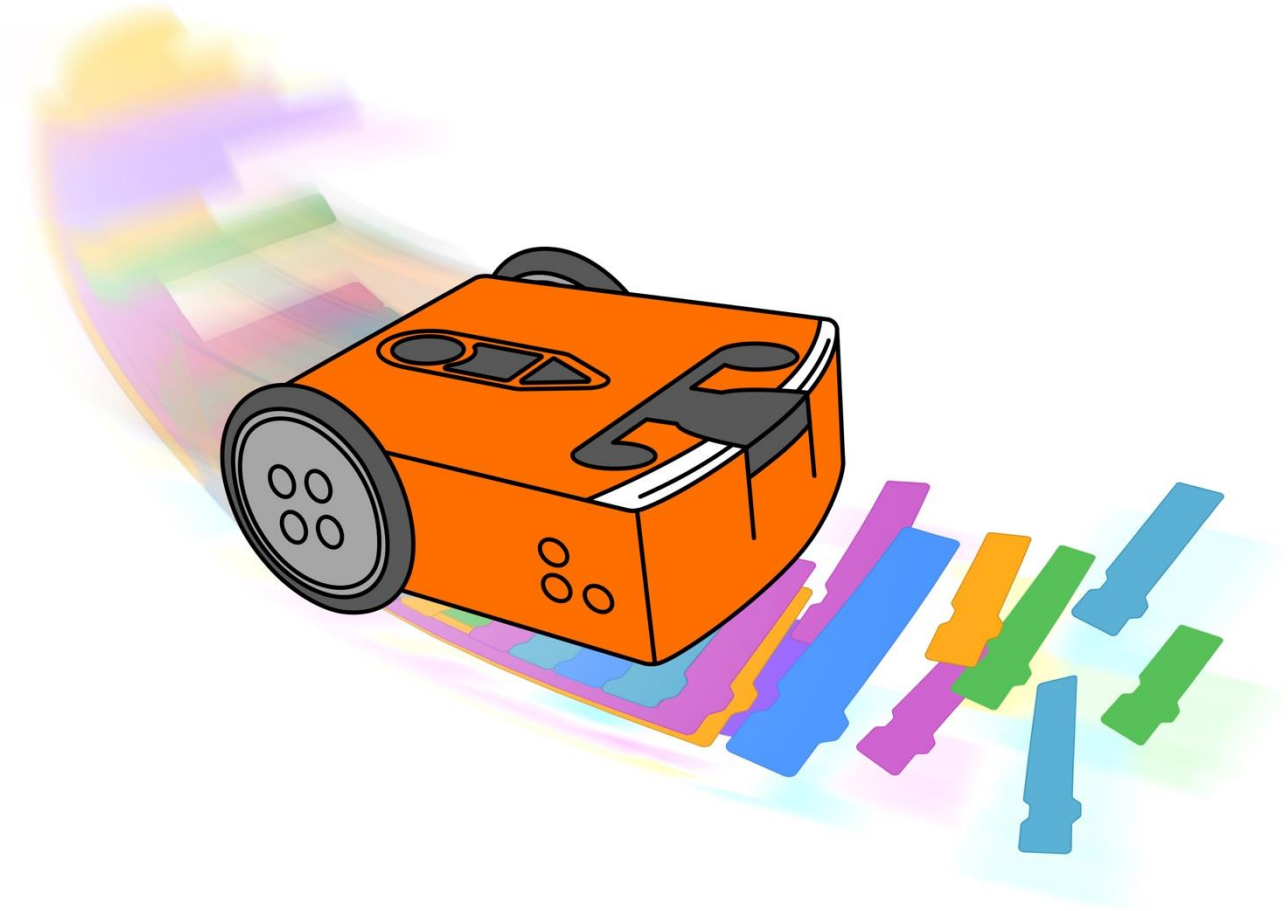
EdScratch lesson activities

Student worksheets and activity sheets



The EdScratch Lesson Plans Set by Kat Kennewell and Jin Peng is licensed under a Creative Commons Attribution-ShareAlike 4.0 International License.

Document number: 3.2.3.4.1 Rev 1.1

Contents

Unit 1: Get started 6
U1-1.1 Let's explore our Edison robots 7
U1-1.1a Change it up: Bricks, blocks and Edison 10
U1-1.2 Let's explore barcode programming 11
U1-1.2a Change it up: Sumo wrestling 15
U1-1.2b Change it up: Make your own barcode? 16
U1-1.2c Change it up: TV remote control barcodes 17
U1-1.2d Challenge up: Edison soccer 19
U1-1.2e Challenge up: Build and control the EdTank 20
U1-1.2f Challenge up: Build and control the EdDigger 21
U1-1.2g Challenge up: Build and control the EdRoboClaw 22
U1-2.1 Let's explore the EdScratch environment 23
U1-2.1a Challenge up: Download another! 28
U1-2.1b Change it up: Does EdScratch = Scratch? 29
U1-2.2 Let's explore warning messages 30
Activity sheet U1-1: Line-tracking border 32
Activity sheet U1-2: Sumo ring 33
Activity sheet U1-3: TV remote controls 34
Unit 2: Move it! 35
U2-1.1 Let's explore how computers 'think' 36
U2-1.1a Change it up: Make a PBJ sandwich 39
U2-1.1b Change it up: Human robots 40
U2-1.2 Let's explore going step-by-step in EdScratch 41
U2-1.3 Let's explore driving Edison 43
U2-1.3a Challenge up: Maze madness 45
U2-1.3b Challenge up: Self-walking pet 46
U2-2.1 Let's explore Edison's outputs 47
U2-2.1a Challenge up: Drive the maze safely 51
U2-2.2 Let's explore input parameters 52
U2-2.2a Change it up: Teach Edison to count to 9 54
U2-2.2b Challenge up: Teach Edison to count to 9 out loud 57
U2-2.3 Let's explore Edison's musical talents 58
U2-2.3a Change it up: Play a song in a round 62

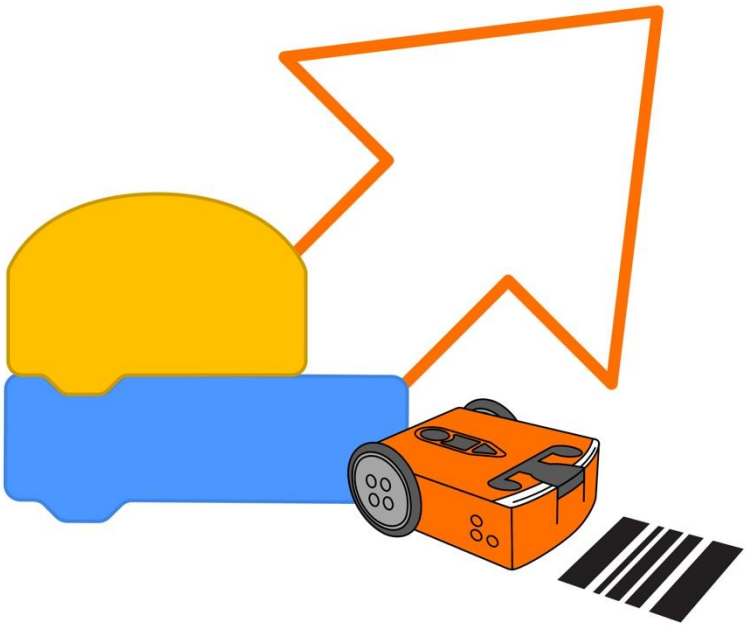
U2-2.3b Challenge up: You are the conductor....63
U2-2.4 Let's explore bugs and debugging....64
U2-2.5 Let's explore Edison's motors....68
U2-2.5a Challenge up: Spinning garden....73
U2-2.5b Challenge up: Spinning solar system....74
U2-2.5c Challenge up: Cartographer and navigator....75
U2-2.5d Challenge up: Writer and director....76
Activity sheet U2-1: Go step-by-step....77
Activity sheet U2-2: Driving track....78
Activity sheet U2-3: Mini maze....79
Activity sheet U2-4: Digital display 2....80
Activity sheet U2-5: Digital display 5....81
Activity sheet U2-6: Digital display 7....82
Activity sheet U2-7: Digital display 8....83
Unit 3: Got loops?....84
U3-1.1 Let's explore repeating steps....85
U3-1.1a Change it up: Drive a triangle....88
U3-1.1b Change it up: Drive a hexagon....89
U3-1.1c Challenge up: Choose your shape....90
U3-1.1d Challenge up: Drive a circle....91
U3-1.1e Change it up: Drive a square?....93
U3-1.1f Challenge up: Doodle-bot challenge....94
U3-1.2 Let's explore loops and sequence....95
U3-1.3 Let's explore forever loops....97
U3-1.3a Challenge up: Earworm....99
U3-1.4 Let's explore stacking and nesting loops....100
U3-1.4a Change it up: Edison the designer....104
U3-1.4b Challenge up: Dance party!....105
U3-2.1 Let's explore interrupting the main program....106
U3-2.1a Change it up: Try a clap instead....109
U3-2.1b Challenge up: Cheater bot....110
U3-2.1c Challenge up: Pick one....111
U3-2.2 Let's explore comments in coding....112
U3-2.2a Challenge up: Create and comment....115
U3-2.2b Challenge up: Share your comments....117

Activity sheet U3-1: Drive a square 118
Activity sheet U3-2: Drive a triangle 119
Activity sheet U3-3: Drive a hexagon 120
Activity sheet U3-4: Drive a circle 121
Activity sheet U3-5: Drive a quadrilateral 122
Activity sheet U3-6: Repeating squares 123
Activity sheet U3-7: Driving designs 124
Unit 4: What if 125
U4-1.1 Let's explore using conditionals 126
U4-1.1a Change it up: Robot error or human error? 130
U4-1.2 Let's explore if statements 134
U4-1.3 Let's explore if statements and sequence 137
U4-1.4 Let's explore stacking and nesting if statements 140
U4-1.4a Challenge up: Build a pulley 144
U4-2.1 Let's explore pseudocode 145
U4-2.1a Change it up: Find the answer 148
U4-2.2 Let's explore Edison's line tracker 149
U4-2.2a Change it up: Drive inside a border 152
U4-2.3 Let's explore algorithms 153
U4-2.3a Challenge up: There's more than one way to follow a line 156
U4-2.4 Let's explore Edison's obstacle detection 158
U4-2.4a Change it up: Faster, faster, smash? 161
U4-2.4b Challenge up: If line, go right. If obstacle, go left 162
U4-2.4c Change it up: Where is the obstacle? 163
U4-2.4d Challenge up: 3D maze 164
U4-2.5 Let's explore messaging with Edison 165
U4-2.5a Change it up: Remote-controlled flag machine 168
U4-2.5b Challenge up: Build and control the EdCrane 170
U4-2.5c Challenge up: Firefighting water cannon 172
U4-2.5d Challenge up: Semi-automated digger 174
U4-2.5e Challenge up: Hazardous material removal 176
U4-2.5f Challenge up: Homing pigeons 178
Activity sheet U4-1: If-else maze 179
Activity sheet U4-2: Pseudocode step-by-step 180
Activity sheet U4-3: Line tracker test zone 181

Activity sheet U4-4: Non-reflective border 182
Activity sheet U4-5: If line, go right 183
Activity sheet U4-6: Programmable TV remote codes 184
Unit 5: Versatile variables 185
U5-1.1 Let's explore expressions 186
U5-1.2 Let's explore Edison's light sensors 189
U5-1.2a Change it up: Edison the moth 192
U5-1.2b Challenge up: Edison the cockroach 193
U5-1.3 Let's explore variables 194
U5-1.3a Challenge up: Spiralling spider trap 199
U5-1.3b Change it up: Drive a random square 200
U5-1.4 Let's explore using variables with sensor data 201
U5-1.4a Challenge up: Edison the sprinter 203
U5-1.4b Change it up: Edison-controlled flag machine 204
U5-1.4c Change it up: Hey Edison, where do I go? 207
U5-1.4d Change it up: The Edison chorus 212
Activity sheet U5-1: Four lines 217
Unit 6: Inventor's time! 218
U6-1.1 Let's explore the design-build-test cycle 219
U6-1.1a Challenge up: Invent an imaginary creature 225
U6-1.1b Challenge up: Invent a cotton ball launcher 226
U6-1.1c Challenge up: Invent a burglar alarm 227
U6-1.1d Challenge up: Invent a mousetrap 228
U6-1.1e Challenge up: Invent a combination safe 229
U6-1.2 Let's explore a haunted house 230
Activity sheet U6-1: Six ideas 240
Index 241

EdScratch is developed by Microbric Pty Ltd using open source software created and maintained by the Scratch Foundation. The Scratch Foundation does not sponsor, endorse, or authorize this content. See scratch.mit.edu for more information.

Unit 1: Get started

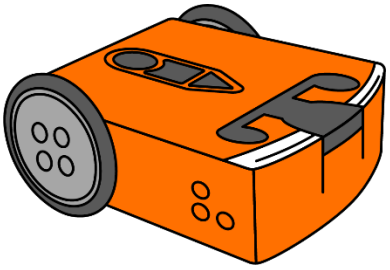


Edison

Name_____

U1-1.1 Let's explore our Edison robots

This is Edison, the programmable robot.



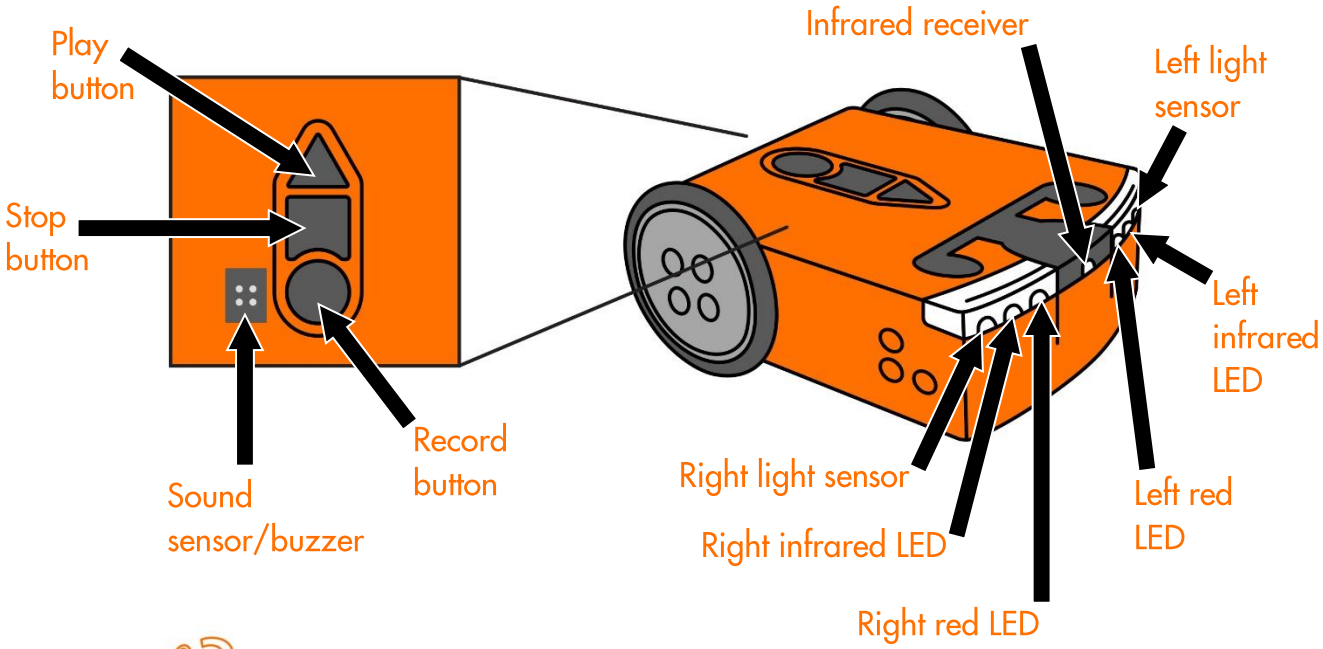
There's a lot we can do with our Edison robots. We can program the robot to do things like drive using its motors, flash its LED lights or make sounds. We can also use Edison to build robotic creations, complete mazes and a whole lot more!

Before we start using Edison, we need to get to know a bit more about the robot.

Edison uses sensors and motors to interact with the world. Edison also has three buttons a power switch and several removable parts. Knowing where Edison's parts are and what they do will help you use Edison.

Task 1: Look at Edison from the top

Have a look at the top of your Edison robot. Try to find all of the parts labelled in the picture on your Edison robot.





Why is that?

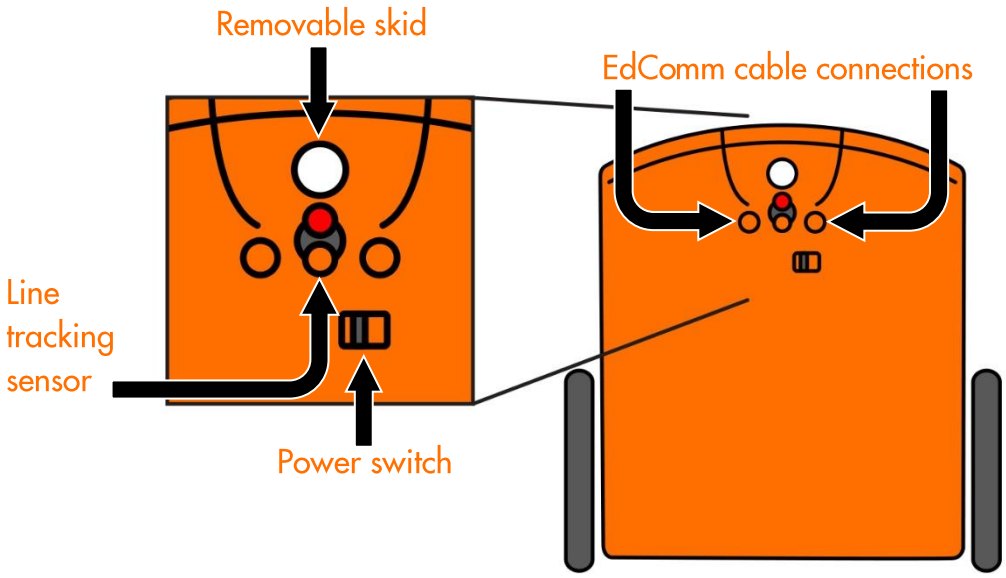
The top of Edison is made of clear plastic. This way you can see the electronic components that make Edison work. One of the most important parts is the black-coloured square that sits just above the tip of the 'play' (triangle) button. Can you see it?

