Name _____

Let's explore stacking and nesting if statements

Conditionals are powerful code elements in any computer language, including EdScratch. Using conditionals like 'if' statements in EdScratch lets you write all types of interesting programs for your Edison robot.

All the conditional blocks, including both the **if** block and the **if-else** block, are in the Control category in EdScratch. Loops are also in the Control category. This is because both conditionals and loops allow you to control the flow of your program. The **if** block and the **if-else** block have something else in common with loops too: you can stack or nest them in programs.

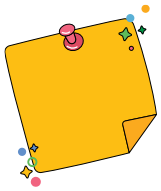
Why is that?

In block-based programming languages like EdScratch, adding blocks together is sometimes called stacking blocks. When you use multiple loops together in a program one after another, you can say you are stacking the loops. You can also stack if and if-else blocks with each other and with loops.

Likewise, just like you can nest loops by putting one loop block inside another loop block, you can nest if and if-else blocks with each other and with loops too!

What's going to happen this time?

When a program has multiple loops or conditional blocks stacked or nested together, it can be a bit confusing to follow the flow of the program. To understand what the program is going to do, you need to think about each action that is going to happen in sequence.



Don't forget

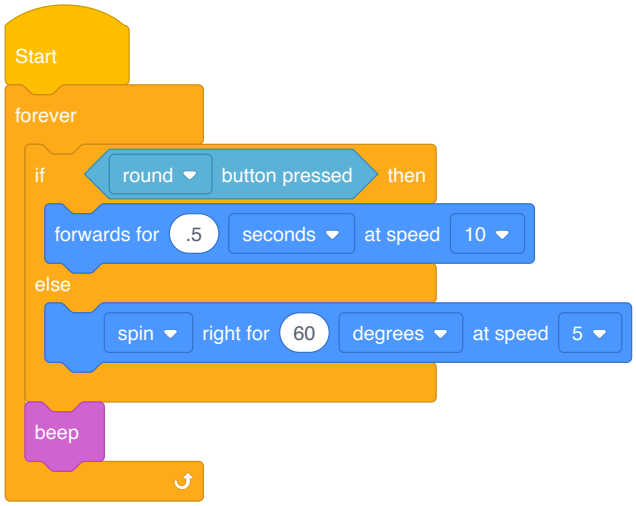
All EdScratch programs you make work in the same basic way. The program tells the robot to start with the top block and then do each action one-by-one. Once a block is executed, the program moves on to the next block.

While loops and conditionals do control the flow of a program, all programs still follow sequential order. When you look at a program with nested loops and conditionals, keep in mind that this step-by-step flow is always happening. Remembering this will help you be able to follow what is going on in a program.

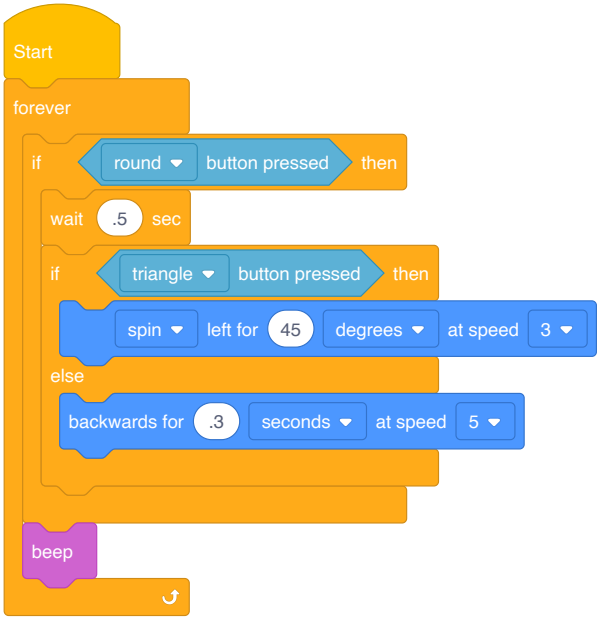
Name _____

Look at the following programs, and answer the questions:

Program 1:



Program 2:



If you run program 1 but never press the round button, what will happen? Why?

If you run program 2 but never press the round button, what will happen? Why?

If you run program 2, what do you need to do to get the robot to drive backwards? Why?

