Name _____

Let's explore messaging expressions

Believe it or not, computer languages are mostly for people, not computers.

Why is that?

Computers can actually only 'think' in numbers, not words or code blocks. For example, Edison cannot understand the blocks in EdScratch the way they look on your computer screen. The blocks need to be changed into a format that Edison can understand before the program can be downloaded. This process, which can take a bit of time, is the reason why it can take a little while for the Program Edison button in the pop-up window to appear when you are downloading a program from EdScratch to Edison.

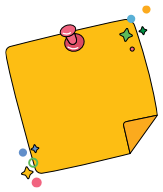
Computer languages, like EdScratch, make it much easier for us to create programs. Without a computer language, you would need to write every single command using nothing but 1s and 0s!

The blocks in EdScratch make it much easier for us to give Edison the inputs we want and to write programs telling Edison what to do. It is important to remember, however, that Edison sees all the code we program as numbers. Likewise, when Edison stores data in its memory from a sensor, this data is stored in the form of numbers.

We can use numbers in different ways in the programs we write. Being able to use numbers, and a bit of mathematics, in computer programs allows us to get the robot to do many different things. One way we can use numbers and mathematics in EdScratch is in **expressions.**

Jargon buster

In code, an **expression** is a question that can be evaluated and resolved as being either 'true' or 'false'. Expressions can be used with other code, especially conditional code like **until** blocks or **if** blocks, to control the flow of a program.



Don't forget

Inputs are the information and instructions that you give a computer.

Expressions let us compare two numbers or bits of data to each other. We can then tell Edison what to do depending on the result.

To be able to use expressions in an EdScratch program, you need to know how to read, evaluate and resolve expressions the way a computer does.

Resolving expressions

In EdScratch, the **Operators** category contains the special blocks you need to write an expression. These blocks use mathematical notations (in other words, symbols) to compare the left side to the right side of the expression. For example, 'Is A less than B?' is written in EdScratch as 'A < B' and 'Is A the same as B?' is written as 'A = B'.

Look at the list of the expressions that you can write in EdScratch:

Expression	Meaning
A = B	Is A the same as B?
A ≠ B	Is A not equal to B?
A > B	Is A greater than B?
A >= B	Is A greater than or equal to B?
A < B	Is A less than B?
A <= B	Is A less than or equal to B?

You can replace the 'A' and 'B' in the expressions with any value. You can also do computations to those values. For example, '(A + 2) > B' means 'Is A plus 2 greater than B?'

In code, expressions work in a specific order. When your expression includes computations, the expression will complete the computations first. It will then compare the left side of the expression to the right side and resolve to either 'true' or 'false.'

Because expressions have numbers and sometimes computations inside of them, it is tempting to think about them like mathematical problems which will have an answer that is a number. You need to think about expressions like a computer does, however. First, **evaluate** the expression by completing any computations on either side of the expression. Then, **resolve** the expression by comparing the left side of the notation to the right side. In other words, answer the question the expression is asking as either being 'true' or 'false'.

Why is that?

Even though expressions evaluate numbers and can contain mathematics, they always resolve to only one of two answers: either 'true' or 'false.' That's why they are so helpful when used with conditional code. Rather than having to worry about what the exact value is going to be, we can work inside sets of 'true' and 'false.'

For example, say you want a program to keep doing something as long as a value is greater than 10. Instead of writing a bunch of nested if blocks to check the exact value, you can just use one expression that says X > 10. The program will check that value, no matter what X is set to, and as long as X is greater than 10, the condition is true!

There is a special name in computer science for data which has only one of two possible values like this: we call this type of data Boolean data.

Name _____

Being able to understand what expressions mean and what they resolve to will help you follow what's going on in a program just by 'reading' it. This is an important skill in programming, known as **tracing.**

When you use programs that have values, and computations being done to those values, take a moment to trace through the program to make sure you understand what is going on. This can help you understand what should happen in the code and debug your program if things aren't working the way they should.

Jargon buster

Tracing code means working through a program line by line, recording important values. Being able to trace through a program and understand what is happening lets a programmer work out how their code should run, even when there are many different values inside that program. Tracing code can help find logical errors or bugs in the code, but it is also useful when you just need to understand what is happening in a program.

Try it out!

Try resolving the following expressions.
First, write out what each expression means, then resolve it to either true or false.

For these questions, if A = 2 and B = 4, what does each of the following expressions mean (in other words, what question is it asking) and what does each resolve to (true or false)?