This is the robot's microchip. The microchip is basically a tiny computer, which is sometimes called a micro-computer. It contains the central processing unit (CPU). That's basically Edison's brain!

Name_____

Task 2: Look at the bottom of Edison

Flip Edison over. Look at the picture and try to find all of the parts labelled in the picture on your Edison robot.



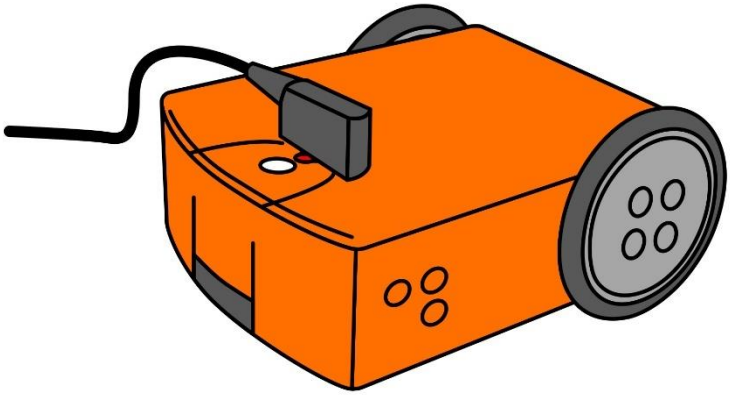
Task 3: Remove and attach the wheels, the skid and the EdComm cable

Sometimes you may want to use Edison in different ways, such as having the robot sit on its side. That's why a few of Edison's parts can be detached from the robot. Both of Edison's wheels can be taken off. Try removing one of the wheels by pulling it straight out away from the robot. Look at the powered socket where the wheel attaches. Be sure to put the wheel back in!

Next, look at Edison's plastic skid. The skid is the clear bit of plastic on the bottom of Edison near the line tracking sensor. Most of the time, you will want to keep the skid in the robot. The skid is very little, and the clear plastic can make it hard to see, so be careful whenever you remove it! You don't want to drop and lose the skid!

There is one other component which we will use a lot with the Edison robot called the EdComm cable.

You will use the EdComm cable to download your programs to Edison from your programming device, like your computer. The EdComm cable has a connection for Edison on one end, and the other end connects to the headphone socket on your computer.



For practice, try connecting the EdComm cable to Edison.

Name_____

Task 4: Turn Edison on

Whenever we want to use Edison, we need to turn on the robot. Try to turn Edison on now.

1. What happens when you turn the robot on? Describe what happens including what you saw and what you heard. Write your answer here:

_____

_____

_____

_____

_____



Don't forget

Whenever you finish using Edison, make sure you turn the robot back off!

Name_____

U1-1.1a Change it up: Bricks, blocks and Edison

Take a good look at Edison. Do you see all the bumps and holes on the top, sides and bottom of the robot?

You've probably seen studs just like the ones on the top of Edison and on Edison's wheels before. Why do you think the robot has those studs plus the holes on the sides and bottom of Edison?

Those are all connection points to build with Edison using any LEGO brick compatible building system.

There are lots of things we can build using Edison and different types of building systems. In this activity, your goal is to build something with LEGO bricks and Edison.

What to do

Get your Edison robot, grab some blocks and let your creativity and imagination flow!

Try adding blocks onto Edison's top, bottom, sides or wheels. Decorate Edison however you would like!

Once you finish, write a description or draw a picture of what your Edison looked like all brick-and-blocked up. How did you build with Edison?

Name_____

U1-1.2 Let's explore barcode programming

Just like all robots and computers, Edison needs programs to function.

What is a computer program?



Jargon buster

A **computer program** is a collection of instructions that tell a computer to perform a specific task.



Jargon is a term for the special words or expressions used by people in a particular group, like a type of job. Jargon is often difficult for people outside of that profession or group to understand. Computer programming uses some words and phrases that might seem like 'jargon' to you now – but these new vocabulary words will soon become very familiar! The jargon buster boxes in these lessons will introduce you to new terms.

Now, let's get back to programming Edison.

Using barcodes to program Edison

Edison comes with some programs already loaded in the robot. We can get the robot to access and run these programs by using special barcodes.



Why is that?

Edison's microchip has the ability to store some things, like programs. These programs are stored in the robot's **memory**. We can tell Edison which of these programs we want to run by driving over special barcodes.

Whenever you use one of Edison's special barcodes, you need to follow the same four steps:

1. Place Edison facing the barcode on the **right side** of the barcode.
2. Press the record (round) button **three times.**
3. Wait while Edison drives forward and scans the barcode.
4. Press the play (triangle) button **one time** to run the program.

Let's try using some of Edison's barcodes.

Name_____

Task 1: Clap-controlled driving

This program uses Edison's sound sensor. The sound sensor can detect loud sounds, like when you clap your hands. This program tells Edison to 'listen' for a clap.

Have Edison read the barcode.





Don't forget

To program Edison with a barcode, always follow these steps:

1. Place Edison facing the barcode on the **right side** of the barcode.
2. Press the record (round) button **three times.**
3. Wait while Edison drives forward and scans the barcode.
4. Press the play (triangle) button **one time** to run the program.

Scan the barcode, then put Edison on the floor or table before you press the play (triangle) button. After you press the play button, clap your hands one time. Edison will turn to the right.

Next, clap your hands two times. Edison will drive forward.

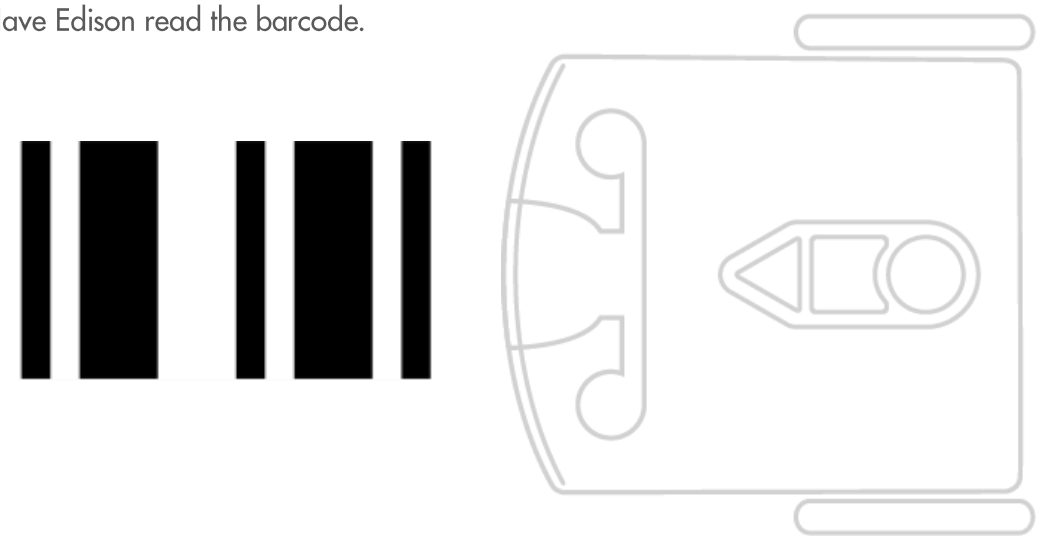
If Edison can't detect your clapping, try tapping your finger on the top of the robot near the sound sensor instead.

Name_____

Task 2: Avoid obstacles

This program uses Edison's infrared light sensor to detect and avoid obstacles in the robot's path.

Have Edison read the barcode.



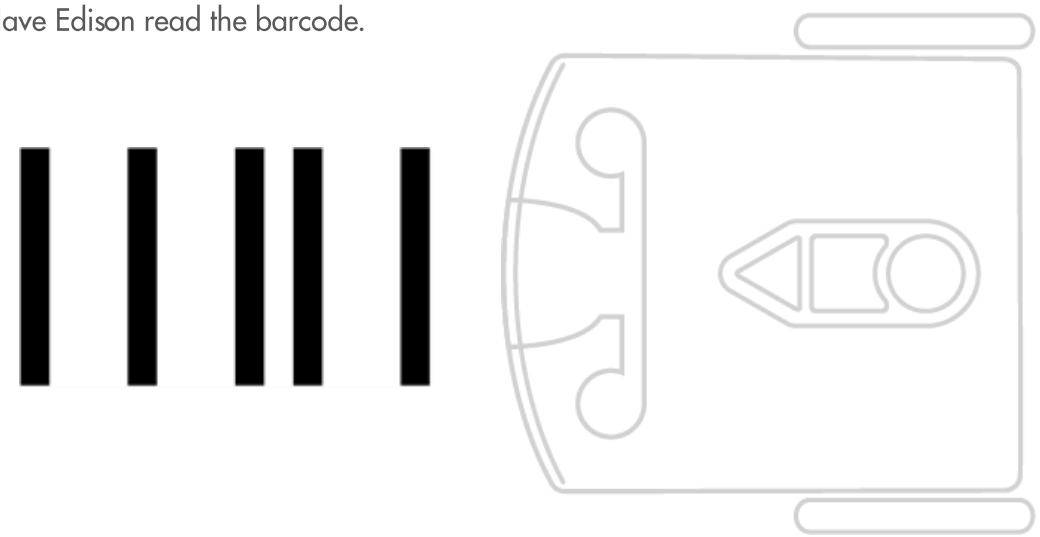
Before you press the play (triangle) button, you need to put Edison on the floor or table with some obstacles. Make some obstacles for Edison by putting some objects around Edison. Choose objects that are at least as tall as Edison and are not see-through. You can also use your hands to make little 'walls' for Edison.

Press the play (triangle) button. Watch what happens when Edison detects an obstacle.

Task 3: Follow a torch

This program uses Edison's light sensor to detect and follow a bright light. You will need a torch, a flashlight, or some other way of making a bright light for this program to work.

Have Edison read the barcode.



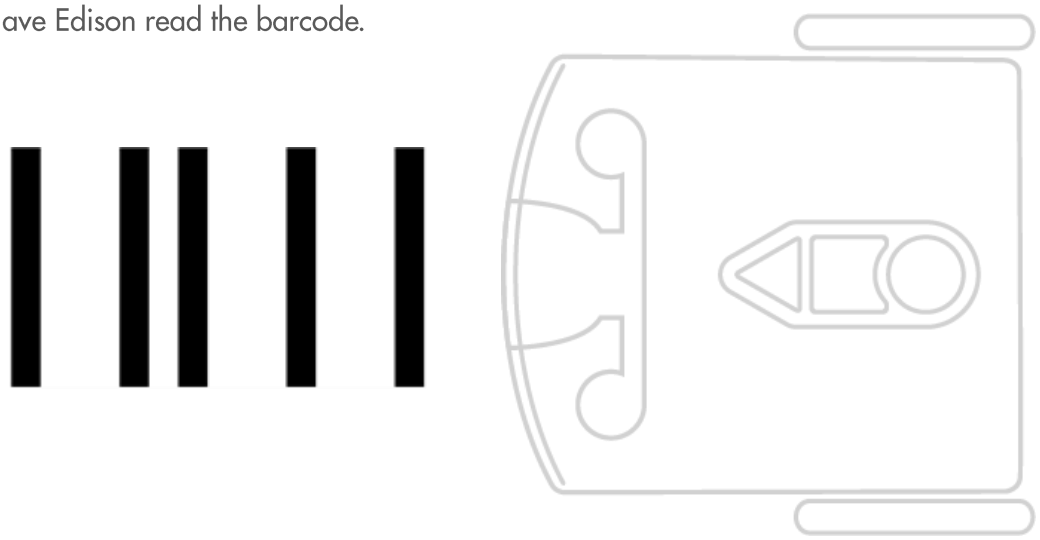
Put Edison on the floor or table and get your torch ready before you press the play (triangle) button. Shine your torch at Edison. The robot will follow the bright light.

Name_____

Task 4: Follow a line

This program uses Edison's line tracking sensor to detect and follow a dark line. You will need a dark line for Edison to follow. Use activity sheet U1-1, an EdMat or make your own line for Edison to follow.

Have Edison read the barcode.

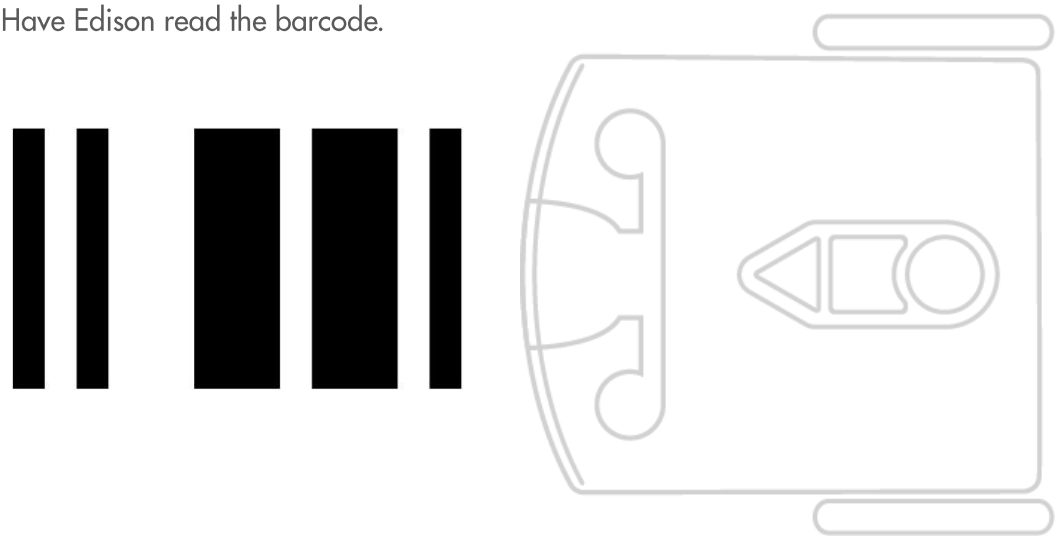


Get your activity sheet ready. You need to start the robot on the white surface near the black line. Put Edison next to the black line, **but not on the line**. Press the play (triangle) button. Edison will find and follow the line.

Task 5: Bounce in borders

This program uses Edison's line tracking sensor to detect and avoid dark surfaces. You will need a shape with a dark outline to 'trap' Edison. Use activity sheet U1-1, activity sheet U1-2, an EdMat, or make your own shape to trap Edison.

Have Edison read the barcode.



Get your activity sheet ready. You need to start the robot on the white space inside of the black line. You can put Edison next to the black line, **but not on the line**. Press the play (triangle) button. Edison will 'bounce' around inside the dark borders.

Name_____

U1-1.2a Change it up: Sumo wrestling

One of Edison's pre-set programs is actually a combination of two of Edison's other programs – bounce in borders and obstacle detection.

What does this combined program do? It allows two or more Edison robots to sumo wrestle!

The obstacle detection part of the program helps each of the robots to find the other robots. The line detection part of the program helps Edison find a line to knock the other robot out of the ring.

You will need a shape with a dark outline to be the sumo ring for the battling Edison robots. Use activity sheet U1-2, an EdMat or make your own sumo ring.

You will need to work together for this activity. Scan the barcode with at least two Edison robots.





Don't forget

To program Edison with a barcode, always follow these steps:

1. Place Edison facing the barcode on the **right side** of the barcode.
2. Press the record (round) button **three times.**
3. Wait while Edison drives forward and scans the barcode.
4. Press the play (triangle) button **one time** to run the program.

Get your sumo ring ready. If you want, you can mark the different Edison robots with labels or by attaching coloured bricks. Put all the Edison robots in the ring.

Press the play button (triangle button) on all the robots at the same time.

Each Edison robot will start to drive around the inside of the ring slowly, looking for the other robots. When one Edison detects another robot, it will speed up to hit it and try to push it out of the ring.

The Edison that stays in the ring wins!

Name_____

U1-1.2b Change it up: Make your own barcode?

Using barcodes with Edison is a lot of fun! People often want to know if they can make their own barcode for Edison.

Have a think. What do the barcodes do? Do you remember what is happening when we use a special Edison barcode? Do you think it is possible to make your own barcode for Edison to read?



Why is that?

The barcodes simply tell Edison to run the correct pre-set program when the triangle button is pressed. The actual programs are stored in a section of Edison's memory that does not ever get changed, so it is not possible to add additional programs that can be read using barcodes.

So, no, we cannot make our own barcode for Edison. But let's pretend that we can!

What to do

Pretend that you could make your own barcode for Edison that would run a program. What would that program tell Edison to do?

1. Write a description of your pretend program. Explain what Edison would do if the robot could run your program.

_____

_____

_____

_____

_____

2. Draw your pretend barcode.

Name_____

U1-1.2c Change it up: TV remote control barcodes

There is another set of special barcodes we can use with Edison. These barcodes allow Edison to react to button-press commands from your TV or DVD remote control.



Why is that?

The remote-control barcodes are a special type of barcode. These barcodes let Edison store a code from a remote control. Edison can then reference that code later.

Unlike other Edison barcodes, these barcodes don't activate a program in Edison on their own. Instead, these barcodes tell Edison to look for a remote-control code. If Edison detects a remote-control code it recognises, the robot performs a pre-programmed action.

Using these barcodes along with a TV or DVD remote control will let you drive Edison around like a remote-controlled car!

Task 1: Plan out your remote-control pairing

Look at the eight barcodes on activity sheet U1-3. Six of the barcodes tell Edison to run programs which control how Edison will move. The last two barcodes tell Edison to run programs that make sounds. You need to pair each barcode program with a different button on your remote control.