Choose one of the two programs above to write in EdScratch. Download the program and test it out in your Edison robot.

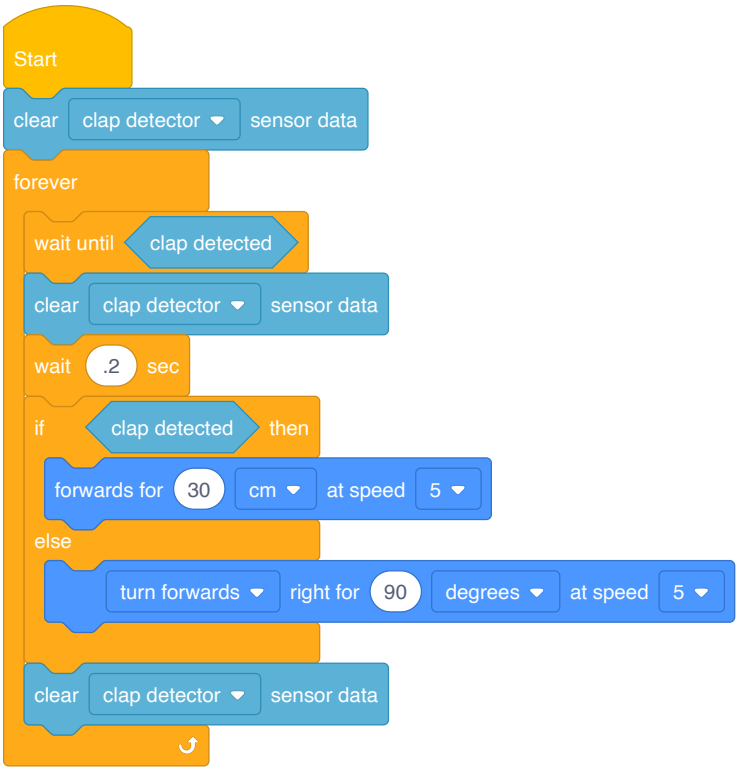
Experiment to see how these programs with nested 'if' statements work.

Name _____

Clap-controlled driving

By nesting an **if-else** block inside a **forever** loop, we can build a clap-controlled driving program for Edison. The program needs to make Edison wait until a clap is detected, then either drive forwards or turn 90 degrees, depending on if the robot detects one clap or two claps.

Look at this clap-controlled driving program:



This program uses a special block from the **Sensing** category called the **clear sensor data** block.

Take another look at the clap-controlled driving program. Can you follow how the code flows?

Write the clap-controlled driving program in EdScratch. Download the program to your Edison robot and run it. Experiment to see how the program makes Edison respond to claps.

Hint

Don't forget to use comments! Adding good comments makes keeping track of what's meant to happen in a program a whole lot easier. This is especially helpful in programs with nested loops and conditionals!

The clap-controlled driving program nests an if-else block inside a forever loop. Why do you think this is the case? What would happen if you didn't use the forever block?

Why is that?

Remember that Edison has different sensors, including the bit of tech that lets the robot detect sounds like claps. These sensors generate data when they detect specific events. Some of this sensor data is stored in Edison's memory. This stored data can sometimes be a problem, making the robot react to an old event because the robot still 'remembers' the old event.

When Edison checks if a condition has been met, if there is stored data, the robot will think that the condition has been met, even if it has not! That's why it's good coding practice to clear the sensor data. This is especially important when you use sensor events in conditionals nested inside loops. You don't want the data from a previous loop to affect the next loop!

It is also best to clear the data at the start of a program, just in case the robot has old data stored from a previous program.

Name _____

Let's explore pseudocode

Making good computer programs takes more than just writing code. You also need to be able to problem-solve when things go wrong with a program. Planning your programs before you begin coding is another important and useful skill in programming.

Just like you can plan out a story using a storyboard or plan out an essay using an outline, **pseudocode** is a tool that programmers use to help plan their programs before they start coding.

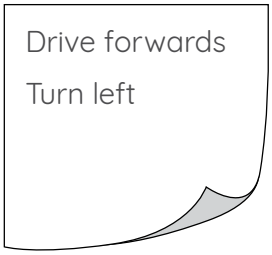
Jargon buster

Pseudocode is a way of writing out a program in a simple, easy-to-read format. Instead of worrying about syntax, pseudocode uses normal words to describe what the program will do.

