are up to. In other words, it will depend on how many times the **repeat** loop has run.

Why is that?

One of the main reasons we use variables in programs is that a variable can hold a piece of information, even when the value of that information changes. This might sound really confusing but think about it like this:

Let's pretend you have a variable called **YearsOld** and the bit of information that this variable holds is your age. At your first birthday, **YearsOld** was set to 1. After that, each year on your birthday, **YearsOld** is **incremented,** which changes it to a new value: (**YearsOld** + 1).

A year after your first birthday, **YearsOld** is incremented. Because the value of **YearsOld** was 1, and since 1 + 1 = 2, the new value of **YearsOld** became 2. This repeats again on your next birthday, where the value of **YearsOld** is incremented and becomes 3.

The value of **YearsOld** changes each birthday, but the type of information doesn't change. **YearsOld** is your age, no matter how many birthdays you have!

When our program first starts, DriveLength will be set to 1, but because there is an increment block inside the looped code, the value will change each loop.

DriveLength is also being used in another place inside the looped code in this program:

The input value of this **wait** block will depend on the value of **DriveLength** and will also change each loop. Be careful though!

While the value of the **wait** block will change depending on the value of **DriveLength,** this **wait** block will not change the value of **DriveLength.** Only a block from the **Data** category can change the value of a variable.

Why is that?

Whenever you want to use a variable or a block from the **Operators** category as an input in a **wait** block, you need to use this special **millisecond wait** block. Edison actually 'thinks' in milliseconds, not seconds, so using this block makes it possible to do computations which Edison will be able to understand.

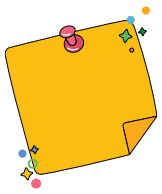
You might also notice that this wait block is set to milliseconds, not seconds.

Remember, **1 second = 1000 milliseconds.**

Let's trace the program to work out what the value of the variable **DriveLength** and the input value of the **wait** block are going to be at different points when the code in this program runs.

Trace through the program to work out the values. Fill out the table with what the starting value of variable **DriveLength** will be at the beginning of each loop repetition, what the input value of the **wait** block will be in that loop, and what the new value of **DriveLength** will be after the **increment** block inside the loop runs. The first two rows have already been filled in for you.

In loop #	Starting value of DriveLength	Wait block input value (in milliseconds)	New value of DriveLength
1	1	200	2
2	2	400	3
3			
5			
7			
10			



Don't forget

Tracing code means working through a program line by line, recording important values.

Name _____

Write and run the program

What is the program going to make the robot do when you run it in Edison?
Write the program in EdScratch. Download it and run it in your Edison robot to see what it does. Then answer the questions.

What does Edison do when you run this program? What 'shape' does the robot drive in?

Look at the code in the program. Explain why the robot drives in the pattern you see when you run the program in Edison. What in the code makes the robot move in that pattern?

Challenge up: Spiralling spider trap

Some spiders build webs which spiral into a central point in the middle. Can you program Edison to drive so that the robot spirals inward, like a spider laying a trap?

What to do

Write a program in EdScratch that will get Edison to drive in an inwards-spiral shape. Edison should drive, then turn, then repeat over and over, spiralling inwards.

Your program should use a variable to help you control how far Edison drives each time.

Think about how far you want Edison to drive each section compared to the previous section. You will also need to decide how far Edison should turn between each driving section.

Download your program and test it using your Edison robot. Experiment using different input values to see what works best.

Hint

You might want to look at the previous program 'Let's explore variables' for some inspiration to help you write your program.

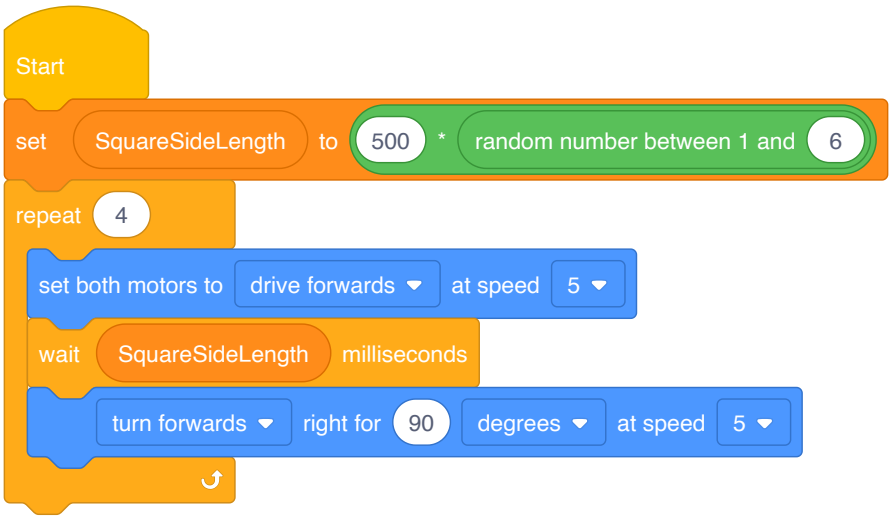
On the next page write your program in the text box or add a photo of your program in the image box.