To make it easier to control Edison with the remote control, match the program action to a button on the remote control that makes sense. For example, you could use an 'up' arrow (like on 'volume up') for the drive forwards program.

Look at the remote control you are using and decide on a button for each program. Write down your button choice for each program:

Program	Remote control button
Drive forwards	
Drive backwards	
Spin left	
Spin right	
Turn left	
Turn right	
Play beep	
Play tune	

Name_____

Task 2: Program Edison with the remote control

Program Edison with each TV remote control barcode one by one.



Don't forget

To program Edison with a barcode, follow these steps:

1. Place Edison facing the barcode on the **right side** of the barcode.
2. Press the record (round) button **three times.**
3. Wait while Edison drives forward and scans the barcode.

When pairing an Edison to a TV remote, you need to do a different final step:

4. Press the button on your TV remote that you want to match to that barcode's action.

When using these remote-control barcodes, you **don't press the play (triangle) button** on Edison. Instead, press the button you just paired on the remote control. When Edison detects the remote code signal, the robot will perform that barcode's action.

Try it out!

Try controlling Edison with your remote control using the actions you've programmed with the remote-control barcodes.

Name_____

U1-1.2d Challenge up: Edison soccer

You can use a TV or DVD remote together with the TV remote control barcodes to control your Edison robot's movement. Can you control Edison well enough to play soccer against another Edison robot? Grab an opponent, set up the field and play a match to determine the Edison soccer champion!

What to do

You will need to work together for this activity. Make sure each player has an Edison robot and a remote control. All the Edison robots playing in the match need to be programmed with the TV remote control barcodes. Use the barcodes on activity sheet U1-3.



Hint!

You might want to use different commands from the other players. Otherwise you might end up controlling your opposition!

You also need to set up a field for the robots to play on, plus goals and a ball.

How can you design the field? What size ball will work best?

Experiment to see what works!



Name_____

U1-1.2e Challenge up: Build and control the EdTank

The TV remote control barcodes let you control your Edison robot to move in different ways. The remote-control barcodes that control Edison's movement are actually controlling the robot's motors. What happens if the motors don't have wheels attached, but something else?

What to do

In this activity, you will build and control the EdTank.

The EdTank is a remote-controlled tank that you can drive around. You can use a second Edison robot as well to build a top layer to the tank with a cannon which you can use to fire a rubber band.

Use this link

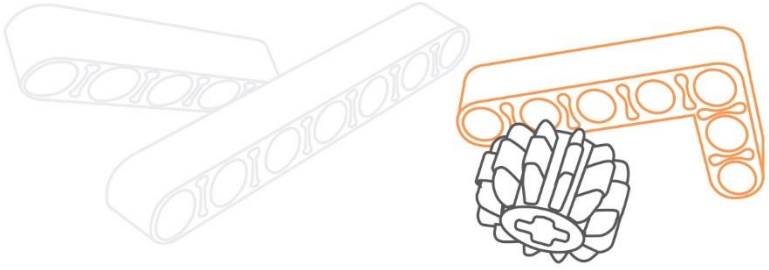
Go to meetedison.com/content/EdCreate/EdBuild-EdTank-instructions.pdf

This link will take you to the step-by-step instructions for building and programming the EdTank.

Try it out!

Once you have built and programmed the EdTank, try driving it around!

1. Do you notice any differences in how the EdTank drives compared to how Edison normally drives when the robot just has its wheels attached? Think about what might cause any differences you notice. What might be affecting how the EdTank drives?



Name_____

U1-1.2f Challenge up: Build and control the EdDigger

The TV remote control barcodes let you control your Edison robot to move in different ways. The remote-control barcodes that control Edison's movement are actually controlling the robot's motors. What happens if the motors don't have wheels attached, but something else?

What to do

In this activity, you will build and control the EdDigger.

The EdDigger is a remote-controlled excavator, or digger, with a scoop that you can drive around. The digger scoop of the EdDigger can lift and lower, plus it can carry small objects.

Use this link

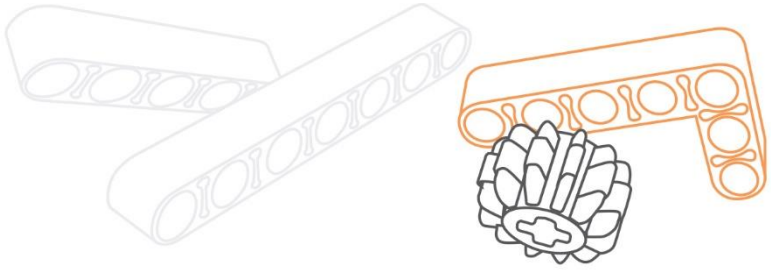
Go to meetedison.com/content/EdCreate/EdBuild-EdDigger-instructions.pdf

This link will take you to the step-by-step instructions for building and programming the EdDigger.

Try it out!

Once you have built and programmed the EdDigger, try driving it around! Be sure to try to scoop up some objects too!

1. Can you operate the digger scoop smoothly? What do you need to do to get the EdDigger to scoop up objects? What do you need to do to get the EdDigger to drop off objects it was carrying?



Name_____

U1-1.2g Challenge up: Build and control the EdRoboClaw

The TV remote control barcodes let you control your Edison robot to move in different ways. The remote-control barcodes that control Edison's movement are actually controlling the robot's motors. What happens if the motors don't have wheels attached, but something else?

What to do

In this activity, you will build and control the EdRoboClaw.

The EdRoboClaw is a remote-controlled robotic arm with a moving base that you can drive around. The robotic arm of the EdRoboClaw can open and close to pick up and carry objects.

Use this link

Go to meetedison.com/content/EdCreate/EdBuild-EdRoboClaw-instructions.pdf

This link will take you to the step-by-step instructions for building and programming the EdRoboClaw.

Try it out!

Once you have built and programmed the EdRoboClaw, try driving it around! Make sure you try to pick up and carry some objects too!

1. Experiment with different objects using the EdRoboClaw. What types of objects can you carry? What types of objects didn't work? Think about what the objects that worked well have in common with each other. What makes a good object for the EdRoboClaw?

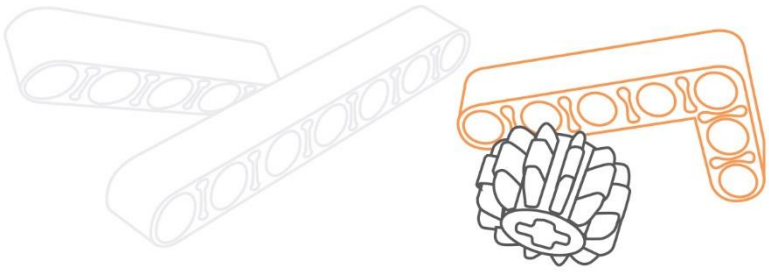
_____

_____

_____

_____

_____



Name_____

U1-2.1 Let's explore the EdScratch environment

One of the best things about Edison is that you can make your own programs for your robot! To write a program for Edison, we need to use some special **software**.



Jargon buster

All computers have two main parts: hardware and software.

Hardware is the physical parts of a computer (or robot).

Software is the set of programs and applications that make hardware, like a computer or a robot, run.

The software we will use with Edison is a robot **programming language**.



Jargon buster

A **programming language** is a set of rules and instructions used to write computer programs. EdScratch is a programming language specially designed for programming Edison robots.

The programming language we will use is called EdScratch. Let's learn a bit about the EdScratch programming language.

Task 1: Check out EdScratch

You can access EdScratch online.



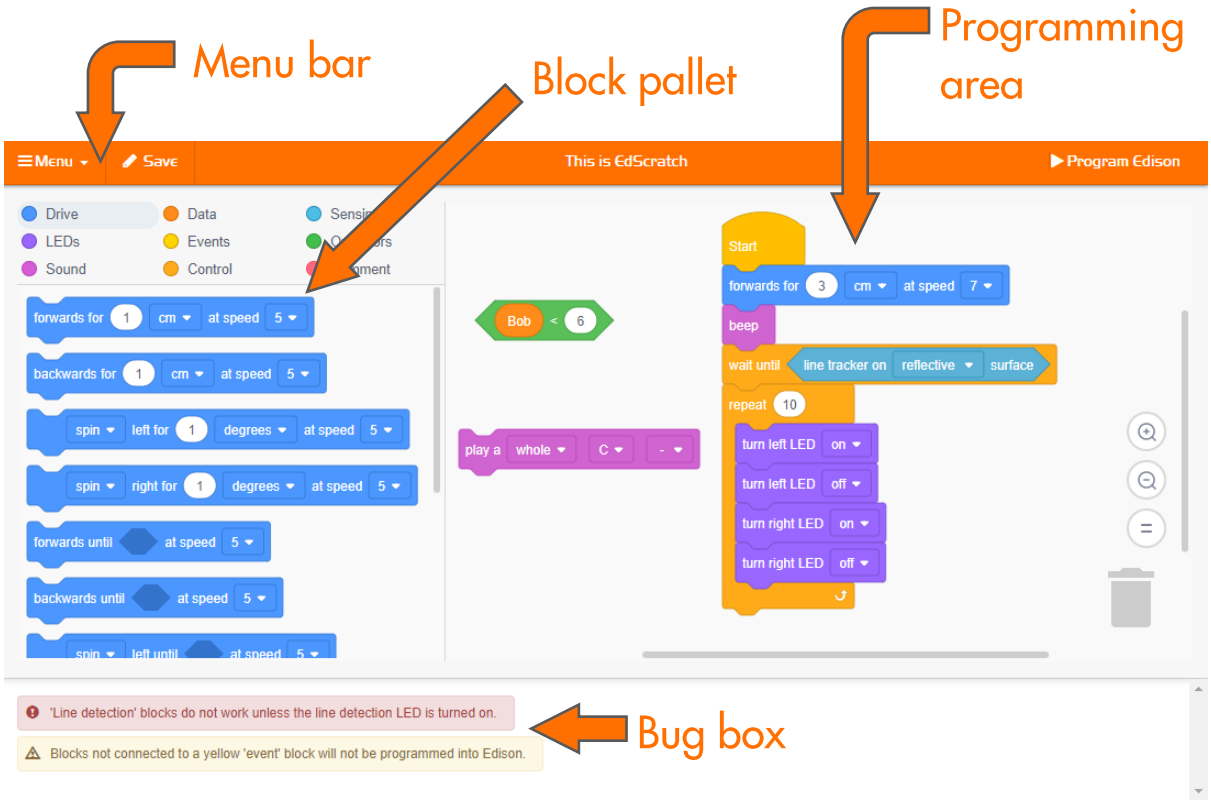
Use this link

Go to www.edscratchapp.com

Whenever you want to program Edison using EdScratch, you will always need to go to the EdScratch app.

Name_____

Here is what the EdScratch environment looks like:



The EdScratch programming environment has four main parts:

Block pallet

All of the blocks you can use are in the block pallet. To use a block, select it from the block pallet, and drag it into the programming area.

Programming area

The large area where you can connect blocks together into programs is called the programming area. Drag and drop blocks from the block pallet into this area to use them in your program.

Menu bar

Options such as 'Save' and 'Load' are accessed from the menu bar. The menu bar also has the 'Program Edison' button.

Bug box

Below the block pallet and programming area is the bug box. Warning messages will show up in the bug box.

Look at EdScratch on your computer. Find each of the four main parts of the EdScratch environment.

Task 2: Load and download the test program

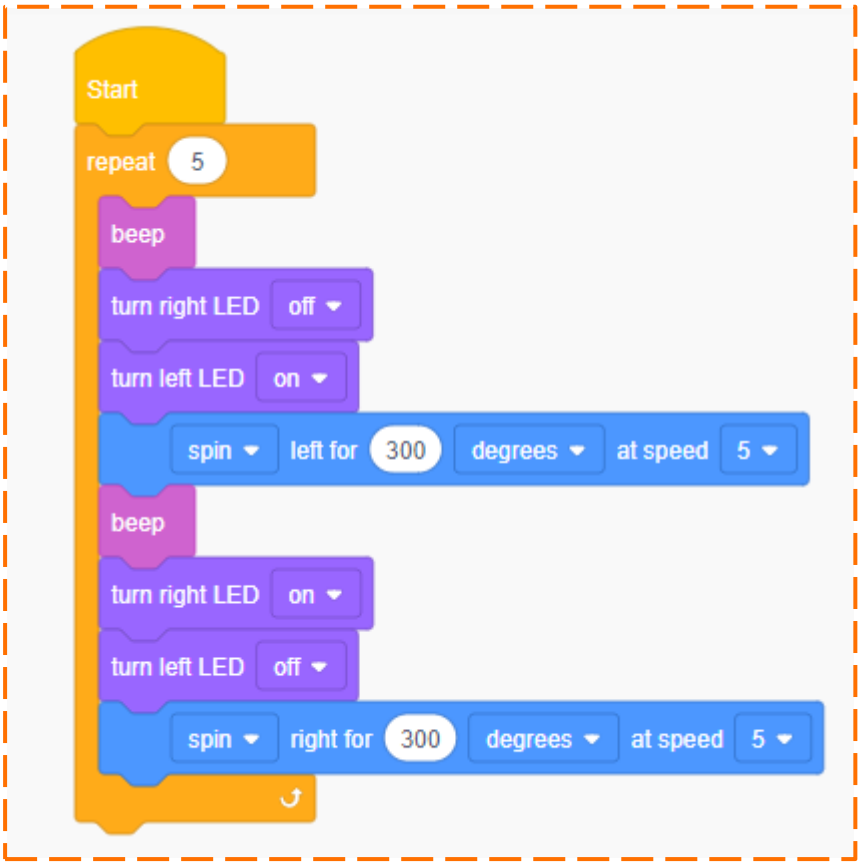
EdScratch has some demo programs already written. Try loading and downloading the demo program called Test_program.

Load the Test_program demo program

To load the demo Test_program, follow these steps:

1. In EdScratch, go to the menu bar and select the menu drop-down. Find and select the option called **Load Demos**. This will open a pop-up window with all of the demo programs.
2. Find and select the program called Test_program. The program will load in the programming area.

Here is what the Test_program looks like:



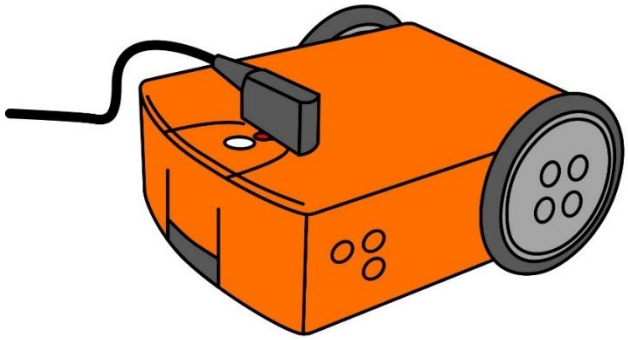
Once the program loads in the programming area, you can download it to your Edison robot.

Name_____

Download Test_program to Edison

Whenever you want to download a program from EdScratch to Edison, you need to follow these steps:

1. Connect Edison to your computer using the EdComm cable.
2. Make sure the volume is turned up all the way on the computer.
3. Press the record (round) button on Edison **one time**.
4. Go to the menu bar in EdScratch and click on the Program Edison button.
5. A pop-up window will open. Once the program is ready, a button called Program Edison will appear at the bottom of the pop-up window.
6. Click on the Program Edison button in the pop-up window.





Why is that?

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 can take a bit of time.

That's why it can take a little while for the Program Edison button in the pop-up window to appear.

You will hear the program downloading to Edison. Once it is done downloading, Edison will make the 'success' beep. Don't unplug Edison until you hear the beep!



Why is that?

Edison will let you know if the program downloads correctly by making the 'success' beep. This is the same sound you hear when you first turn Edison on.

There's another sound Edison might make if a program does not download correctly. We call this the 'fail' sound. It means something went wrong when the program tried to download. If Edison makes this sound, try starting your download again.

After you hear Edison make the 'success' beep, unplug the robot from the EdComm cable. Press the play (triangle) button **one time** to run the program.

Name_____

Try it out!

Load the Test_program demo program in EdScratch. Download and run the program with your Edison robot. Then answer the following questions.

1. Which part of the EdScratch environment is the Program Edison button located in?

_____

_____

2. How many warning messages are there in the bug box when you load the Test_program?

_____

_____

3. What does the robot do when you run the Test_program? Describe what happens.

_____

_____

_____

_____

_____

Name_____

U1-2.1a Challenge up: Download another!

There are multiple demo programs in EdScratch. Choose a program other than Test_program from the demo program list.

Try downloading and running the demo program of your choice to see what that program does.

1. What was the name of the program you chose?

_____

2. What did you expect the program to do? Did the program do what you expected?

_____

_____

_____

3. Look at the program you chose in EdScratch. Think about what the robot does when you run the program. What do you notice? How do the blocks in the program relate to what the robot does when you run the program in Edison?

_____

_____

_____

_____

_____

Name_____

U1-2.1b Change it up: Does EdScratch = Scratch?

Does EdScratch look familiar to you? It might, especially if you have done any projects using the programming language Scratch.



Why is that?

EdScratch does look an awful lot like Scratch. That's on purpose! In fact, EdScratch was built using Scratch as a base.

So, if EdScratch was made using Scratch as a base, does that mean EdScratch is the same thing as Scratch?

Nope!

EdScratch and Scratch are different programming languages. They do have some things in common, but there are a lot of things different about the two languages as well. Most importantly, you cannot program your Edison robot using Scratch. You have to use EdScratch for that!

What other things are different about EdScratch and Scratch? What do these two programming languages have in common?

Try it out!

Have a look at Scratch.

Compare it with EdScratch. What is the same and what is different?



Use this link

You can see Scratch using this link www.scratch.mit.edu

Don't forget to look at EdScratch too! Go to www.edscratchapp.com

1. Find three things about Scratch and EdScratch that are the same. Describe each one.

_____

_____

_____

2. Find three things that are different in Scratch and EdScratch. Describe each one.

_____

_____

_____

Name_____

U1-2.2 Let's explore warning messages

Some programming languages have special features to make it easier to use that language. One example of this is the bug box in EdScratch.