Pseudocode looks a bit like a simplified programming language, but it isn't based on any specific programming language. That's why pseudocode can be used to plan programs in any coding language.

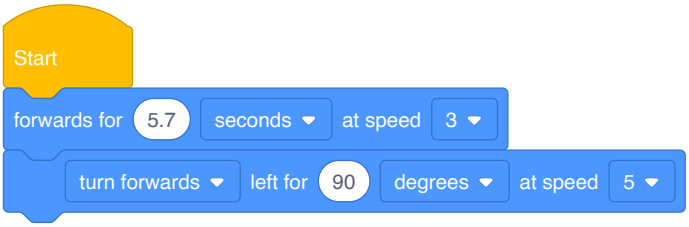
When you plan out your program using pseudocode, you don't need to write everything out in detail or worry about exactly how it will end up looking when you code it. You just need to write a simple version of your plan that makes it easy to follow the flow of the program.

Here's an example of some pseudocode for a driving program for Edison:



The pseudocode outlines the basic plan but doesn't include all of the details. Those get worked out later when you code the program.

Here's the program that the pseudocode translated into:



Do you see how the basic plan from the pseudocode translated into the program?

Writing out a plan in pseudocode first makes it easy to get the structure of your program worked out. You can then adjust it and fill in the details when you write the code.

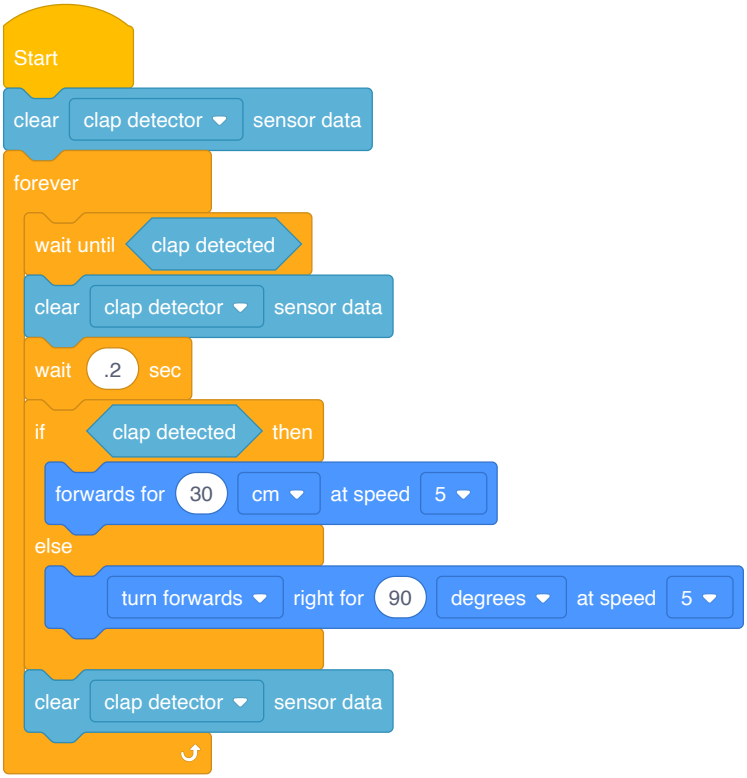
To make your pseudocode easy to read, you should keep it neat. Write your pseudocode so that it flows the same way the code will when you program it in EdScratch. That means you should write each step one after another, line-by-line.

Using pseudocode is especially helpful to plan programs that use control structures like loops or conditionals. You should indent actions that are inside of loops or conditionals to show that this code is inside of the loop or 'if' statement. Organising your pseudocode in this way makes it a lot easier to understand

Name _____

Here is an example of some pseudocode describing a clap-controlled driving program and the corresponding program in EdScratch:

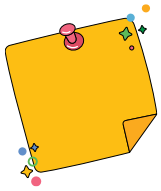
```
Clear clap data
Forever
Wait until clap
Clear clap data
Wait
If clap
Drive forward
else
Turn right
Clear clap data
```



See how the actions line up in both the pseudocode and the program?

Try it out!

