Name _____

Let's explore using conditionals

To get a computer, like an Edison robot, to do what you want, you have to give it very specific instructions in the form of a computer program. The computer then follows your code step by step. It does whatever you told it to do.

What if you want the computer to make a decision on its own?

Most computers, including your Edison robot, cannot decide complicated things the way a person can, but you can get Edison robots to make simple decisions. The robot still needs you to give it exacting instructions to follow so that it knows what decision it is making and the rules, or **conditions**, for making that decision. To write this sort of program you need to use a type of coding structure known as a **conditional.**

Jargon buster

A **conditional**, which is sometimes called a **conditional statement**, is an element of code that is dependent on something else. This bit of code will only happen if its **condition** is met.

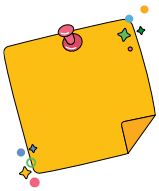
In code, a **condition** is a predetermined circumstance or set of factors that need to be met in order for conditional code to run.

One way you can use conditionals with Edison in EdScratch is by writing a program telling the robot to do something **until** a condition has been met.

Look at the EdScratch blocks in this picture:

All of these blocks are conditionals that use the same until condition formula. Each block tells Edison to do an action until a specific condition is met. But what is the condition?

Look at the **until** blocks in the picture again. Do you see the diamond-shaped hole in each block? You give a condition to an **until** block by putting a special input parameter in that hole.



Don't forget

There are three styles of input parameters in EdScratch:

- numbers you type into a block using your keypad,
- drop-down menus where you choose an option from inside the block, and
- round or diamond-shaped holes which you fill with special blocks.

Each input parameter in a block gives a different piece of information to Edison that the robot will need in order to run that command. You can think of input parameters as the answers to questions the robot has about what you are asking it to do.

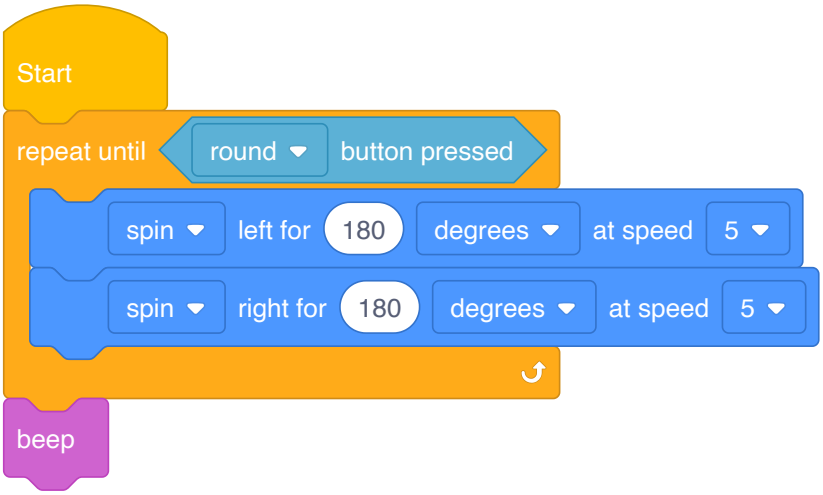
Just like any other code block, conditionals need all of their input parameters filled in to work correctly. When you use any of the **until** blocks in EdScratch, you need to give the robot the condition by using a diamond-shaped input parameter.

Open up the EdScratch programming environment and look at the different blocks. Which block categories contain blocks that you think you could use to give a condition input parameter to one of the **until** blocks? Why do you think that?

Repeat until...

Look at this EdScratch program:

This program uses a **repeat until** loop, which is an indefinite loop.



Why is that?

Any loop that repeats for an undefined number of times is an indefinite loop.

The **forever** block in EdScratch is one example of an indefinite loop because it repeats indefinitely. The **repeat until** block is another example because it will loop until its condition is met. The condition might be met after just one loop, or maybe it will be met after 20 loops, or it might never be met! Because we don't know exactly how many times the block will loop, it is an indefinite loop.