Sometimes when we write a program for Edison in EdScratch, something isn't quite right. When this happens, a warning message will show up in the bug box.



Don't forget

The bug box is located below the block pallet and the programming area in EdScratch.

There are two types of warning messages: yellow warning messages and red warning messages.



Why is that?



Yellow warning messages are caution messages. This is EdScratch saying "Heads up! This might not work the way you want it to work." You can download a program even if there are yellow messages in the bug box.



Red warning messages are like 'stop' messages. These messages are EdScratch saying "Sorry! This program won't make sense to Edison." If there are any red warning messages in the bug box, you will not be able to download the program to Edison.

Whenever you write programs for Edison, it is a good idea to check the bug box before you try to download the program. The warning messages can help you fix up your program!

Try it out!

In EdScratch, find and load the demo program called Warning_messages_demo.



Don't forget

To get to EdScratch go to www.edscratchapp.com

Go to the menu bar and select the menu drop-down. Find and select the option called **Load Demos**. This will open a pop-up window with all of the demo programs. Find and load the program called Warning_messages_demo.

Name_____

Once the program loads in EdScratch, answer the following questions.

1. Try downloading this program to your Edison robot. What happens? Does it work? Why or why not?

_____

_____

_____

2. Read the red message in the bug box. Look at the program. Can you fix the problem? Describe what you did to fix the red message.

_____

_____

_____

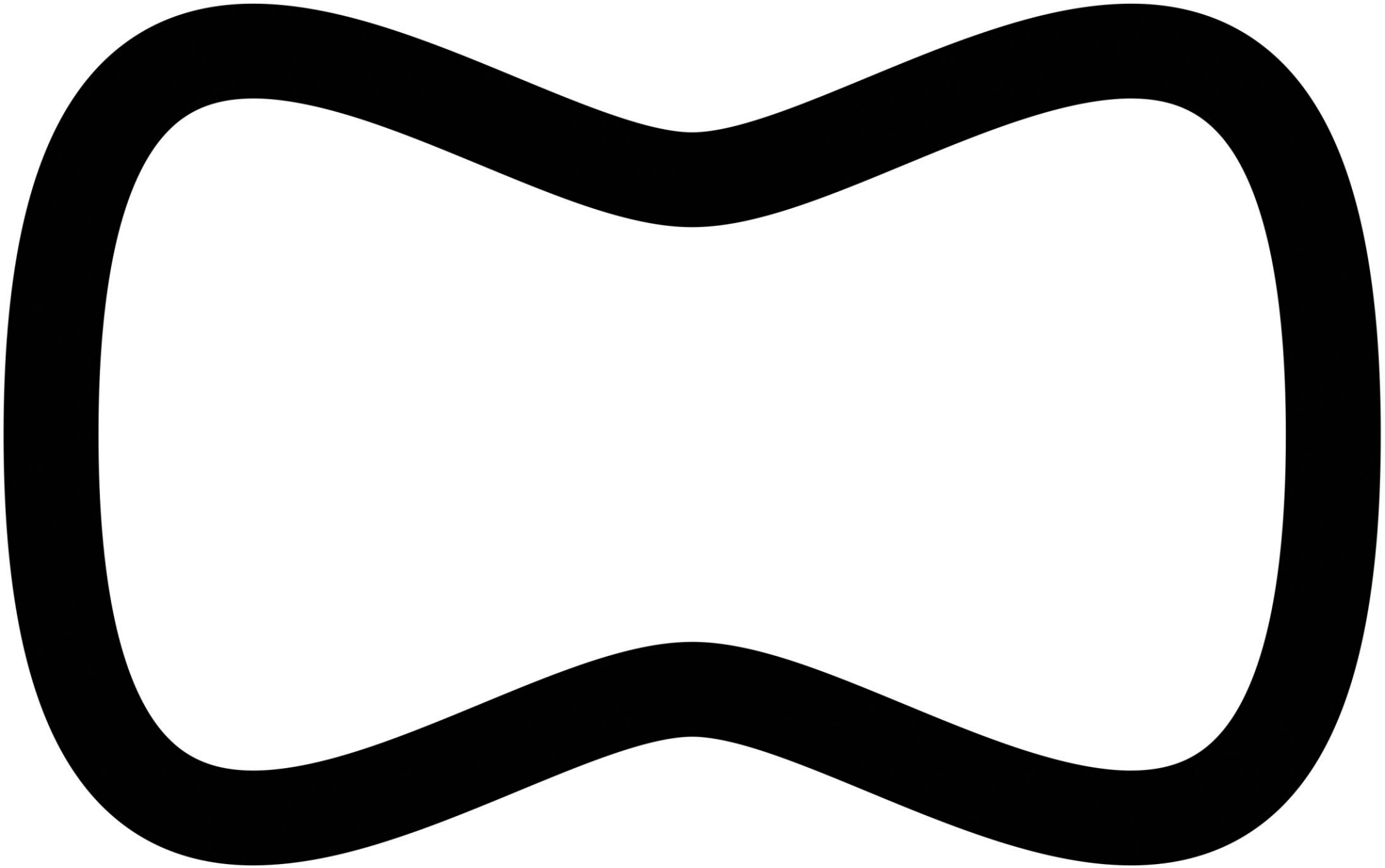
3. Read the yellow message in the bug box. Look at the program. If you download the program while that yellow message is there, which blocks will not be programmed into Edison?

_____

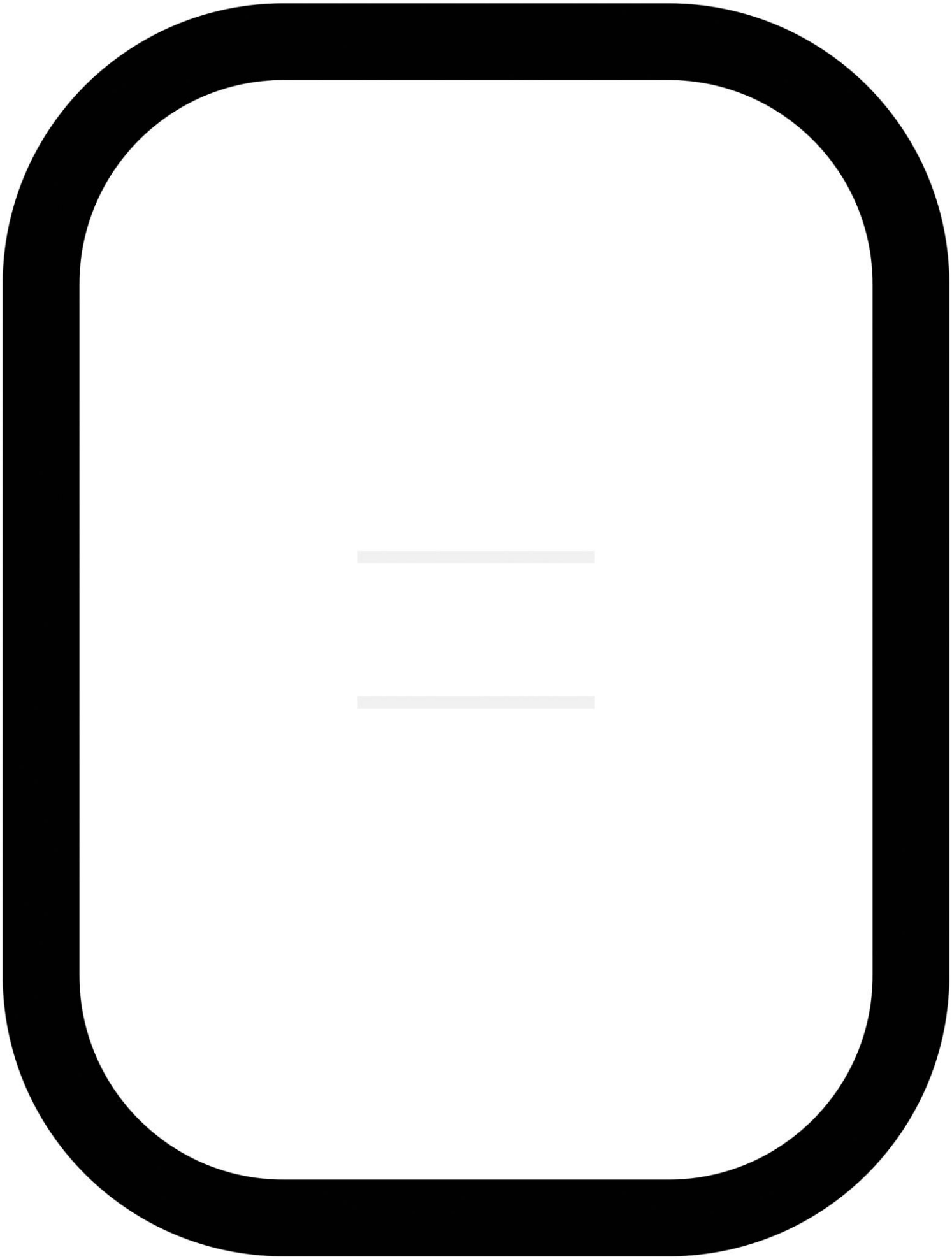
_____

_____

Activity sheet U1-1: Line-tracking border



Activity sheet U1-2: Sumo ring



Activity sheet U1-3: TV remote controls



Drive forwards



Drive backwards



Spin left



Spin right



Turn left



Turn right

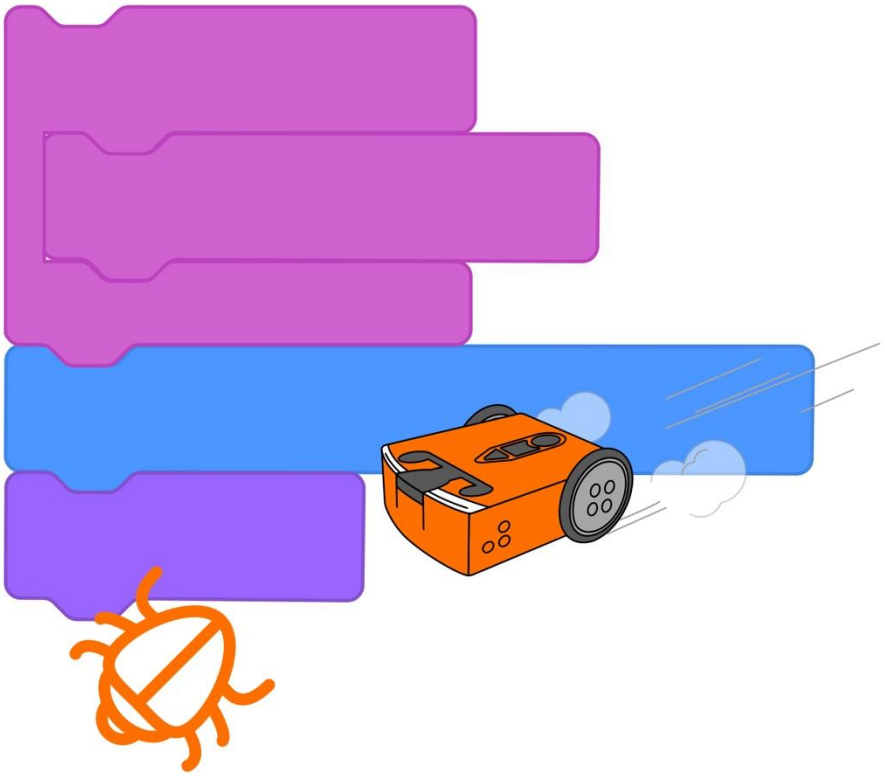


Play beep



Play tune

Unit 2: Move it!



Edison

Name_____

U2-1.1 Let's explore how computers 'think'

Imagine a computer. Now imagine a person. Which one do you think is smarter?

This is a bit of a trick question. Believe it or not, most computers cannot do anything without help from people.



Why is that?

Both computers and people can follow instructions, but people can also think for themselves. We can learn and change what we do based on new knowledge.

Most computers, however, cannot do that. An Edison robot, for example, cannot think for itself. It can only follow instructions. Where do those instructions come from? A person like you!

People give computers instructions by giving them computer programs.

To make a good computer program for our Edison robot, or any computer, we need to write that program in a way that the computer can understand. To do this, we need to try to think about things as if we were a computer.

This kind of thinking is called **computational thinking**.



Don't forget

A computer program is a collection of instructions that tell a computer to perform a specific task.



Jargon buster

Computational thinking means thinking about a problem or task similar to how a computer thinks. It is a way of logically working through problems, breaking them down into smaller pieces, finding patterns, and then using the information to come up with a step-by-step solution.

In other words, computational thinking is a way of planning, problem-solving and analysing information the same way a computer does.

Whenever you want to write a program for your Edison robot, you need to use computational thinking to help you work out what to do. By learning to think in a way that will make sense to Edison, you will be able to give the robot instructions to get it to do what you want.

One of the most important things about giving instructions to Edison is the order in which you give the instructions.

Name_____

The importance of going step-by-step

Computers, including Edison robots, are very good at following the instructions that we give them as computer programs. In fact, an Edison robot will follow the instructions in a program *exactly* as they are written. That's why one of the most important parts of computational thinking is using **sequence**.



Jargon buster

Sequence means going in order, step-by-step.

Imagine you want to bake a cake. You might look up a recipe in a cookbook. To make the cake, you would then follow each step one by one. That's sequence!

Whenever you write a program for Edison, you will need to use sequence in the same way. You need to tell Edison exactly what to do, in the exact order you want the robot to do each step.

Task 1: Follow step-by-step

If your teacher tells you to go to the door, what actions do you have to take to get there? You probably don't think about how many steps you need to take. You just do it! If there is a desk in your way, you simply turn and walk around it.

That's not how a robot works. To get your robot to the door, you would need to give it very careful instructions with each step explained one by one. In other words, you would need to tell the robot each action you want it to take in sequence.

Thinking about doing something sequentially like this takes some practice. People are so good at 'just doing' things, we don't usually think about what it is that we are doing broken out into each and every step.

Try following some exact step-by-step directions to see how it feels. Use activity sheet U2-1 to answer the following questions.

1. Start on the ice cream cone, pointing towards the heart. Turn right. Move forwards 2 squares. Where are you?

   _____

2. Start on the panda bear, pointing towards the bicycle. Move backwards 1 square. Turn left. Move forward 2 squares. Turn right. Move forwards 1 square. Where are you?

   _____

3. Start on the star, pointing towards the cat. Turn left. Turn left again. Move backwards 2 squares. Turn right. Move forward 1 square. Turn right. Move forwards 1 square. Turn left. Move backwards 2 squares. Where are you?

   _____

Name_____

Task 2: Give step-by-step instructions

Let's practice giving exact instructions, describing each item step-by-step. Use activity sheet U2-1 to answer the following questions.



Hint!

Use these commands to write your answers:

move forwards *move backwards* *turn left* *turn right*

4. Write directions for this: start on the rainbow, pointing towards the dog. End on the bird.

_____

_____

_____

5. Write directions for this: start on the rainbow, pointing towards the dog. Do NOT touch the dog. Do NOT touch the cat. End on the bird.

_____

_____

_____

6. Write directions for this: start on the diamond, pointing towards the beachball. Do NOT use any 'move forwards' commands. End on the beachball.

_____

_____

_____

Name_____

U2-1.1a Change it up: Make a PBJ sandwich

Do you know how to make a peanut butter and jam sandwich? Could you teach a robot?

There are lots of things that you know how to do so well that the task seems very easy. These 'easy tasks' can actually be made up of lots of individual steps, however. Making a peanut butter and jam sandwich is a good example. To make the sandwich, you have to do many different things, and you have to do each one in the right **sequence** to end up with a sandwich.



Jargon buster

Sequence means going in order, step-by-step.

Task 1: Write down how to make a sandwich step-by-step

1. To make a sandwich, what do you need to do? Write down each step sequentially.

Task 2: Follow the directions

Your teacher will help you set up for this task.

Make a sandwich following the directions that have been written down. Be sure you follow the directions EXACTLY as they are written. Don't add any extra steps!

2. How did your sandwich turn out?

Name_____

U2-1.1b Change it up: Human robots

There are lots of things you know how to do that you think are easy but, actually, have lots of individual steps. Robots cannot lump individual steps together very well. Instead, robots need to have clear instructions that lay out every single move step-by-step. Explaining what needs to happen with clear instructions which are in **sequence** is important to get a robot to do what you want.



Jargon buster

Sequence means going in order, step-by-step.

Task 1: Write down the step-by-step instructions

Your teacher will help you set up for this task.

1. To get the human robot to the goal, what do you need to do? Write down each step sequentially.

_____

_____

_____

_____

_____

_____

_____

_____

_____

_____

_____

_____

_____

Task 2: Follow the directions

Your teacher will help you set up for this task. Be sure you follow the directions EXACTLY as they are written. Don't add any extra steps!

2. Did the human robot reach the goal? Did you need to change anything in your directions to get the instructions to work?

_____

_____

Name_____

U2-1.2 Let's explore going step-by-step in EdScratch

When you write a program for your Edison robot in EdScratch, you are writing the instructions that will tell the robot what to do and in what order to do each thing. Each EdScratch block is one action you are telling the robot to take.

The order in which you connect the blocks together in your program tells the robot in what sequence to do each action. Edison will do the actions in order, one at a time, starting with the top block.