Let's try reading some pseudocode instructions. Use the activity sheet on page 130 and follow the pseudocode instructions to find the answers to the question.



Don't forget

Pseudocode should always outline the basic plan, but doesn't need to include all of the details. How much detail you include is up to you and can vary depending on the program.

Follow the pseudocode. Where will the program end?

```
Start on C facing east
Forwards until food
Left 90 degrees
Repeat 6 times
  Forwards 1
  If animal
    Left 90 degrees
```

```
Start on H facing east
Repeat 3 times
  Forwards 1
  If living thing
    Right 90 degrees
  else
    Left 90 degrees

Backwards 1
```

```
Start on D facing west
Repeat 3 times
  Forwards 3 -
  if animal
    Left 90 degrees -
  else
    If food
      Right 180 degrees

Repeat 3 times
  Forward until letter
  Right 90 degrees

Backwards until number
```

Name _____

Hint

Think about how **until, if** and **if-else** conditional code works.

Will the program run the conditional code in an **if** block if the condition is **not** met? What happens instead? What about in an **if-else** block?

Make sure you follow the pseudocode just like a computer would!

(now go to the next page)

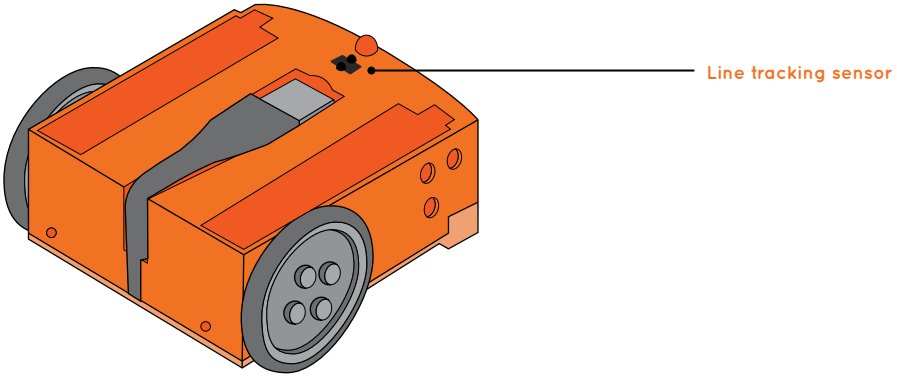
Name _____

Let's explore Edison's line tracker

Edison robots have different sensors that can detect different things. One of these sensors is the line tracking sensor.

Meet Edison's line tracking sensor

The line tracking sensor is the sensor that lets Edison see the difference between dark and light surfaces. The sensor is located on the bottom of Edison, near the power switch.



The line tracking sensor is made up of two parts: a red LED and a light sensor. Look at your Edison robot's line tracker. Do you see the two parts of the sensor?

The line tracking sensor works by shining light from the red LED onto the surface below the robot. The light sensor then measures how much of that light bounces up from the surface. Edison stores the value of the reflected light as a light reading. The more light that is reflected back to Edison, the higher the light reading.

Will a white surface or a black surface reflect more light back to Edison? Use the activity sheets on page 131 - 132 to test whether a white or a black surface is more reflective to Edison.

Turn Edison on and press the round button twice so that the red line tracking LED comes on. Lift Edison up from the paper slightly and have a close look at the round spot of light that the LED shines onto the surface. Compare how bright the spot of light appears when placed on a black surface and then on a white surface.

Hint

The more light that is being reflected, the brighter the spot will appear on the surface below.

Which surface reflects more light back to Edison, a white or a black surface? Why do you think that?

By measuring how much reflected light is coming from the surface below the robot, the line tracking sensor lets the robot 'see' the difference between dark and light surfaces. Edison doesn't see colours like a human does, however.

The robot can only tell if a surface is **reflective** or **non-reflective**. A reflective surface will shine back a lot of light from the red LED, and a non-reflective surface will shine back very little light.

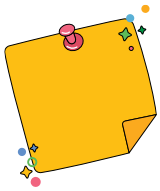
Edison sees white surfaces as reflective and black surfaces as non-reflective. What about other colours?