Write the program from the picture in EdScratch and download it to your Edison robot. Run the program and test it to see how it works.

When you run this program, what do you need to do to get Edison to beep? Why is that? Hint: look at the program and follow each command in sequence.

Name _____

Event + condition = event conditions

One of the main ways to use conditionals in EdScratch is by having an event be the condition. This is called an **event condition.**

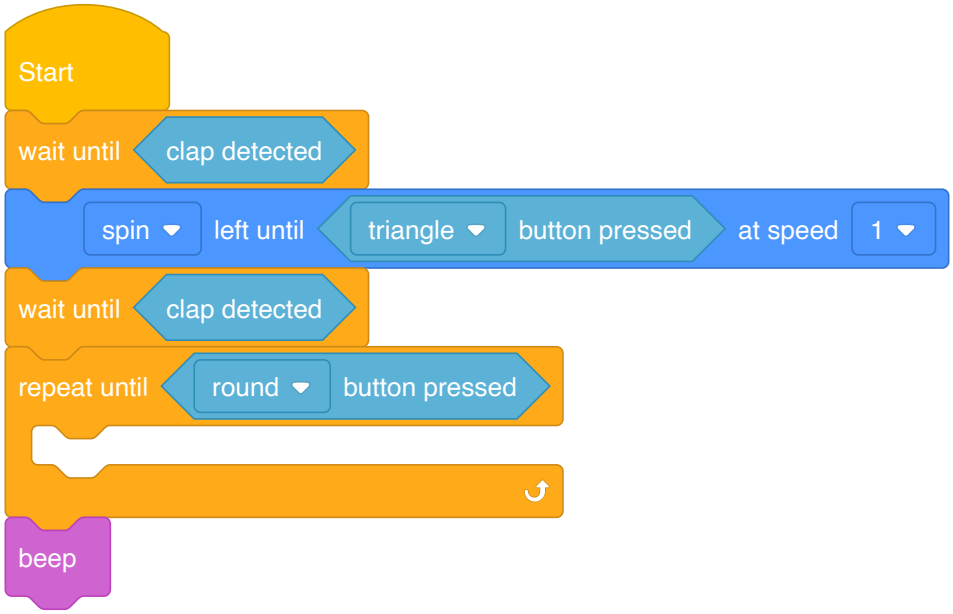
Jargon buster

Remember that in programming, an **event** is something that happens outside of the program code that affects how the program runs.

An **event condition** is a condition that requires a specific event, like a button press, to happen for the condition to be met, triggering the conditional code to run.

You can use event conditions with **until** blocks in EdScratch. Edison will keep doing the action of the **until** block until the event happens. Once the event occurs, Edison will move on to the next block in the program.

This program uses a lot of **until condition** blocks with event conditions:



Write this program in EdScratch and download it to your Edison robot. Run the program.
Can you get the program to complete every step and end successfully, with the robot returning to standby mode?

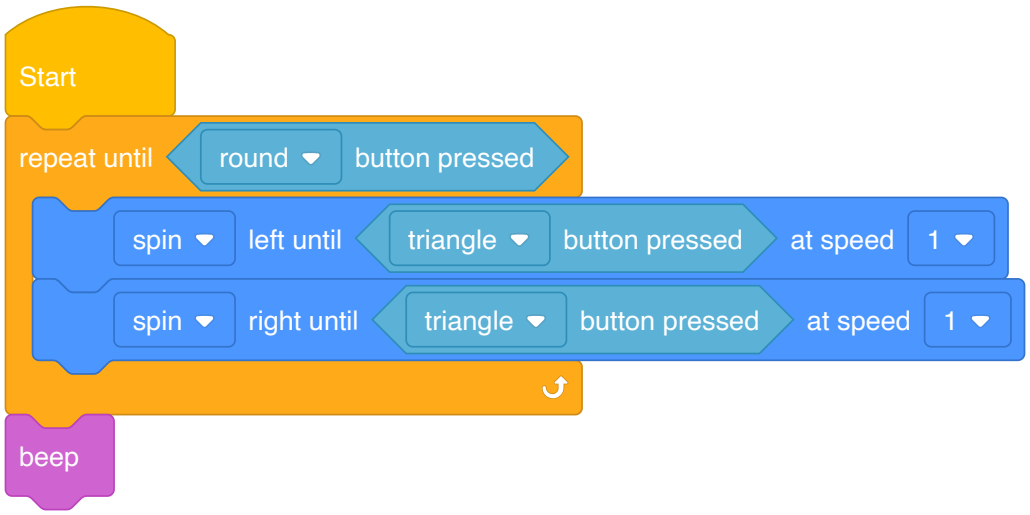
Once the program is running, how many events need to occur for the entire program to complete successfully?