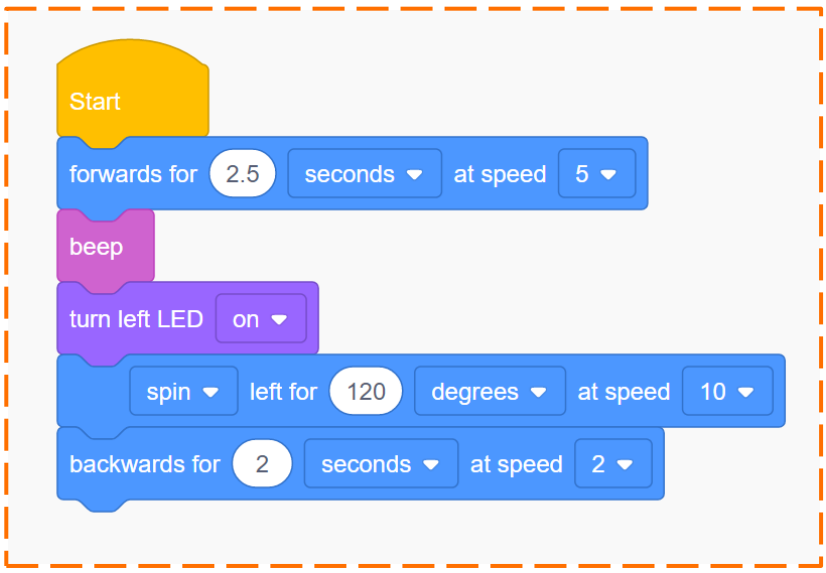


Don't forget

Sequence means going in order, step-by-step.

Task 1: What will Edison do?

Look at this EdScratch program:



All EdScratch programs need to have the yellow Start block at the very top. This lets the robot know that the program starts with the first block below the Start block.

Read each line block by block, starting with the top block below the Start block.

1. If you download this program to Edison, what will the robot do? Be sure to write your answer in the correct sequence.

_____

_____

_____

_____

_____

Name_____

Task 2: Write the program

Go to the EdScratch app on your computer and write the program from the picture. Find each block in the block pallet and drag it into the programming area.



Hint!

You can change the numbers in a block by clicking on the number and changing it using your keyboard.

You can change a drop-down item in a block by clicking on the down arrow and selecting the option you want.

Make sure that you put all the blocks in the correct sequence in your program.

Once you have written your program, download it to your Edison robot. Run the program and watch what Edison does.

Look at what you wrote down about what you thought the robot would do in your answer in task one. Compare what Edison does when you run the program to your answer. Does the robot do what you predicted? If not, what is different? Check your program and your answer to see if you can spot, and fix, the difference.

Task 3: Change the sequence

Try changing the sequence of your program. Change the order of at least three of the blocks in your program. Don't add any more blocks or change any of the options inside a block – only change the sequence.

Download your new program to Edison and run it in your robot.

2. Which blocks did you move to change the program's sequence? Write down your new program.

_____

_____

_____

_____

_____

Name_____

U2-1.3 Let's explore driving Edison

One of the groups of blocks in the EdScratch block pallet is the Drive category. All the blocks in this category relate to using Edison's motors. One of the things you can use the motors to do is drive the robot like a car.

No driver needed! Just a programmer

How do you 'drive' an Edison robot? By **programming** the robot with **code**!



Jargon buster

Programs are the sets of instructions you create for a computer, or an Edison robot, to follow. The stuff inside a program is sometimes called **code**.

People often use the words **code** and **program** to mean the same thing. For example, you could say 'write a program for Edison' or 'write some code for Edison'. Either way it means 'write commands to instruct the Edison robot what to do using a programming language it understands.'

When you use EdScratch to create a set of instructions for Edison, you can say you are **programming** or that you are **coding** or both. Which makes you a **programmer** and a **coder**!

Let's try programming Edison using drive blocks.

Task 1: Drive a straight track

For this task, you need to get Edison to drive a straight track. Use activity sheet U2-2. You need to write a program so that Edison can drive the track. Start Edison on the outline and have the robot stop after it crosses the 'finish' line.

Go to the EdScratch app on your computer. Look at the blocks in the Drive category. Which blocks will you need to write your program? Test your program by downloading it to your Edison robot and running it using the activity sheet. Did it work?



Don't forget

You can change the numbers in a block by clicking on the number and changing it using your keyboard.

You can change a drop-down item in a block by clicking on the down arrow and selecting the option you want.

Name_____

1. You can code a solution for this task using just one block! Which block will work? Fill out the block below to match what you used in your successful one-block program.



Task 2: Drive a maze

For this task, you need to get Edison to drive through a maze. Look at the maze on activity sheet U2-3. Think about the different actions Edison will need to take to drive the maze. Be sure to consider the sequence of the actions too!

2. What actions do you think Edison will need to take to complete the maze? Write down a plan to get Edison through the maze.

_____

_____

_____

_____

Use this plan to help you write a program in EdScratch for Edison to drive through the maze. You will need to use several different blocks in your program to be able to complete the maze. You will also need to figure out what **input parameters** to use in each block.



Jargon buster

The things you can change inside a block, like the numbers and choices in the drop-down lists, are called **input parameters**.

Start Edison on the outline in the maze and have the robot stop after it crosses the 'finish' line. Be sure to have Edison stay inside the lines all through the maze - no cheating!



Hint!

Your program might not work the first time – and that's okay! Part of coding is experimenting and problem solving. If your program isn't working, think about the actions you need Edison to do one by one to complete the maze.

Are you missing any actions in your program? Are there any actions out of order?

You can also try changing some your input parameters. Sometimes changing an input parameter even just a little bit makes a big difference!

Name_____

U2-1.3a Challenge up: Maze madness

Did you get Edison to drive the maze on activity sheet U2-3 successfully? Try these other maze challenges! Remember to have Edison stay inside the lines all through the maze - no cheating!

Mirror track

Complete the maze by starting from the 'finish' line and driving to the start.

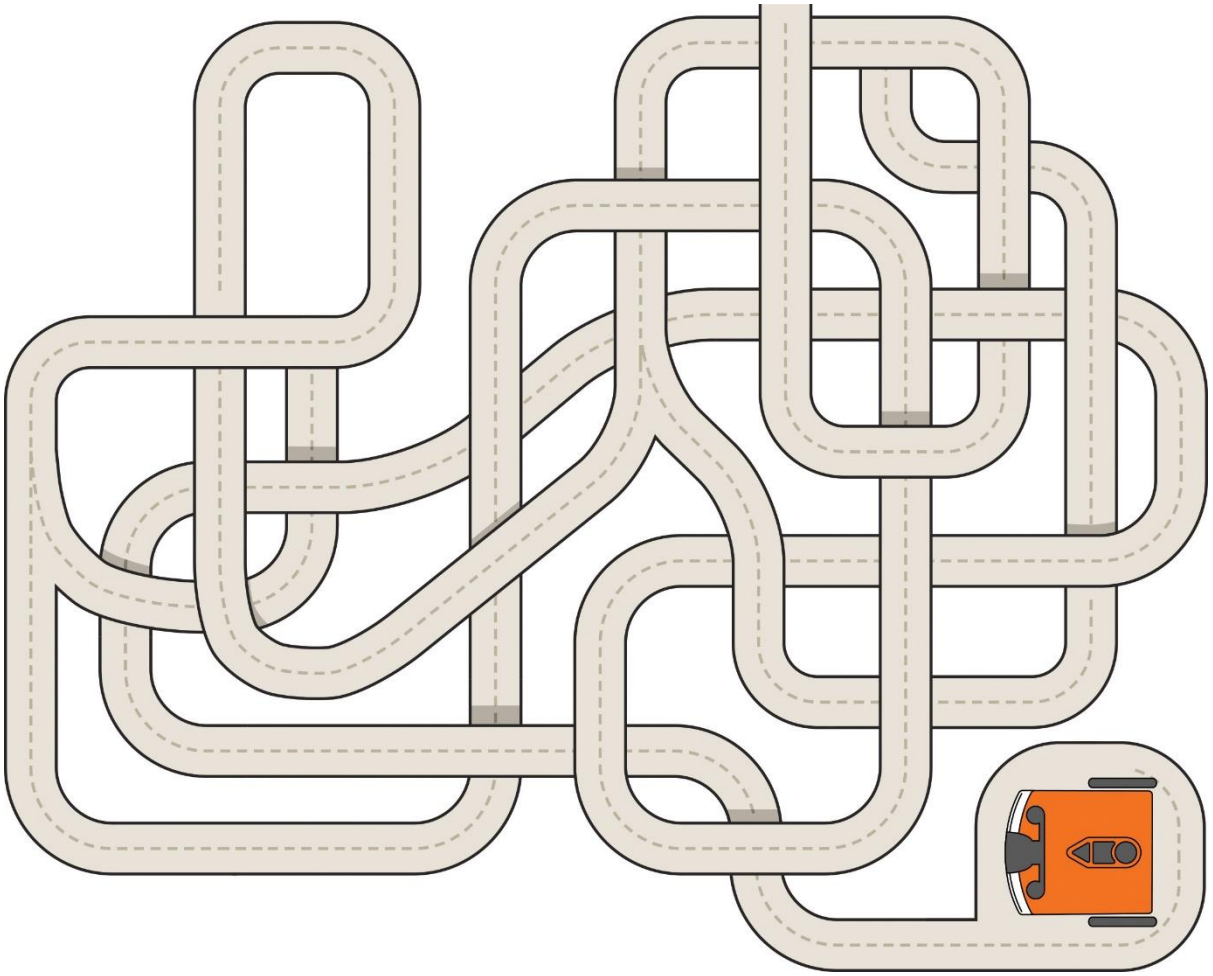
Backwards bot

Complete the maze by driving backwards from the beginning of the maze all the way to the end.

Make your own maze

Create your own maze for Edison to drive through, then write an EdScratch program for Edison to be able to complete your maze.

Once you solve your maze, swap with a partner. While they work out a solution to get Edison through your maze, you need to figure out how to get Edison to drive through their maze!



Name_____

U2-1.3b Challenge up: Self-walking pet

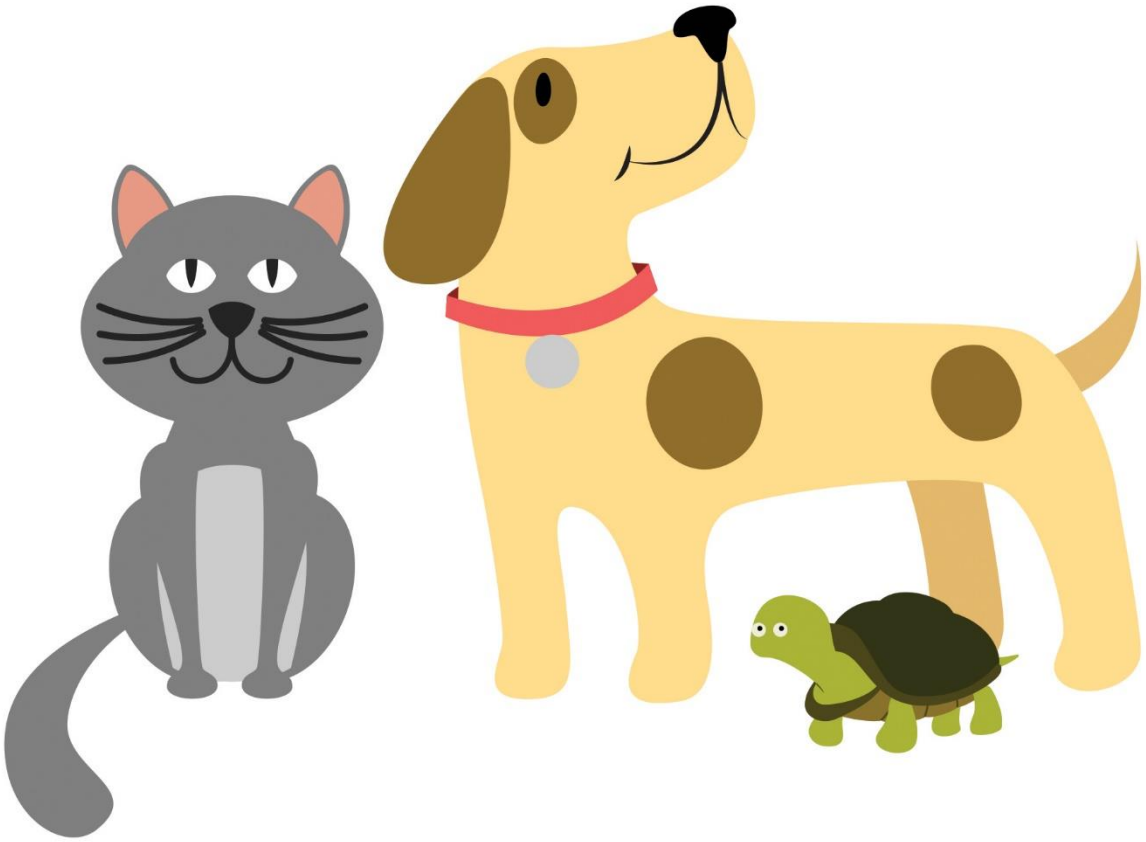
Having a pet means taking care of that pet. You have to feed it, clean up after it, and make sure it gets exercise. If only there were an easier way…

In this activity, try turning Edison into an animal that can take itself for a walk.

How can you make Edison into a self-walking pet? Here are a few ideas:

- You could decorate Edison to turn the robot into a pet.
- You could build a pet that attaches to Edison.
- You could make something that goes on the outside of Edison which Edison can move around.

Create your pet. Then write a program so that your pet will go for a walk and come back to you!



Name_____

U2-2.1 Let's explore Edison's outputs

What do computers do? Computers **process** information. This means that computers take information from somewhere and do something with that information. For example, you can give a computer two numbers and tell it to add them together. The computer can then add those numbers and show you the result.

This cycle of information coming in, the computer doing something with the information and then creating some result is called the **input-process-output cycle**.



Jargon buster

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

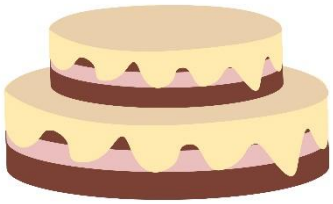
A **process** is what the computer does with a computer program full of information and instructions. This is sometimes called 'running' the program.

Outputs are the results you get from a computer. What the computer displays, or how the robot behaves, are the outputs you get based on the information and instructions you gave the computer.

We call this process of **inputs** going into a computer, that computer **processing** the information, and then generating some type of **output** the **input-process-output cycle**.

The input-process-output cycle isn't only used in computers. You can see this cycle in action in your daily life too.

Baking a cake is a good example. You **input** ingredients into a pan and put that pan into the oven. The **process** of baking then happens in the oven. After a while, the **output** of a cake is ready!



Inputs, outputs and Edison

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.

Your Edison robot has three main types of outputs: outputs using the motors, outputs using the LEDs and outputs using sounds. In EdScratch, the blocks related to Edison's main outputs are organised into three different categories: Drive, LEDs and Sound.

Name_____

1. Look at the blocks in the Drive, LEDs and Sound categories in EdScratch. Which category contains the code blocks you would need to input into a program to get Edison to generate each output? Match each output to the category where you can find the blocks you would need.

Output	Category
 Turn Edison's lights off ☐	● Drive
 Play a musical note ☐	● LEDs
 Spin Edison right ☐	● Sound

Task 1: Flash and beep

Edison has two red LED lights. When you turn Edison on, you can see these two LEDs begin to flash.

Edison also has a special bit of tech which you can see just to the left side of the round button on the top of the robot. This is a buzzer and a sound sensor all in one. It can detect noise, but it can also make noise!

For this task, you need to write a program which will have Edison use both the LED and buzzer outputs. Write a program in EdScratch **using eight blocks** which tells Edison to do the following things in sequence:

- ☐ turn on the left LED light
- ☐ beep
- ☐ turn off the left LED light
- ☐ beep
- ☐ Turn on the right LED light
- ☐ beep
- ☐ Turn off the right LED light
- ☐ beep

Download and test your program with your Edison robot.

2. Did the program work the way you expected? Could you observe the robot perform each step of the program?

_____

_____

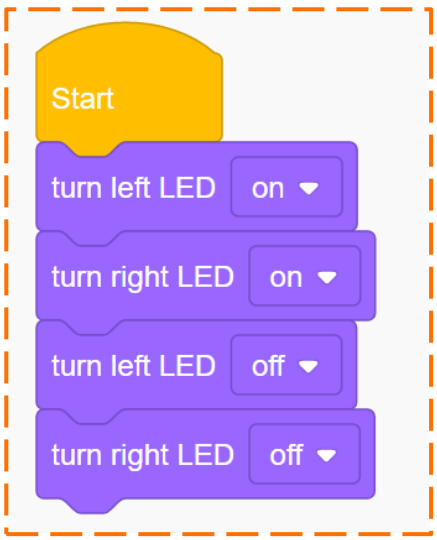
_____

Name_____

Task 2: Make Edison blink

Some of Edison's outputs, like turning a LED on or off, happen very quickly. In fact, this can happen so fast, it can be really hard to see.

Try writing the following program in EdScratch:



Download it and run it with your robot. Can you see Edison blink?

Because the robot flashes its LEDs when it is in standby mode, it will be really hard to see this program in action.



Why is that?

Edison moves through each EdScratch block one at a time, but the robot is able to process each block very quickly. Computers can process information very fast – that's one of the things that makes them so useful!

If you want Edison to pause after it completes one block before going onto the next block, you have to tell it that.

One of the EdScratch block categories is the Control category. The blocks in the Control category allow you to control the flow of your program. One of the blocks in this category is the wait block:



Name_____

This block tells Edison to wait the amount of time you specify before moving on to the next block. Let's try using this block in a program.

Modify the 'blink' program from before but use this new control block to make the program work better. You want a program where it's very easy to see Edison 'blink'. Experiment using at least one wait block. Test using wait blocks in different places in your program to see what works best.

3. What does your program look like? Which blocks does it use, in which order? Write your program below. Be sure to include the input parameters you used.

Mini challenge!

When a person blinks, what happens? Their eyes start open and then…

Look at your blink program. Can you adjust your program to make it more like a blink?

Name_____

U2-2.1a Challenge up: Drive the maze safely

You can write programs for Edison which tell your robot to use multiple types of outputs.



Don't forget

Your Edison robot has three main types of outputs: outputs using the motors, outputs using the LEDs and outputs using sounds. In EdScratch, the blocks related to Edison's main outputs are organised into three different categories: Drive, LEDs and Sound.