Will Edison see a red surface as reflective or non-reflective? What about a blue surface? Or green? Use the activity sheets on pages 131 - 132 to test all three colours using the red line tracking LED. Hint: if there is a bright spot similar to what you see on a white surface, a lot of light is being reflected, and the robot will see that colour as 'reflective'.

Colour	Reflective or non-reflective?
Red	
Blue	
Green	

Drive until a black line

We can use Edison's sensors to create inputs in EdScratch programs, telling Edison to look for different types of events and instructing the robot what to do when those events occur.



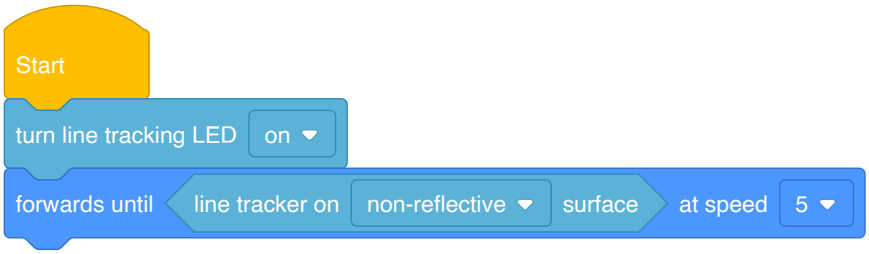
Don't forget

Inputs are the information and instructions that you give a computer. When you write a program for your Edison robot, you are telling the robot what you want it to do by giving it inputs. Edison's microchip then processes the information to tell the robot what to output as part of the input-process-output cycle.

An event is something that happens outside of the program code that affects how the program runs. An event might be a button being pressed or information being relayed from a sensor.

Let's try using the line tracking sensor in a program which tells Edison to drive until it detects a black line. To write this program, you will need to use blocks from the **Sensing** category in EdScratch.

Look at this program:



The first code block in this program turns the line tracking LED on. Whenever you want to use the line tracking sensor in a program, you need to turn it on.

Why is that?

Some of Edison's sensors are always on and checking for events. The sound sensor that can detect claps is an example of this 'always on' type of sensor.

Other sensors, like Edison's line tracker, are off by default. You need to include code in your program to turn these sensors on. Just turning the line tracking LED on isn't enough, however. You also need code to tell the sensor what event to check for (reflective surface or non-reflective surface) and what to do if that event is detected.

Write the program in EdScratch and use the activity sheet on page 131 to test it with your Edison robot. Line your robot up on the outline facing the black line on the activity sheet and run your program. Does Edison stop at the black line?

Mini challenge!

Will your program make Edison stop at the coloured lines on the activity sheet as well as the black line? Why or why not?

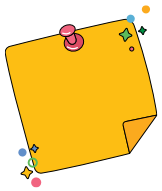
Think about whether or not the program will make Edison stop at each of the colours, then test to see if you predicted correctly!

Change it up: Drive inside a border

You can use Edison's line tracking sensor to write a program that keeps Edison driving inside a black border.

What to do

The first thing you need to do is plan your program using pseudocode.



Don't forget

Make sure your pseudocode is nice and neat and easy to read. Write each step one after another, line-by-line. You should indent actions that are inside loops or conditionals to show that this code is inside the loop or 'if' statement.

Your pseudocode doesn't need to include all of the details, but should clearly outline your plan so that you can use it to write the code later.

Your program should have Edison drive until it detects a black line. If the robot detects a black line, it should back up, then turn away from the line, and then start driving again until it detects a black line.

Write your pseudocode below

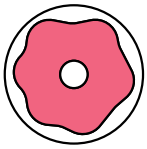
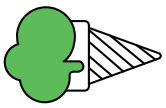
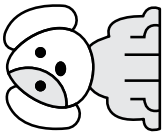
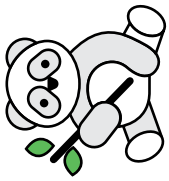
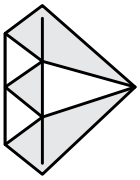
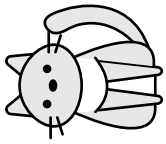
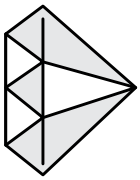
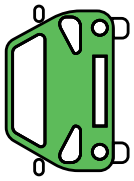
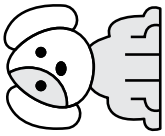
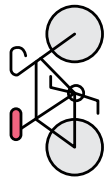
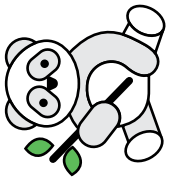
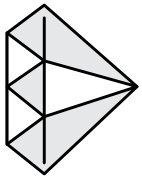
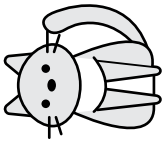
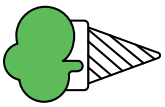
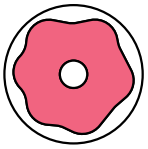
Use your pseudocode as a guide to help write your program in EdScratch. Download it to your Edison robot and test it using the activity sheet on page 133.