The first code block in this program tells the robot to **wait until clap detected.**
Can you think of an example of something in real-life that might use this type of **wait until condition** code? What sort of device might use a program with a **wait until condition** as the first action?
What condition would trigger the conditional code? What would the program make the device do once the event condition occurred?

Change it up: Robot error or human error?

Sometimes when we write a program for Edison, it seems like the robot just won't do what we want. Like all computers, there are limits to what Edison robots can do. But before you blame the robot for your program not working, ask yourself if the problem is with the robot... or if it is with the human.

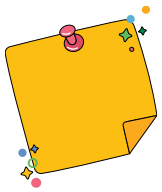
Look at this program which uses three different conditional statements:



This program isn't getting Edison to behave the way the programmer wants:

"Once the program is running, if I press the round button, I think that the robot should stop moving, then beep, and then the program should end. That's not what's happening! I also tried pressing the triangle button twice, then pressing the round button, but the robot just keeps spinning. I think the robot must be broken."

Is the robot really broken? Are there bugs in the program? Has the programmer made a logical error? Or is it something else?



Don't forget

Logical errors are problems with the logic, or the way of thinking, in a program. If a program does not work the way you expect, you may have a logical error. Edison might be running the program exactly as you have written the code, but your way of thinking about the program makes it seem like it isn't working.

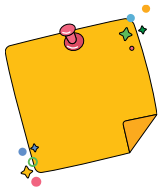
Remember, computers cannot think like a human. That's why you have to use **computational thinking** to plan, problem-solve and analyse information the same way a computer does.

What to do

Let's see if we can help figure out what's going on with the program and explain it to the programmer.

The first thing to do is run the program for yourself. That way you can see how it works and start checking for any bugs. Running the program will also let you start checking the programmer's thinking to see if they have made any logical errors.

Write the program in EdScratch and download it to your Edison robot. This program uses event conditions, so you need to start the program and then test it to see if the robot is behaving as you expect it should when each event occur.



Don't forget

To get the program running, press the 'play' (triangle) button one time. This just starts the program, it doesn't count as a button press inside the program..

Get the program running by pressing the 'play' (triangle) button. What does this get the robot to do?

Is that the behaviour you expected? Why or why not?

Now press the triangle button. The robot will start spinning right. Why does this happen?

The programmer said there were two problems with how this program works:
Problem one: "Once the program is running, if I press the round button, I think that the robot should stop moving, then beep, and then the program should end. That's not what's happening!"

Problem two: "I also tried pressing the triangle button twice, then pressing the round button, but the robot just keeps spinning."

Run the program in your Edison robot again. Follow the steps the programmer described to see if you can replicate the problems the programmer experienced. Do you have the same issues as the programmer? Do you think the problems are human problems or robot problems?

What's going on here?

Let's look at the two problems one at a time.

Problem one: "Once the program is running, if I press the round button, I think that the robot should stop moving, then beep, and then the program should end. That's not what's happening!"

This is a human problem. The programmer made a logical error when thinking about how the program should make Edison behave.

To understand the error, try running the program in your Edison again. This time, once the program is running, press the round button one time. Then press the triangle button two times.