For this activity, you will need to write a program telling Edison to drive the maze on activity sheet U2-3. This time, however, Edison needs to be a very safe driver. On the road, drivers use their indicator lights and horn to alert other drivers. Edison can do these things too!

Drive the maze starting at the outline and driving forwards to the finish line. Your program should end after Edison crosses the finish line.

Your program needs to tell Edison to pause and use the LED lights to 'indicate' before making each turn in the maze. Make sure other drivers would be able to see the LEDs indicate!



Hint!

If you are going to turn left, which LED should you use to indicate? What about when you turn right?

Your program should also use the 'beep' block at least one time somewhere in the program.



Hint!

Pretend the 'beep' block is like a horn in a car. Where in your program might it make sense for Edison to beep?

Name_____

U2-2.2 Let's explore input parameters

Every block in EdScratch contains inputs which tell Edison what you want the robot to do. You have probably noticed that lots of the blocks also have **input parameters**.



Jargon buster

Input parameters are the things you can change inside a block, like the numbers and choices in the drop-down lists. You can think of input parameters as specific pieces of information needed in an input.

For example, if you want Edison to drive forward, you need to give the robot specific information about that command, such as how far to drive and at what speed.

There are three styles of input parameters in EdScratch:

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

Each input parameter in a block gives a different piece of information that Edison will need to be able 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. For example, if you want Edison to drive backwards, there are three questions you need to answer:

1. **Question:** How far do you want the robot to go?

   → **Answer:** distance input parameter

2. **Question:** What units are you using to measure distance?

   → **Answer:** distance units input parameter

3. **Question:** How fast do you want the robot to go?

   → **Answer:** speed input parameter

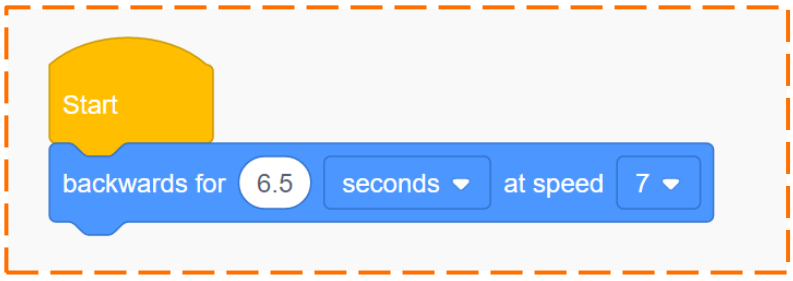
You always need to fill in all the input parameters in a block to give the robot all the information it needs.

Name_____

Try it out!

The input parameters in a program give you lots of information about what that program is asking the Edison robot to do. Try reading the following programs and answer the questions.

Look at this program:



1. In this program, what is the input parameter for **distance units**?

   _____

2. In this program, what is the input parameter for **speed**?

   _____

3. What is the full program telling Edison to do? Highlight all of the input parameters in your answer in another colour or by underlining or circling each one.

_____

_____

Look at this program:



4. How many input parameters are there in this program?

   _____

5. Remember that input parameters are answers to questions the robot needs to know. What question is the spin input parameter answering? *Hint:* you might want to look at the other options for this input parameter in EdScratch to help you think about your answer.

_____

_____

Name_____

U2-2.2a Change it up: Teach Edison to count to 9

Have you ever looked at how old digital signs and clocks display numbers? The numbers are displayed using lines which make up a rectangle-shaped grid. Each number, from 0 up to 9, can be displayed by using some combination of the lines in that grid.

Digital display numbers don't have curves or diagonal lines. They only have straight lines and right angles. That makes them perfect patterns for Edison to drive!

Task 1: Teach Edison a number

Look at activity sheets U2-4, U2-5, U2-6 and U2-7. Choose one of the activity sheets to use.

Write a program for Edison so that the robot can trace over the digital display number you chose. Start the robot off of the digital display number and drive so that the robot traces over every segment of the number.

1. Which digital display number did you use?

   _____

2. What does your program look like? Which blocks does it use, in which order? Write your program below. Be sure to include the input parameters you used.

Name_____

3. Look at the distance input parameter in the blocks in your program. What do you notice about the inputs you used? How could this help you plan out a program for a different digital display number?

_____

_____

_____

Task 2: Teach Edison a different number

Choose a different digital display number activity sheet. Use what you learned about the distance input parameter from your last number and write a program for Edison so that the robot can trace over your new digital display number.

4. Which digital display number did you use?

   _____

5. What does your program look like? Which blocks does it use, in which order? Write your program below. Be sure to include the input parameters you used.

Name_____

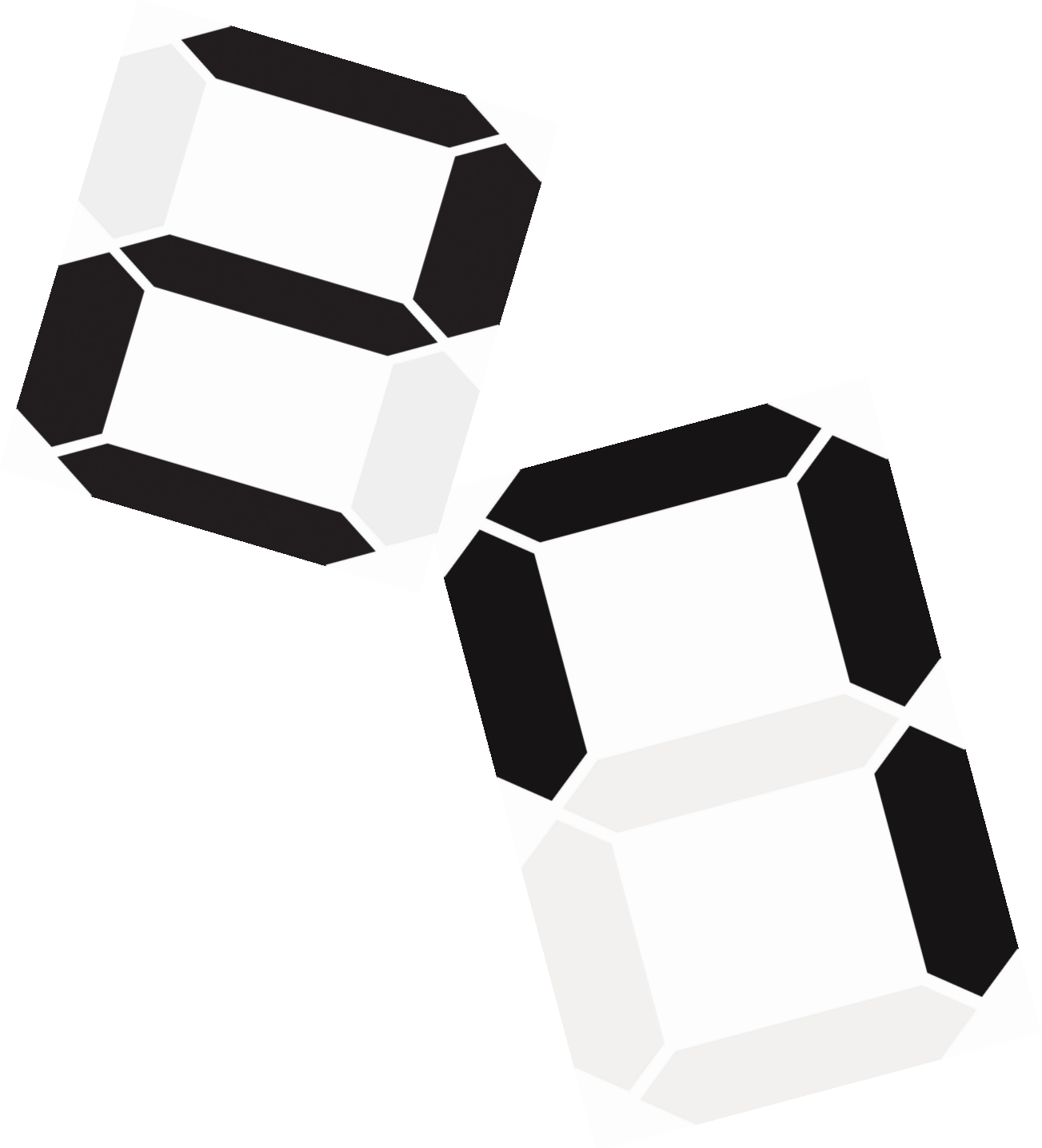
6. Compare your two programs. Are there any patterns you notice that are similar in both? What are they?

_____

_____

_____

_____



Name_____

U2-2.2b Challenge up: Teach Edison to count to 9 out loud

Can you get Edison to trace a digital display number by driving over it and count the same value 'out loud' in just one program?

What to do

Look at activity sheets U2-4, U2-5, U2-6 and U2-7. Choose one of the activity sheets to use.

Write a program for Edison so that the robot traces over the digital display number you chose. You also need Edison to count 'out loud' somehow.

Your program needs to have Edison count the same amount as the digital display number it is driving. For example, if you choose the number 5, your program needs to have Edison give some sort of signal as it 'counts' to 5.



Hint!

Which of Edison's outputs could you use to signal the robot is counting?

Think about the sequence of things you want Edison to do. Will the robot drive the whole path and then count? Count before driving? Drive a little, count to one, then drive a bit more before counting the next number?

How you do it is up to you!

Mini challenge!

What about the rest of the numbers? Make your own digital display number with a different number than the activity sheets.



Don't forget

Digital display numbers don't have curves or diagonal lines. They only have straight lines and right angles. The numbers are displayed using segments which make up a rectangular grid. Each number, from 0 up to 9, can be displayed by using some combination of the lines in the grid.

Once you have made your digital number, test it out! Write a program for Edison to trace and 'count' your number.

Name_____

U2-2.3 Let's explore Edison's musical talents

Outputs using sounds are one of your Edison robot's three main types of outputs. In EdScratch, the blocks related to sound outputs are in the Sound category.



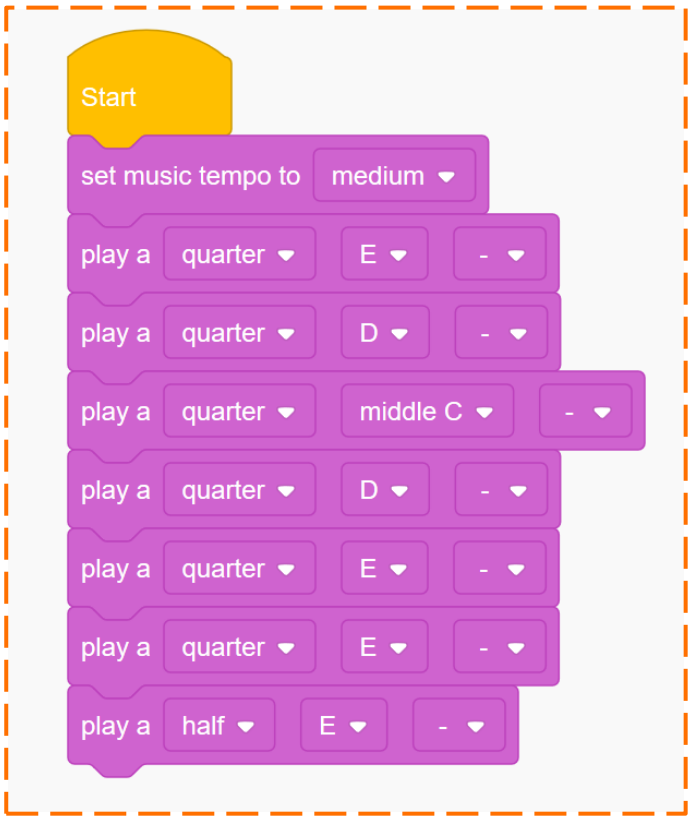
Don't forget

Edison has a special bit of tech which you can see just to the left side of the round button. This is a buzzer and a sound sensor all in one. It can detect noise but it can also make noise, like beeps or musical notes.

Have a look at the Sound category in EdScratch. There are not many blocks in this category, so you might think there is not a lot you can do with Edison and sounds. By using the sound blocks in different sequential orders and with different input parameters, however, you can play whole songs!

Task 1: Play a tune

In EdScratch, write the following program:



Download the program to your Edison robot and run it. This program is the first part of a song you might know. Do you recognise the tune?

Name_____

The first block in the program is the set music tempo block. Try clicking on the drop-down menu in the set music tempo block to see the input parameter options for the block. Choose a different input parameter for the block, then download the adjusted program to Edison. Run your new program in your robot.

1. Which new input parameter did you select for the set music tempo block?

   _____

2. What happened when you played the new program? What changed compared to the original program?

_____

_____

_____

3. What does the set music tempo block do?

_____

_____

_____

4. If you put the set music tempo block at the end of this program instead of the beginning, what would happen? *Hint:* Remember that Edison will read each EdScratch block one at a time from the top of the program in sequence.

_____

_____

_____



Hint!

Not sure what would happen if you make a change to a program? One of the best ways to find out is to make the change, download the adjusted program and test it out using Edison!

You can always experiment in coding!



Use this link

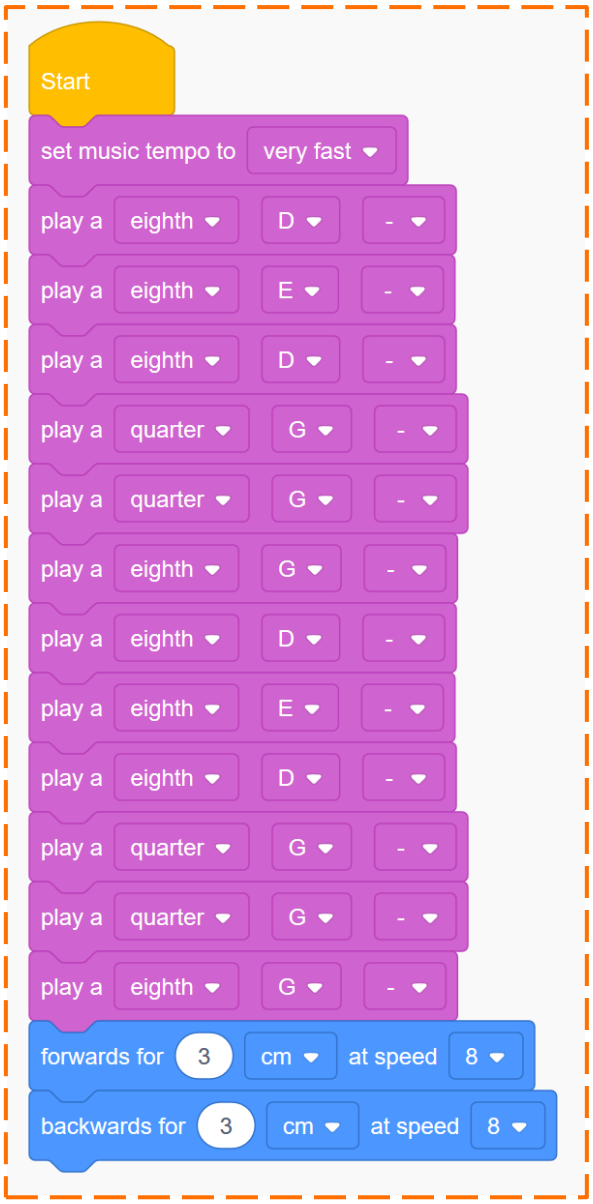
Want to hear Edison play more of this song?

Open the program using this link: https://www.edscratchapp.com?share=Eb12x3Dm

Name_____

Task 2: Move to the music

Look at this EdScratch program:



This program has the music for the first part of the song *The Hokey-Pokey*. This is a great song to dance along to – the song tells you just what moves to do!

In EdScratch, write the Hokey-Pokey dance program. Download the program to your Edison robot and play it.

5. Did your Edison robot move along with the music? Why do you think this is the case?

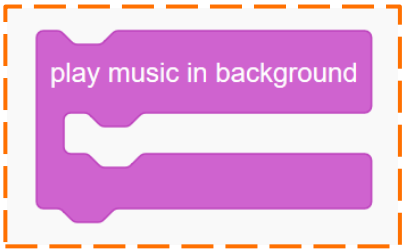
_____

_____

_____

Name_____

Edison can play a musical note while doing something else, like driving, but you have to tell the robot that is what you want it to do. There is a special block in the Sound category in EdScratch you need to use to do this. Here is what it looks like:



The play music in background block is a grouping block. Other blocks can sit inside this grouping block.



Why is that?

The shape of a block in EdScratch can give you a hint about what that block is used for in programs. Look at the play music in background block. See how it has a shape a bit like a mouth? Other blocks can sit inside the opening of this block's 'mouth'.

Any block that sits inside the play music in background block will be affected by this grouping block. Remember, Edison will follow each EdScratch block one at a time. The robot will see the grouping block first and know that any blocks inside that block get the 'play music in background' affect.

Add a play music in background block to your Hokey-Pokey dance program. Think about where in the program this block needs to go.

Download the adjusted program to your Edison robot and play it. Edison should now start playing the song and move at the same time!

Edison's dance moves need a bit of work, however.



Hint!

Remember that if something isn't quite right in a program, a warning message will show up in the bug box. These messages can help you work out what should be inside the grouping block and what should not!

Task 3: Dance along

Add more dance moves to your Hokey-Pokey dance program. You could get Edison to follow the steps in the Hokey-Pokey song lyrics or make up your own dance!

Can you get Edison to dance in time to the music?

Name_____

U2-2.3a Change it up: Play a song in a round

For this activity, you need to use multiple Edison robots together to play a song in a round.



Why is that?

A round is a musical piece where two or more people (or robots!) sing or play the exact same melody, but each begins at a different time. While every participant is singing or playing a different part of the song, the melody still harmonises them together!

Work in a group to turn your Edison robots into harmonising musical stars!

What to do

The first thing you need to do is choose a song for the robots to play. Many nursery rhyme songs work well in a round. One example is the song *Row, Row, Row Your Boat*.