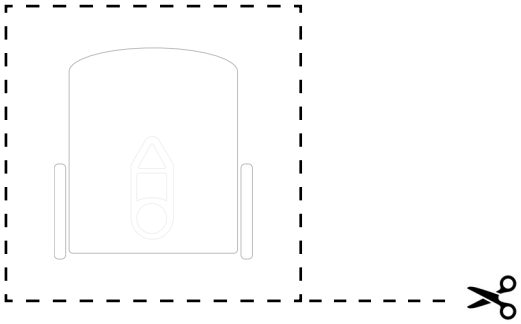
Hint

If your program doesn't work the first time, that's okay! Check the logic of your pseudocode to see if you can spot any issues. Don't forget to check for messages in the bug box too!

Name _____

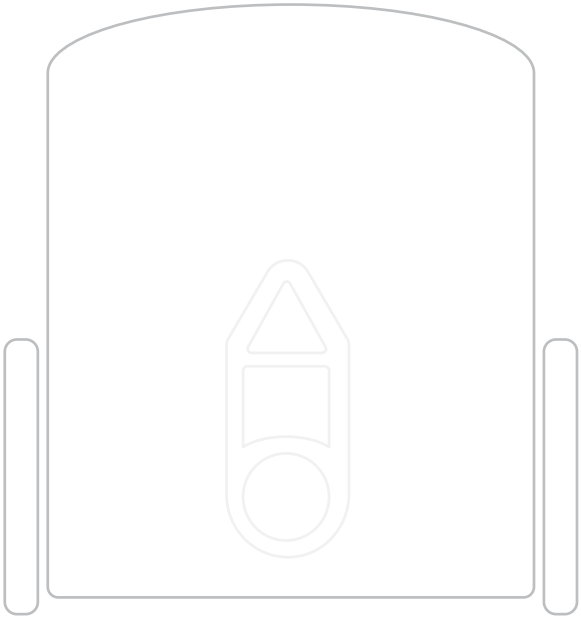
Activity sheet: Pseudocode step-by-step

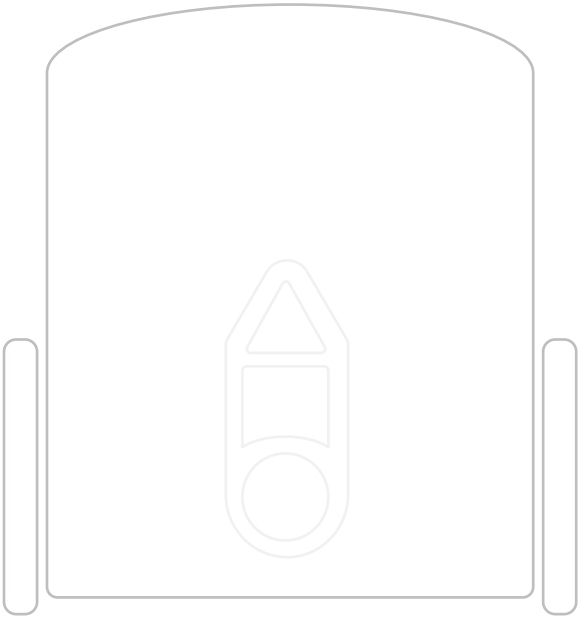
D			2	E	
		9	G		5
J			4		8
	I	7			A
		C	H	1	
	6				
		F		3	B



Name _____

Activity sheet: Line tracker test zone





Name _____

Activity sheet: Line tracker test zone