The robot will spin left, then spin right, then beep, and the program will end.

Why is that?

Remember, the **repeat until condition** block is an indefinite loop that will loop until its condition is met.

The loop tells Edison to do each item of code inside the loop in order, then come back to the top of the loop. If the loop condition has NOT been met, then the loop tells Edison to start the code inside the loop again. You can think about it as the loop asking Edison the question, 'has the round button been pressed?' If the answer is 'no', then the loop sends Edison back into the loop. If the answer is 'yes', then the loop sends Edison to the code that comes after the loop instead.

A program will only check if the loop condition is met at the start of the loop, not while the code inside the loop is running. The robot needs to complete all of the code inside of the loop first, then it will go back to the top of the loop and check the condition.

Our programmer pushed the round button but didn't complete the conditions in the code blocks inside of the loop. That is what caused the error!

In your own words, explain the logical error that the programmer who wrote this program made.

Now, let's look at the second problem.

Problem two: "I also tried pressing the triangle button twice, then pressing the round button, but the robot just keeps spinning."

This is also a human problem, but it is not exactly a logical error. The problem here is that, compared to a robot, humans are a bit slow. That's because robots move through code really, really fast!

Why is that?

Remember, a program will only check if the loop condition is met at the start of the loop.

In this program, as soon as the triangle button is pushed the second time, the **spin right until triangle button pressed** block finishes, and the code moves to the top of the loop to check if the round button has been pressed.

This happens very fast. It takes less than 10 milliseconds before the code checks for the round button press. That's less than 1/100th of a second!

Our programmer did push the round button, but by then, the code had already checked the loop condition and sent the robot back into the loop. The programmer was too late!

Things not working the way we expect them to work happens all the time in coding. This can be very frustrating! Think about what you can do the next time you write a program that doesn't seem to work. Write a message to your future-self. Make some suggestions about what you can do to try to fix the problem.

Name _____

Let's explore if statements

Using conditionals in coding lets you write programs which ask the computer to decide something. Conditionals are how you give the computer exacting instructions to follow so it knows what decision it is making and the conditions for making that decision.

In coding, the most common way to set conditions for a computer is by using an **if statement.**

Jargon buster

An **if statement** is a conditional statement. It is an element of code that is dependent on something else. This bit of code will only happen if the condition is met. That's why it is called an 'if' statement!

You probably use 'if' statements to make decisions in life all the time, maybe without even realising it. Look at these examples:

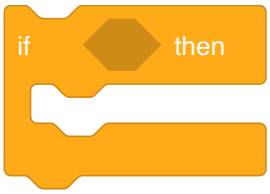
IF it is cold outside, **THEN** I put on a jacket before I leave the house.
IF I am hungry after school, **THEN** I eat a snack.

Think about a conditional you encounter in your daily life. Write it using the 'if___, then ___' formula.

IF

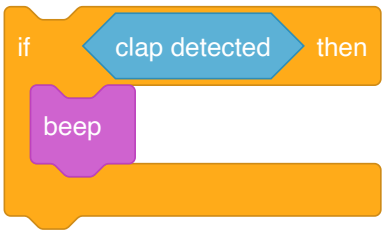
THEN

In code, 'if' statements follow this same formula. Look at the **if** block from EdScratch:



Do you see the 'if__, then ___' formula in the block? When you use an 'if' statement in code, you are telling the computer that IF condition happens, THEN do the conditional action.

For example, IF clap detected, THEN beep:



You can also tell the computer what to do if the condition does NOT happen. To do this, you need to use a type of conditional called an **if-else statement.**

Jargon buster

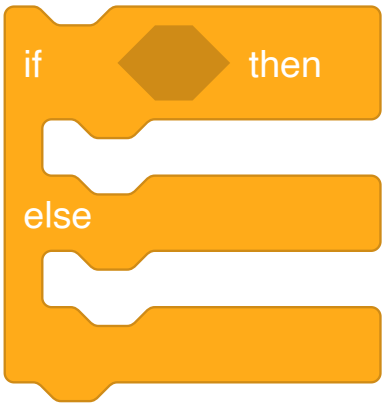
An if-else statement is a conditional statement. Just like all conditionals, this is an element of code that is dependent on something else. An if-else statement tells the program what to do if the condition is met and also tells the program what to do if the condition is NOT met.