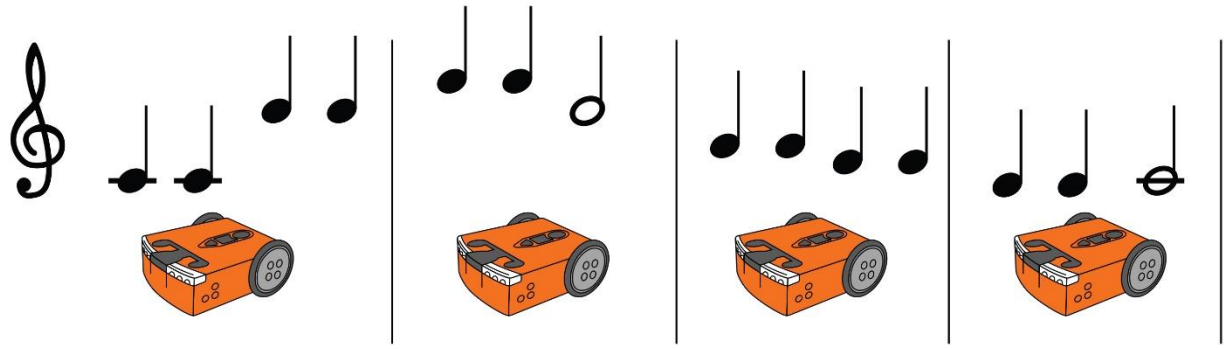
Once you choose your song, you will need to program each robot to play it. Each robot's program will be a little bit different as you need each robot to wait until the right time to join in the round.

Test out your programs with your robots. Then put on a musical show!



Hint!

Which Control block tells Edison to wait?



Name_____

U2-2.3b Challenge up: You are the conductor

What is your favourite song? Is there a theme song to a TV show or a movie you love to hear?

For this activity, the song choice is up to you!

What to do

You are the conductor, and you need to get Edison to play the song of your choice. Write a program for Edison so that the robot plays your song. Test out different tempos to see which works best.

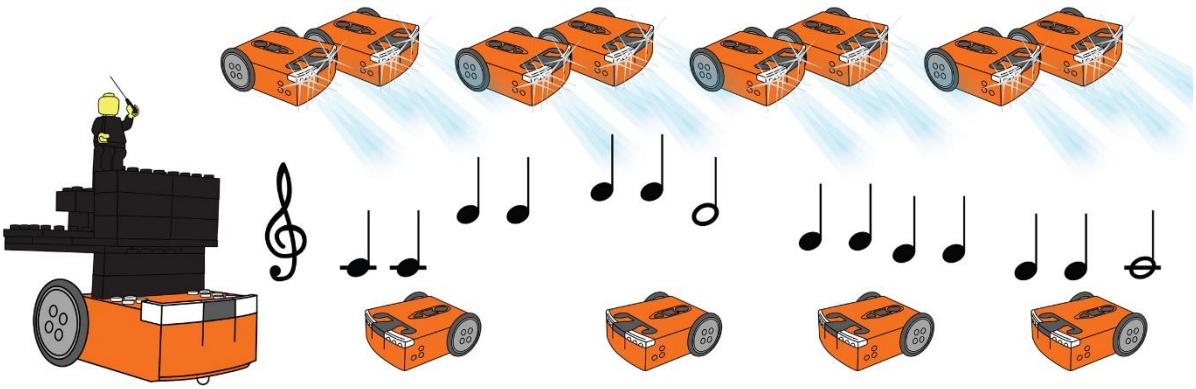
If you want, you can also program the robot to move or dance to your song.

Once you have your program ready, run it in Edison to have the robot perform your musical masterpiece!



Hint!

Which block tells Edison to play music in the background?



Name_____

U2-2.4 Let's explore bugs and debugging

Do you want to know a secret about working with computers and writing code? Here it is:

SOMETHING ALWAYS GOES WRONG!

Okay, so that isn't actually a secret, but it is really important. A major part of computational thinking and working with computers is problem-solving.



Don't forget

Computational thinking means thinking about a problem or task in a way that is similar to how a computer thinks. It is a way of planning, problem-solving and analysing information the same way a computer does.

When things don't work the way you want, remember that is okay! It just means it is time to problem-solve. One of the main types of problem-solving you do in computer science is finding and fixing **bugs**.

Debugging is a major part of coding and using robots. People who work as computer scientists, computer programmers or roboticists professionally spend a lot of time debugging. Probably even more time than they do writing their code!



Jargon buster

When something isn't working in a computer program, it is called a **bug**.

Finding and fixing bugs in a computer program is called **debugging**.



Why is that?

Calling something that's going wrong a 'bug' might seem a bit strange, but that's really what these errors are called.

Why are computer problems called bugs? A woman named Grace Murray Hopper, who is one of the inventors of modern computer programming, came up with this term. She once discovered that the issue causing her computer to malfunction was an actual moth that had gotten into the hardware! She fixed the problem by literally debugging her computer.

The name stuck and now problems with software or hardware in computers are called bugs and fixing them is called debugging!

Name_____

Debugging in EdScratch

The bug box in the EdScratch programming environment is a special feature to help you find and fix any bugs in your code.



Don't forget

The bug box is the area below the block pallet and programming area in EdScratch. If there is a bug or if it seems like something isn't quite right in a program, a warning message will show up in the bug box.

The warning messages that appear in the bug box are there to give you information about any problems or potential problems in your code. These messages are EdScratch's way of saying that there is an error in your code, or, that you might find an error when you run your program in Edison. In coding, there are two main types of errors: syntax errors and logical errors.



Jargon buster

Syntax is the rules of how a programming language works. All languages have rules. For languages people speak, like English or Chinese, there are rules about spelling and grammar and how to write the letters or characters in that language. Syntax is the same thing but for a computer language.

Syntax errors are caused by problems in how you wrote your code which break the rules of the language. These errors are sort of like typos or spelling mistakes in writing.

Logic is an organised way of thinking that makes sense to a computer. Logic determines the flow of a program, how you order things inside a program and what inputs you use to generate the outputs you want.

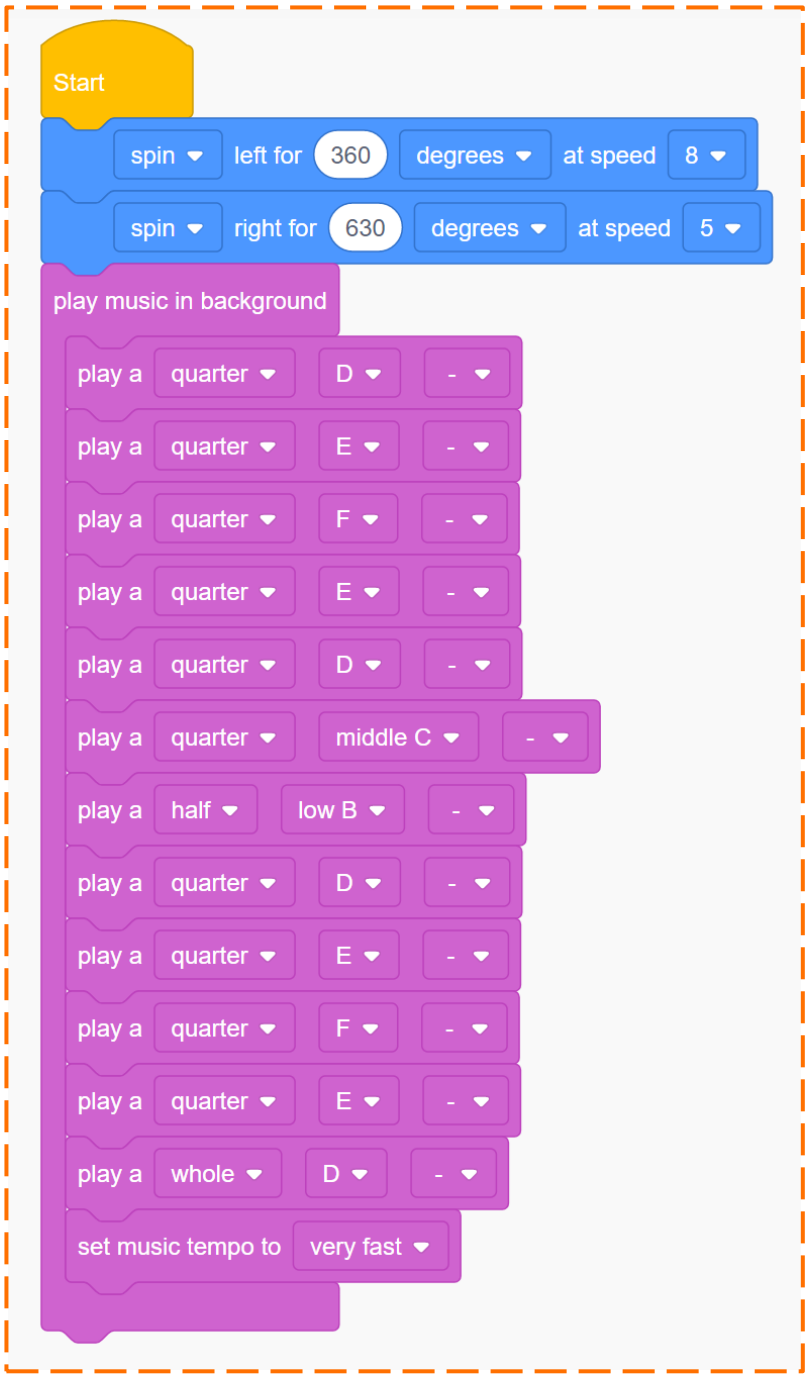
Logical errors are problems with the logic of a program. Logical errors are basically problems with the way of thinking in a program. Programs that do not make sense to the computer and programs that ask a computer to do something that it cannot do are examples of logical errors. If a program works differently than you expected it to work, there is a good chance that there's a logical error somewhere.

Understanding if a bug is the result of a syntax error or a logical error can help you fix the problem. If there is a syntax error in your EdScratch program, you will get a red warning message in the bug box, and you won't be able to download the program to Edison. If you can download your program and run it in Edison, but it doesn't do what you want, that probably means there is a logical error.

Name_____

Try it out!

Look at the following program:



This program is full of bugs. Your job is to fix it!



Hint!

One of the best ways to see what's going wrong – and right – with a program is to test it out using EdScratch and Edison!

Name_____

1. The programmer who wrote this program made two errors with the set music tempo block. One error is a syntax error, and one error is a logical error. Write an explanation of each of these two errors to explain them to the programmer. *Hint:* the programmer wants the music to play at a very fast tempo.

Syntax error: _____

_____

_____

Logical error: _____

_____

_____

This is what the programmer who wrote this code said about what the program is meant to do:

> "I want the Edison robot to play a tune very fast. While the music is playing, I want the robot to spin left for one full circle (360 degrees), then spin back right the same distance at the same speed."

2. The program does not work the way the programmer wants. You need to debug the program and get it working for the programmer. Test your program using Edison and keep debugging until the program works just the way the programmer explained. Describe the bugs you found and what you did to fix them.

_____

_____

_____

_____

_____

_____

_____

_____

_____

_____

Name_____

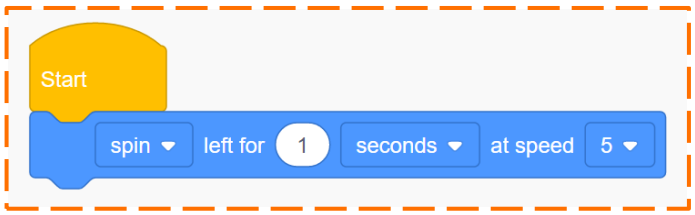
U2-2.5 Let's explore Edison's motors

Edison robots have two motors: one on the left side and one on the right side. Outputs using these motors are one of your Edison robot's three main types of outputs. In EdScratch, the blocks related to motor outputs are in the Drive category.

When you write a program for Edison using blocks from the Drive category in EdScratch, you are telling the motors what to do. Most of the blocks control both of Edison's motors. Does that mean that both the motors do the same thing?

Task 1: Spin that robot

If you wanted to write code to tell your robot to 'spin left' you can make a simple one-block program like this:



The input parameters in that block tell Edison the direction, the distance, the distance units and the speed you want the robot to use in the program.

The direction input parameter that has been selected is spin, which means the whole direction input is spin left. What is that input telling Edison's motors to output?

In EdScratch, write the program using the same input parameters as the one in the picture. Download the program and run it with Edison on the desk or floor.

Now run the program again, but this time hold Edison in your hands. Feel how the wheels are moving. What do you notice?

1. Which direction is the left wheel moving?

   _____

2. Which direction in the right wheel moving?

   _____

Edison's motors don't have to both do the same thing at the same time. Does that mean you could write a program moving just one of the motors? Can you write a program that tells each motor what to do separately?

Open the EdScratch app and look at the Drive category in the block pallet. Look at the different blocks and see if you can discover blocks you could use if you only wanted an output from one of Edison's motors.

Name_____

3. Which blocks do you think only use one of Edison's motors? Why do you think that?

4. You can use Edison to build and invent lots of different things. Imagine you need to create something using Edison which only uses one of Edison's motors. What could you build? How would your creation use the one motor?



Don't forget

The wheels of your Edison robot can be removed from the powered sockets they sit in. These sockets are what Edison's motors actually move.

Task 2: Direction = forward

To get Edison to work the way you want, you have to make sure you give the robot all the information it needs. If the robot doesn't have all the inputs and instructions it needs, the program probably won't work the way you want. This kind of logical error can be frustrating, especially if you think you have given the robot all the information necessary.

One of the main ways you give information to the robot using EdScratch is through input parameters.



Don't forget

Each input parameter in a block gives a different piece of information to Edison that the robot will need to be able to run that command. Input parameters are sort of like the answers to questions the robot has about what you are asking it to do.

Name_____

In EdScratch, some blocks get all the information they need from their own input parameters. Other blocks get information from inside their block, but also need information from somewhere else in the program as well.

Let's make a program for Edison to get the robot to move its motors. For this program to work, there are four questions you need to make sure your program answers:

Question 1: What direction do you want the robot to go?

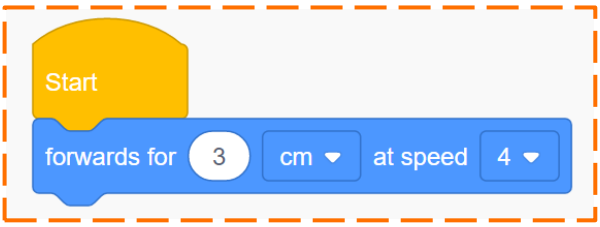
Question 2: How far do you want the robot to go?

Question 3: What units are you using to measure distance?

Question 4: How fast do you want the robot to go?

Your program needs to give the robot an answer to all of those questions.

Look at this program:



If you ran this program in an Edison robot, would the robot have all the information it needed to know what to do? In other words, does this program tell the robot the direction, distance, distance units and speed to move the motors?

5. Fill in this chart. If the information is in the program, write the value of that input in the 'value' column. For example, the value of 'distance' is the answer to the question, 'how far is this program telling the robot to go?'

Information	In the program?	Value
Direction		
Distance		
Distance units		
Speed		

Write the program in EdScratch, download it and run it in your robot.

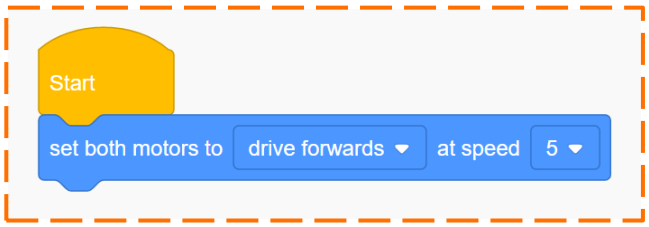
6. What did the robot do when you ran the program?

_____

_____

Name_____

Now look at this next program:



Does this program give the robot all the information it needs?

7. Fill in this chart. If the information is in the program, write the value of that input.

Information	In the program?	Value
Direction		
Distance		
Distance units		
Speed		

Write the program in EdScratch, download it and run it in your robot.

8. What did the robot do when you ran the program? Why did it behave that way?

_____

_____

_____

How can you give the robot the rest of the information it needs so that the robot moves its motors? Experiment in EdScratch to see if you can write a program that uses the set both motors block but no other blocks from the Drive category and gets the robot to move forward.



Hint!

Don't give up! Experimenting, testing and problem solving is how you learn new things in coding.

Feeling stuck? Think about what it is you are trying to do a different way. Look at different blocks in EdScratch and ask yourself, 'if I use this block, will it fix the problem I am trying to solve?'

If you aren't sure, try it and see what happens! Be sure to check the bug box for hints too!

Name_____

9. What did you try? Did it work? Write what you did, what worked, what didn't work and how experimenting made you feel.

Name_____

U2-2.5a Challenge up: Spinning garden

Edison's two motors each control a powered socket, one on the right side of the robot and one on the left. When you want the robot to drive, you attach wheels to the powered sockets using the wheels' axles. The motors turn the axles, which turns the wheels and allows Edison to drive.

How else can the powered sockets be used?

If you turn an Edison robot on its side, you won't be able to drive it like a car. Instead, you can use the robot to be the powered base for an invention!

What could you attach into the powered socket instead of a wheel? What will happen when the motor is turned on?



Don't forget

The wheels of your Edison robot can be removed from the powered sockets they sit in. These sockets are what Edison's motors actually move.

What to do

In this project, you need to use your Edison robots to help create a spinning garden. Work in a group to design a garden that uses Edison robots as the bases for plants, flowers, bees, birds or whatever else you would like to have in your spinning garden.



You can take the wheels off of the robots and use a different axle inside the powered socket or build using a wheel as a base. Each robot needs to have something created and attached to it which can spin in the garden.

Each robot will need to be programmed using EdScratch. Write and test your programs for each robot. You may need to make adjustments to your program depending on the type of object you are using and how you attach that object to the powered sockets of each robot.



Hint!

The set right motor and set left motor blocks are very helpful if you only want one motor to move. Don't forget you need another block, like a wait block, in the program to set the duration for the set motor blocks.