(A * 2) = B
Meaning: _____
Resolves to: _____

A > = B
Meaning: _____
Resolves to: _____

(A + A) ≠ B
Meaning: _____
Resolves to: _____

(A - 1) < (B - 3)
Meaning: _____
Resolves to: _____

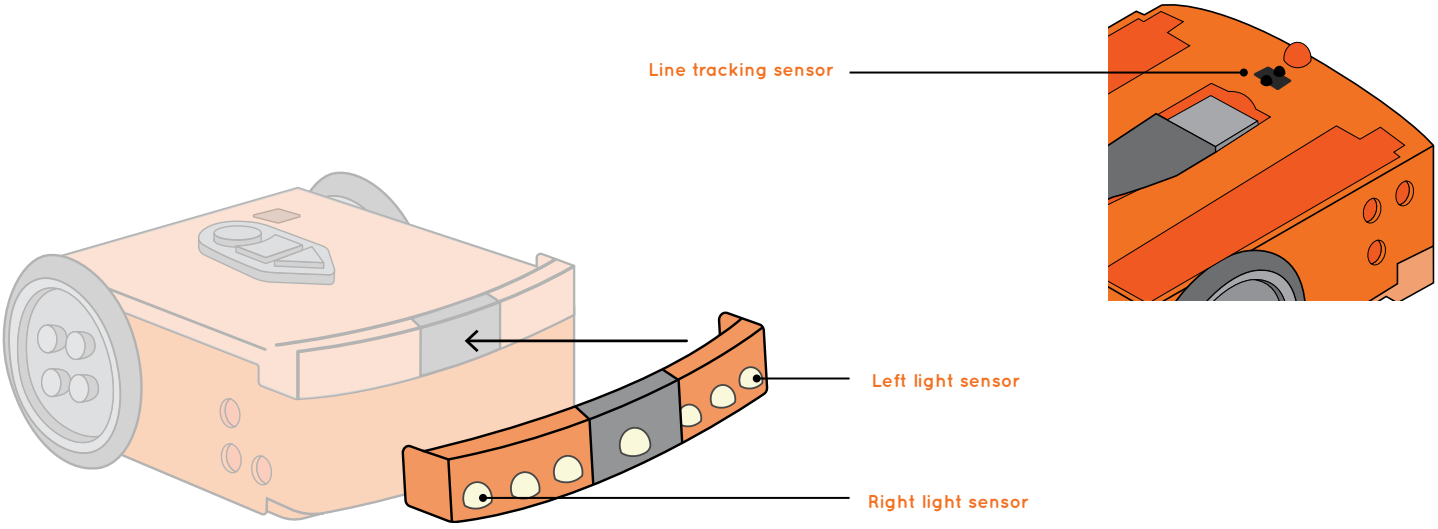
Name _____

Let's explore Edison's light sensors

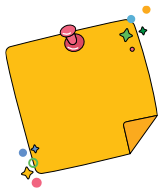
Edison robots have different sensors that can detect different things. One type of sensor Edison has is the light sensors.

Meet Edison's light sensors

Edison's light sensors are the sensors that let Edison detect and measure visible light. Edison's main light sensors are on the top of the robot, one on the right and one on the left of the robot. Edison also has a third light sensor, which is underneath the robot and works as part of the line tracking sensor. Look at your Edison robot. Do you see the different light sensors?



Much like Edison's infrared receiver can detect infrared light, the light sensors can be used to detect visible light, which is the portion of the light spectrum that people can see. All of Edison's visible light sensors work basically the same way as the light sensor in Edison's line tracker works when you use it to detect reflective and non-reflective surfaces under the robot.



Don't forget

The line tracking sensor works by shining light from the red LED in the line tracker onto the surface below the robot. The light sensor in the line tracker 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.

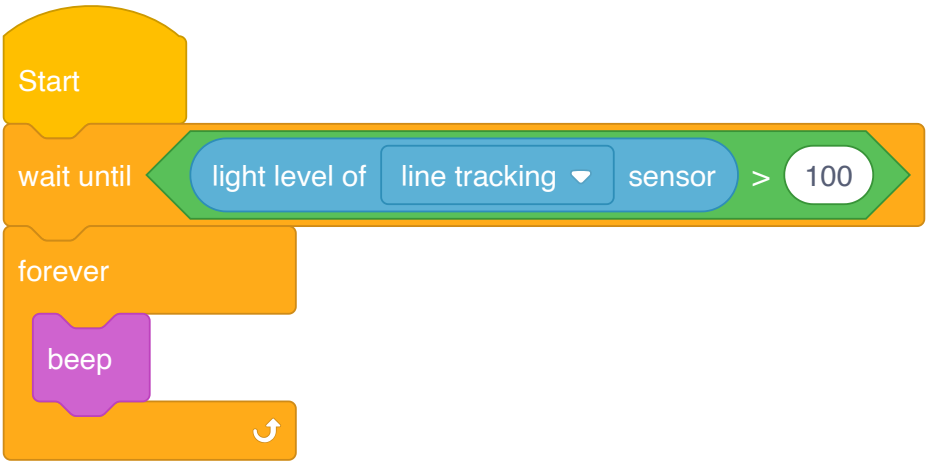
Edison's light sensors don't need to rely only on reflected light from Edison's LEDs. We can also use these sensors to detect the visible light coming from any source near Edison. The sensors measure the detected light and store the value as a light reading. The more light that is detected, the higher the light reading.

We can use the light reading from one of the light sensors as a value in an EdScratch program.

Light alarm

Let's use one of Edison's light sensors to make an alarm that will sound when Edison detects enough light.