The 'else' part of the if-else statement tells the program what to do if the condition is not met.

Name _____

You can think about an if-else statement like a decision point for the program. An if-else statement tells the program: "if the condition is met, do thing A. If the condition is not met, do thing B."

This is what the **if-else** block looks like in EdScratch:



Just like the **if** block, the **if-else** block uses the basic 'if ___, then ___' formula but also says what action to do when the condition is not meet. An if-else statement lets you make a conditional choice:

IF it is cold outside, **THEN** I put on a jacket before I leave the house. **ELSE** I go out in a tee-shirt.

IF I am hungry after school, **THEN** I eat a snack. **ELSE** I wait until dinner time to eat.

Using an if-else statement forces a program to branch. It will either go down one path, or it will go down a different path.

Try it out

Let's practice using the 'if ___, then ___ else ___' formula to see how it makes programs branch.
For this activity, you need to use the activity sheet on page 129. This activity sheet has a special treasure map that can only be solved with if-else statements.

Hint

Did you know that some mathematics symbols are also used in coding? For this activity, you will need to use these symbols:

A = B means 'A is the same as B'

A > B means 'A is greater than B'

A < B means 'A is less than B'

Follow each set of instructions to work out the location of each treasure:

The Orb of Marshmallow

Begin at the start

Repeat 3 times:

IF the number <7, **THEN** go left, **ELSE** go right

Where is the Orb of Marshmallow?

The Omelette of Space Eggs

Begin at the start

Repeat 2 times:

IF the number =9, **THEN** go left, **ELSE** go right.

Repeat 1 time:

IF the number <7, **THEN** go left, **ELSE** go right

Where is the Omelette of Space Eggs?

The Cloak of Pancakes

Begin at the start

Repeat 3 times:

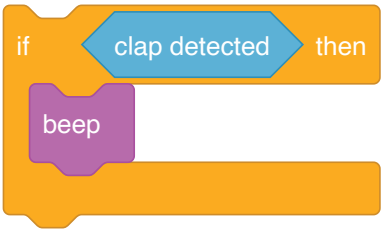
IF the number >2, **THEN** go left, **ELSE** go right

Where is the Cloak of Pancakes?

Name _____

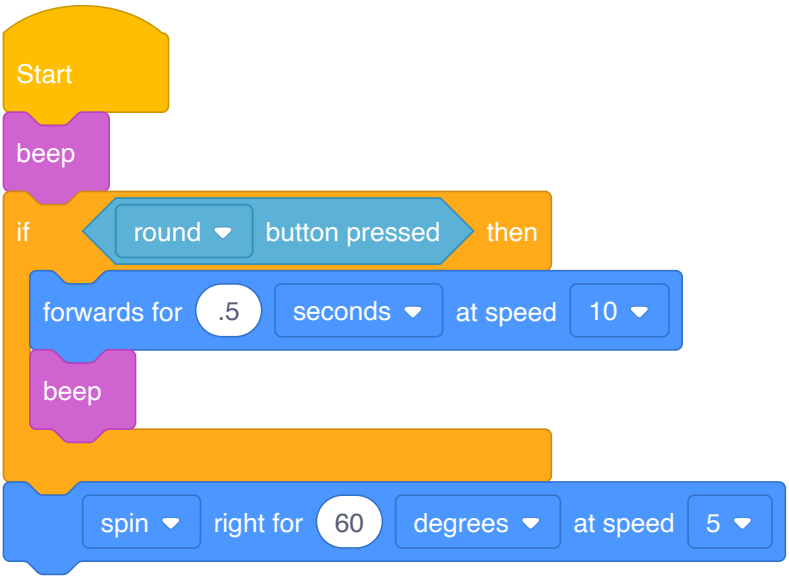
Let's explore if statements and sequence