Name _____

Name _____

Change it up: Drive a random square

Using variables lets us make programs that can do lots of interesting things.
Look at this program:



When Edison runs this program, the robot will drive in a square. But how big will that square be?

What to do

Write the program in EdScratch, then download it and run it with your Edison robot. Watch Edison drive the square. Now run the program again. Edison will again drive in a square, but chances are, this square will be a different size than the first!

Using the **random number** block inside the **set** block like this means that the value of the variable will change each time the program runs.

Each time this program runs, the variable **SquareSideLength** will be set to one of 6 different values. Fill out the table with the different values that **SquareSideLength** will be set to, depending on which random number is selected.

If the random number is:	SquareSideLength will be:
1	
2	
3	
4	
5	
6	

Mini challenge!

How can you tell which random number was used in the program? Think about how you can modify the random square program so that the robot will drive a random square, but also signal somehow to let you know which value was used in **SquareSideLength.**

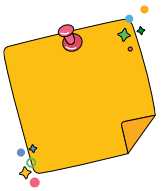
Modify the program and test it out in your Edison robot. Did it work?

Name _____

Let's explore using variables with sensor data

One of the most interesting ways we can use variables in EdScratch is to store and use data from Edison's sensors. Because all of Edison's sensors send back data to Edison as values, you can store a value from a sensor in a variable. You can also use sensor data to affect a variable, for example, by incrementing the variable every time a sensor detects something. By using variables and sensors together in a program, we can get Edison to react to sensor data in different ways.

Let's try using a variable with Edison's line tracking sensor to program Edison to drive until the robot detects four black lines.



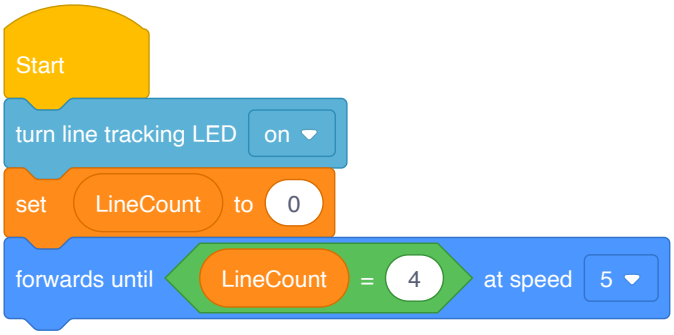
Don't forget

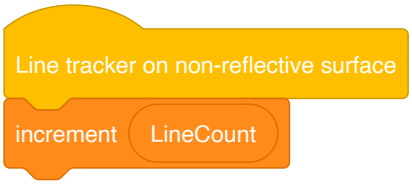
A **variable** is a bit of memory that is used to store some bit of information in a program. .

What to do

Using Edison's line tracking sensor, you can make a program telling Edison to drive until it detects a black line. In this activity, you need to get the robot to drive over multiple black lines, only stopping once it detects the fourth black line.

Look at the following program:





What's going on in this program? When this program is downloaded to Edison, what will it tell the robot to do?

This program has a variable called **LineCount** in it. Why do you think the programmer chose to name the variable that?

Try it out

Write the program in EdScratch, then download it to your Edison robot. Test the program using the activity sheet on page 163, or make your own test space by marking out four parallel black lines on a white surface, such as a large poster, table or the floor. You can put the lines as far apart from each other as you like.

Name _____

Using one of Edison's sensors with a variable lets us make all sorts of programs, using Edison in different ways. What else could you get Edison to do using variables and sensor data? Come up with at least one idea for a program that uses a variable in combination with sensor data. Describe your idea.

Can your idea be coded? Try to write your program in EdScratch. How did it go? Were you able to successfully write your program and test it using Edison? If so, did it work? Write about the process you took turning your idea into a program.

Challenge up: Edison the sprinter

The 400-metre dash is a sprinting event in many track and field competitions, including the Summer Olympics. This event is one of the longest 'sprint' events and that makes it different from other, shorter events. Most athletes who run the 400-metre dash know that they cannot sprint all-out with a 100% effort from start to finish. Instead, they break the race into different phases in order to balance endurance with speed.

We can program Edison to work like a 400-meter specialist, changing speed as the robot progresses through a course.

What to do

Write a program that gets Edison to behave like a 400-meter specialist, increasing speed as the robot passes markers (black lines) on a course. Your program should use Edison's line tracking sensor and a variable to help count how many lines the robot has encountered.

Test your program using the activity sheet on page 163. For this activity, the fourth line is the finish line, so you need to be sure the robot crosses this line, rather than just stopping once it detects that line.

You can also make your own test space by marking out four parallel black lines on a white surface, such as a large poster, table or the floor. You can put the lines as far apart from each other as you like.

Hint

You might want to look at the previous program 'Let's explore using variables with sensor data' for some inspiration to help you write your program.

Write your program in the text box below or add a photo of the program to the image box.