Look at this EdScratch program:



This program tells Edison to wait until the light level of the line tracking sensor is greater than 100, then beep forever.

Why is that?

The light sensors measure any visible light detected and store the value as a light reading. Like all sensor data, Edison stores this value as a number. The expression in this program tells Edison to compare the value of the light reading from the line tracking sensor to the number 100.

Why 100? Remember that the more light that is detected, the higher the light reading. Edison's light sensors can give light readings with values ranging from 0 to just over 1000. That makes 100 a good starting point to test with this program.

Write the light alarm program and download it to your Edison robot. Before you press the play (triangle) button to run the program, cover up the line tracking sensor with something, like your thumb. Make sure the light sensor is completely covered up! Hold the robot so that the line tracking light sensor is pointing away from any bright lights, like the lights in the ceiling or sunshine coming in from a window.

Once you have the robot in position with the light sensor covered up, press play. When you are ready, move your thumb off of the light sensor and aim the robot towards a source of light. Once the robot detects enough light, the **wait until** block's condition will be met, and the code will move on to the next block, triggering the **beep** block 'alarm'.

Name _____

Automatic street lamp

Have you ever seen a street lamp that comes on when it gets dark outside? Chances are that this is done using a light sensor! You can get Edison to behave this same way, turning on the robot's red LEDs when the light level gets too low and turning them off whenever there is plenty of light.

Write a program that gets your Edison to behave like a light-sensing street lamp. Your program should have Edison turn on the red LEDs whenever the robot detects very little visible light but turn the red LEDs off the rest of the time.

Hint

You only need to use one of Edison's top light sensors to give a light level reading for this program. Choose either the left or the right light sensor.

Test different values to compare your light sensor reading to in order to see what works best.

Feeling stuck? Look at the program from task 1. How can you modify it to make your automatic street lamp program?

Download and test your program in your robot. Put Edison into a spot with lots of light, then into a spot without much light. Does your program get Edison to work like an automatic street lamp?

What does your automatic street lamp program look like it? Write it here:

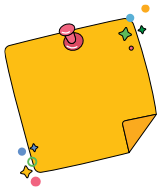
Change it up: Edison the moth

Have you ever noticed how some flying insects are attracted to bright lights? This type of behaviour is called positive phototropism. Moths exhibit positive phototropism, which is why they swarm around a bright light at night time. This kind of behaviour is also found in plants that grow towards the sun.

We can get Edison to mimic this behaviour, following the brightest light the robot detects.

What to do

Let's program Edison to behave like a moth, following a bright light. Your program should get Edison to move towards the brightest light it detects. To write this program, you will need to use the light level readings from Edison's top two light sensors.

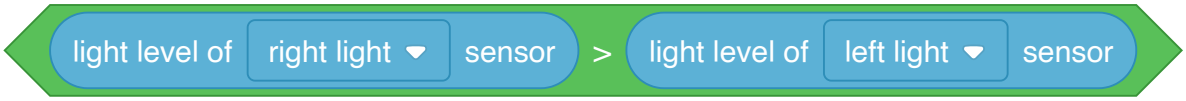


Don't forget

The light sensors measure any visible light detected and store the value as a light reading. Like all sensor data, Edison stores this value as a number. You can compare this value to another value in an EdScratch program and tell Edison how to react depending on the result.

Name _____

You can compare the light level reading from one of Edison's light sensors to a fixed number. You can also compare the light level readings from one light sensor to the light level reading of a different light sensor:



You will need to use both of Edison's top light sensors in your program. Whenever the right light sensor is giving the higher reading, the robot should move towards the right. Whenever the left light sensor's light level reading is higher, however, the robot should move in that direction.

Write your program in EdScratch, then download and test it using your Edison robot. Use a bright light source, like **a flashlight**, and test to see if Edison follows the light around.

Hint

If the room you are testing in is very bright, Edison won't be able to detect the light from your **flashlight.** If there is a bright source of light in the room, like a lot of sunshine coming in from a window, this may outshine your source of light and the robot might head for the other source of light instead!

Write your program in text box below or add a photo in image box

Name _____

Challenge up: Edison the cockroach

Some plants and animals exhibit a behaviour known as **positive phototropism**. These creatures are attracted to light, like moths which swarm around a bright light at night time. Other creatures, like some cockroaches, avoid light. This type of behaviour is known as **negative phototropism.**

Can you get Edison to mimic this behaviour, avoiding the light?

What to do

Program Edison to behave like a cockroach, moving to try to avoid any visible light.
First, plan your program out using pseudocode:

Hint

You can compare the light level reading from one of Edison's light sensors to a fixed number. You can also compare the light level readings from one light sensor to the light level reading of a different light sensor. Which approach will work best for what you are trying to do in this program?

Write your program in EdScratch using your pseudocode as a guide. Download and test your program in your Edison robot. Use a bright light source, like a **flashlight**, and test to see if Edison acts like a cockroach, avoiding the light. **Write your program in text box below or add a photo in image box**