When you use an 'if' statement in code, you are telling the computer what to do if that condition happens. In EdScratch, the **if** block needs you to give the condition by using a diamond-shaped input parameter. You also tell the robot what the conditional action is by putting a block or blocks inside the 'mouth' of the **if** block:



What happens in a program that uses an **if** block when the condition is **not** met?

Look at the following program:



In this program, what needs to happen for the condition in the if block to be met?

Write the program in EdScratch. Download the program and run it in your Edison robot.

What happened when you ran this program? Did the conditional code (the code inside the if block) run?

From what you found, what do you think happens in a program that uses an if block when the condition is NOT met?

Name _____

Why is that?

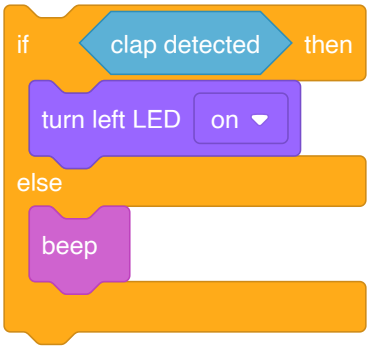
Remember, all programs move through the code step-by-step in sequential order. When the program gets to an if block, it checks to see if the condition has been met. If it has, the program runs the code inside the block. If the condition has not been met, then the program skips the code in the if block and moves on to the next line of code in the program.

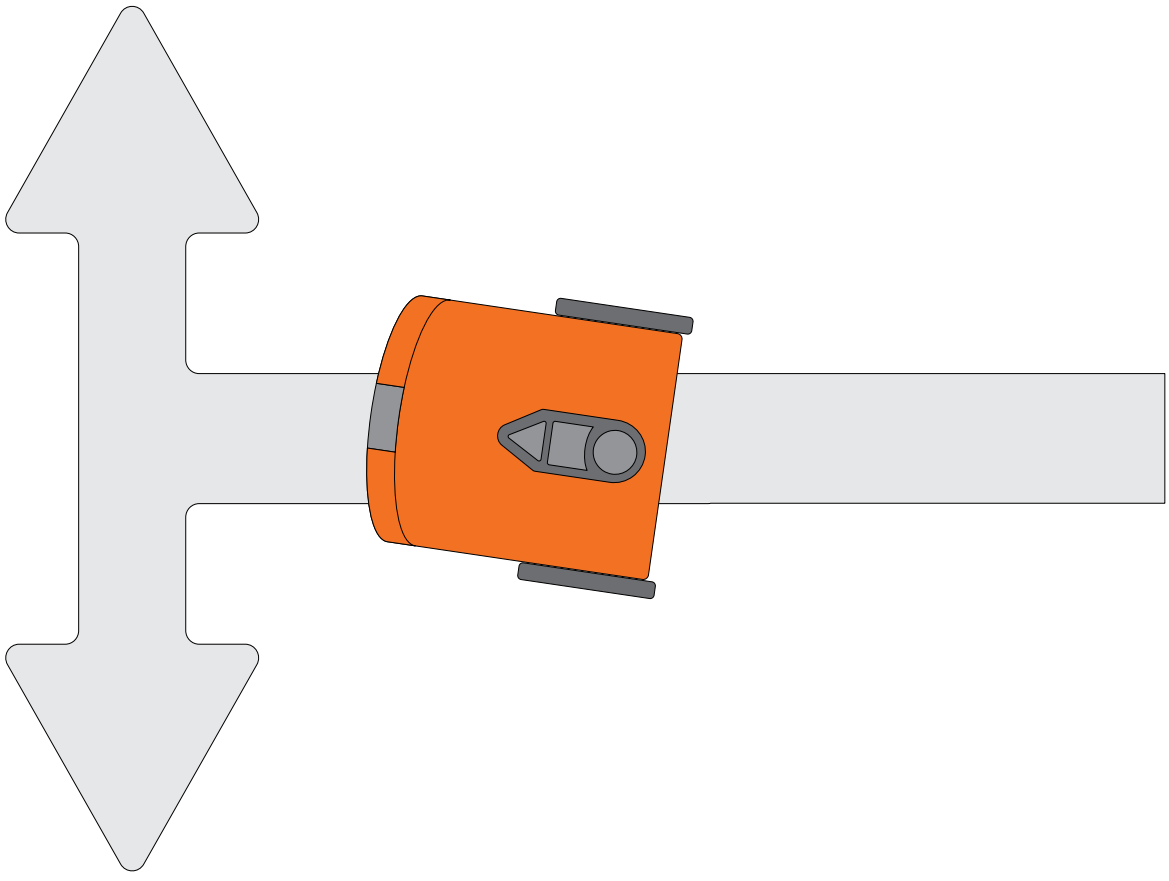
Edison moves from code block to code block very fast. It takes less than 10 milliseconds before the robot is already at the if block checking for the round button press. That's less than 1/100th of a second! It's almost impossible to press the round button in time.

If you want to see the conditional code run, what extra block could you add to the program to give yourself some more time to press the round button?

Just like an **if** block, when an EdScratch program gets to an **if-else** block, it checks to see if the condition has been met. An **if-else** block tells the robot both what to do if the condition is met and what to do if the condition is not met, so the robot has an action to take no matter what.

If the condition is met, the robot will do the code inside the 'if' part of the block. If the condition is not met, the robot will do the code inside the 'else' part of the block:

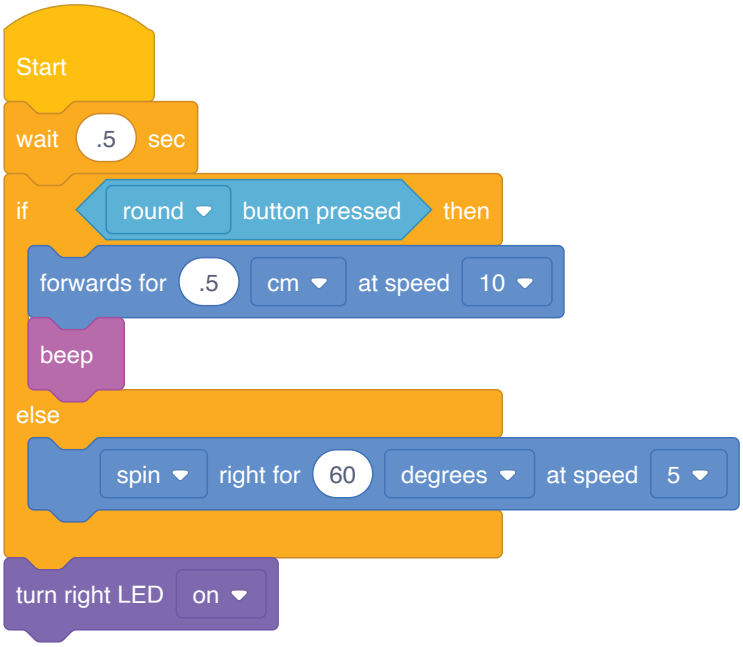




Name _____

Using an if-else statement forces a program to branch because the robot can only do the 'if' or the 'else' code, but not both. Once the robot finishes either the 'if' or the 'else' code, it moves on to the next line of code in the program.

Look at this EdScratch program:



If you ran this program in Edison and the robot did NOT detect a round button press, would the robot beep? Why or why not?

When this program runs, what actions will always happen whether or not the robot detects a round button press? Hint: Follow the program in sequential order. What three things happen no matter what?

Name _____

Activity sheet: If-else maze