Name_____

U2-2.5b Challenge up: Spinning solar system

Edison's two motors each control a powered socket, one on the right side of the robot and one on the left. When you want the robot to drive, you attach wheels to the powered sockets using the wheels' axles. The motors turn the axles, which turns the wheels and allows Edison to drive.

How else can the powered sockets be used?

If you turn an Edison robot on its side, you won't be able to drive it like a car. Instead, you can use the robot to be the powered base for an invention!

What could you attach into the powered socket instead of a wheel? What will happen when the motor is turned on?



Don't forget

The wheels of your Edison robot can be removed from the powered sockets they sit in. These sockets are what Edison's motors actually move.

What to do

In this project, you need to use your Edison robots to help create a model of the solar system where the planets can spin. Work in a group to build your model. Decide how you will build the planets, if you will include moons, how big each solar object will be and how fast each one will spin. How accurate to the real solar system can you make your model?

You can take the wheels off of the robots and use a different axle inside the powered socket or build using a wheel as a base.

Each robot will need to be programmed using EdScratch. Write and test your programs for each robot. You may need to make adjustments to your program depending on the size of each object and how you attach that object to the powered sockets of each robot.



Hint!

The set right motor and set left motor blocks are very helpful if you only want one motor to move. Don't forget you need another block, like a wait block, in the program to set the duration for the set motor blocks.



Name_____

U2-2.5c Challenge up: Cartographer and navigator

A cartographer is a person that makes maps. A navigator is a person who figures out how to get from place to place. In this project, you need to be both!

What to do

The first thing to do in this project is to make a big map. You will use this map with your Edison robot, so it needs to be big enough to allow Edison to drive around on the map.

Decide what place your map will be about. Your map could be of your school, your town, a fictional city or a real place in the world where you want to travel. Whatever you choose, you will need to plan out your map and then make a version big enough for Edison robots to drive on.

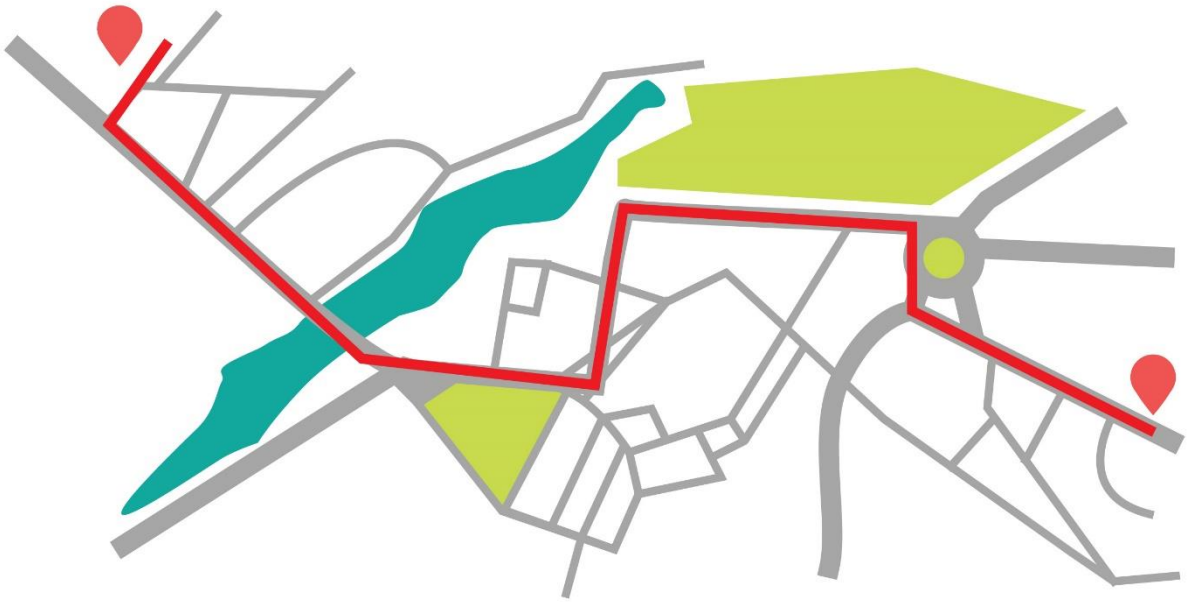
You also need to create programming challenges to be solved using your map. These challenges should tell the programmer where to start the Edison robot, where the programmer needs to have the robot finish, and any rules for the program. For example, you could have a challenge that says: *Start at the school. End at the ice cream shop. Don't go past the park.*



Hint!

Your program rules don't have to be just about where to go and where to avoid. You can also make rules about how Edison travels, such as going backwards or the speed Edison moves. Program rules that require the programmer to use blocks from the LEDs and Sound categories are good too!

Test the programming challenges to make sure a solution is possible for each one. Then trade challenges with a partner or another team. How many challenges can you solve?



Name_____

U2-2.5d Challenge up: Writer and director

Edison cannot speak, but that doesn't mean you cannot use the robot to help tell a story! In this project, you need to write a story, then 'direct' Edison to help tell the tale as if the robot were an actor in a play.

What to do

Write a story using a story map. Have Edison help you present this story. You will 'direct' Edison by programming the robot.

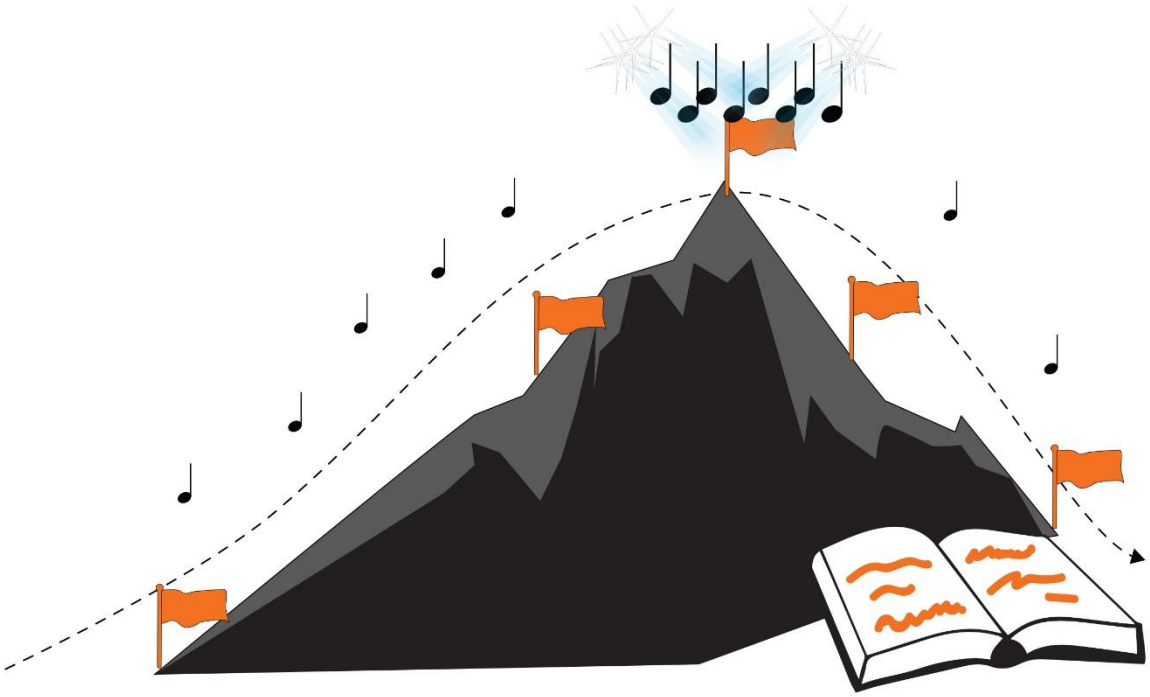


Hint!

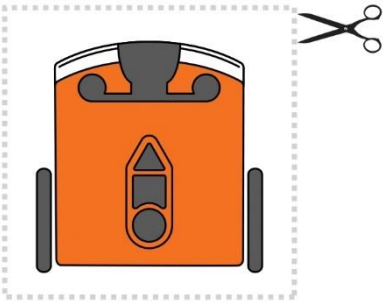
There are lots of ways you can do this project. You could make a story map that is big enough for Edison to drive on, performing actions at each stop on the story map. Or you could have Edison perform actions in time with you as you read the story out loud. Or maybe you can even turn this into a movie by filming Edison performing!

You can use any of Edison's outputs to help tell the story. Choose blocks from the Drive, LEDs and Sound categories. Don't forget about wait blocks! You can use wait blocks to help control the timing of Edison's actions!

Create a program for the robot to follow along and help make the story exciting by doing something at each major point in the story.

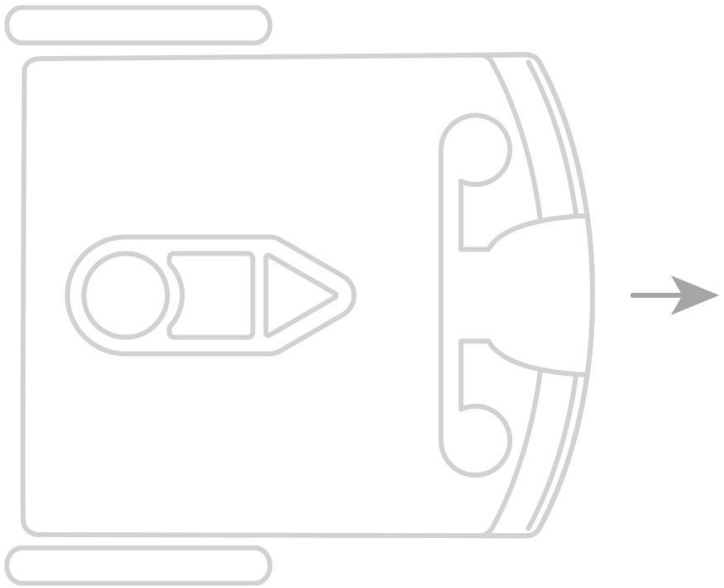


Activity sheet U2-1: Go step-by-step

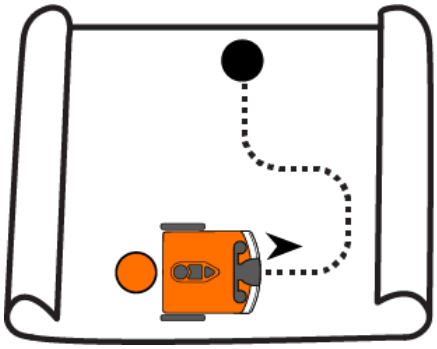


Activity sheet U2-2: Driving track



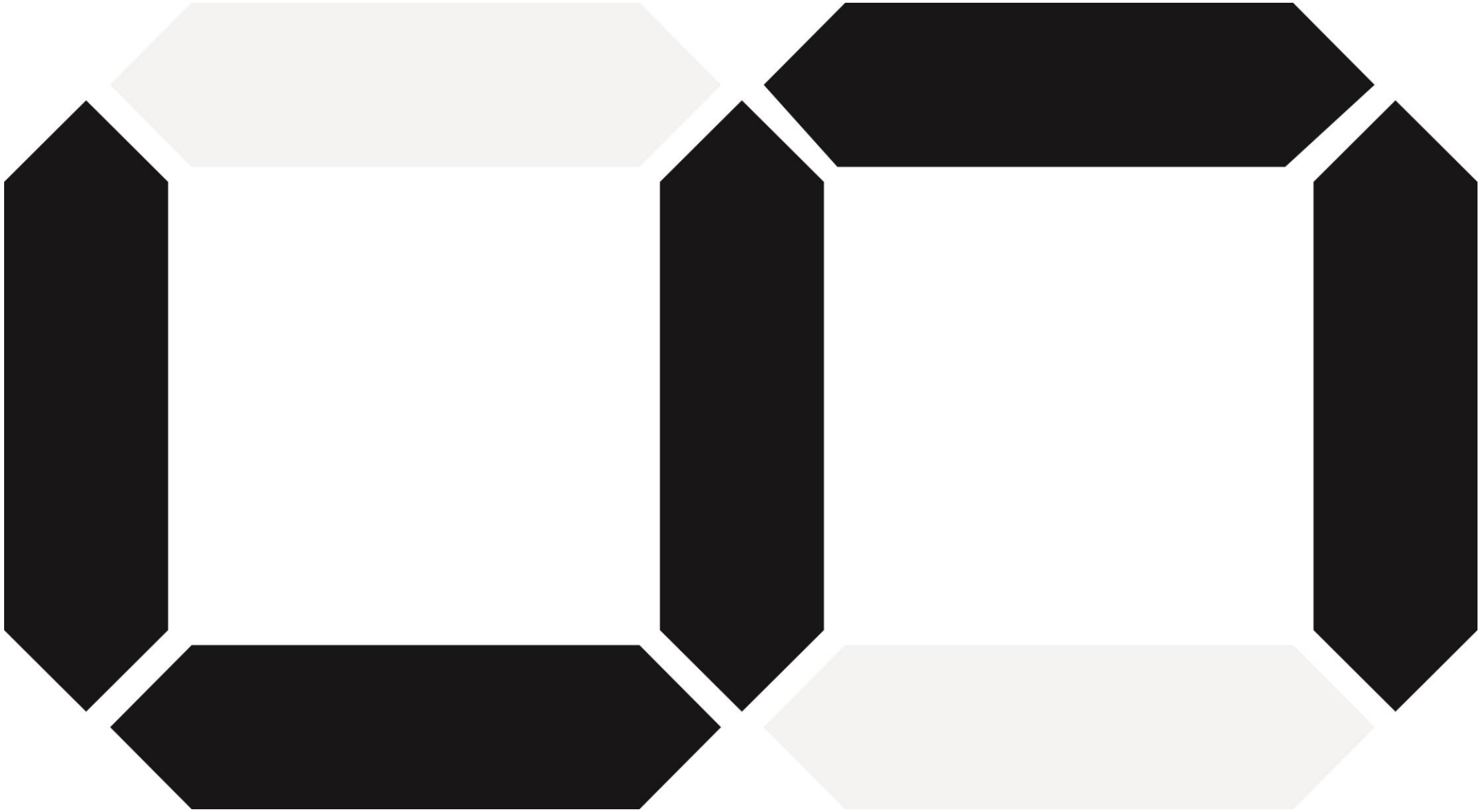
Activity sheet U2-3: Mini maze



Activity sheet U2-4: Digital display 2



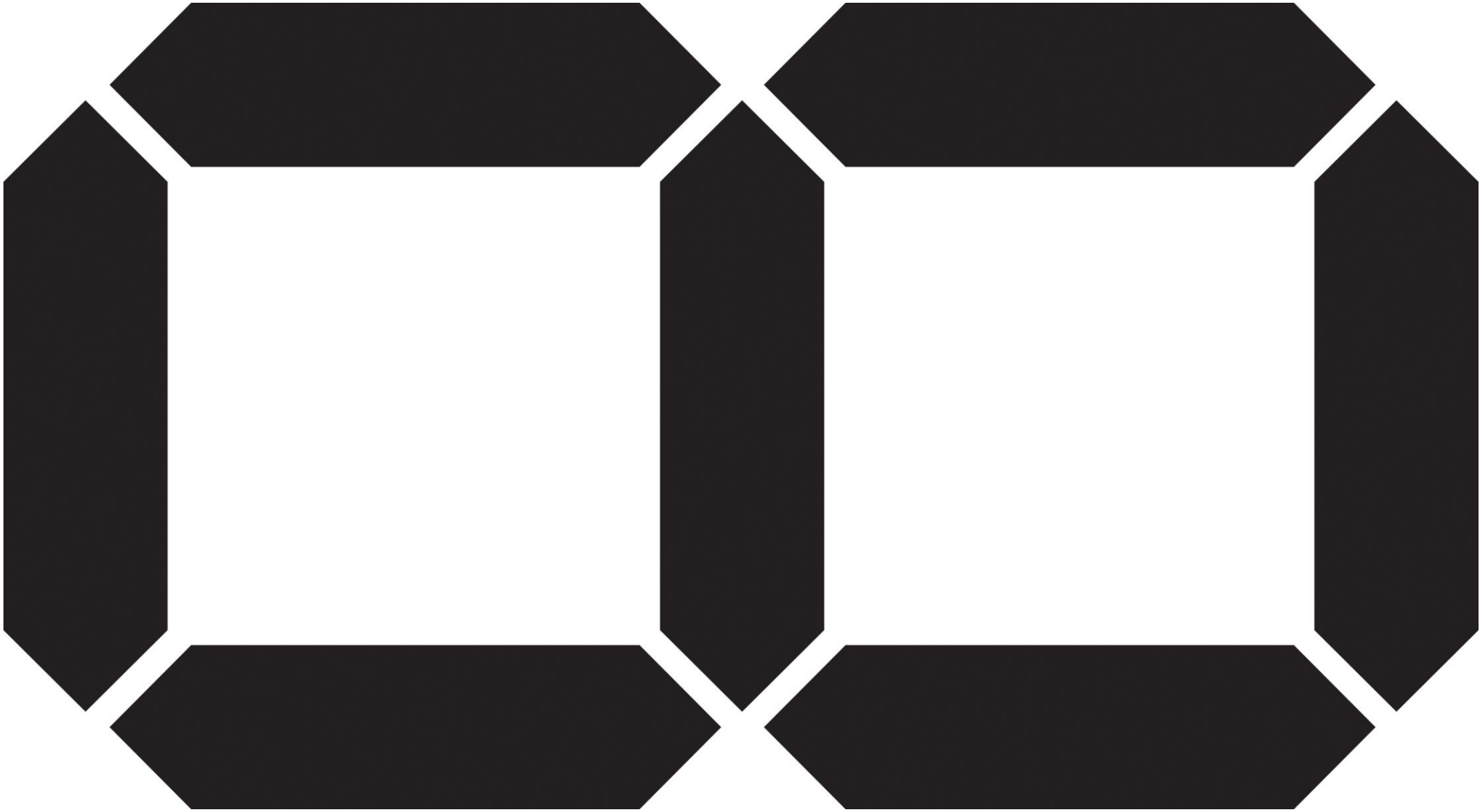
Activity sheet U2-5: Digital display 5



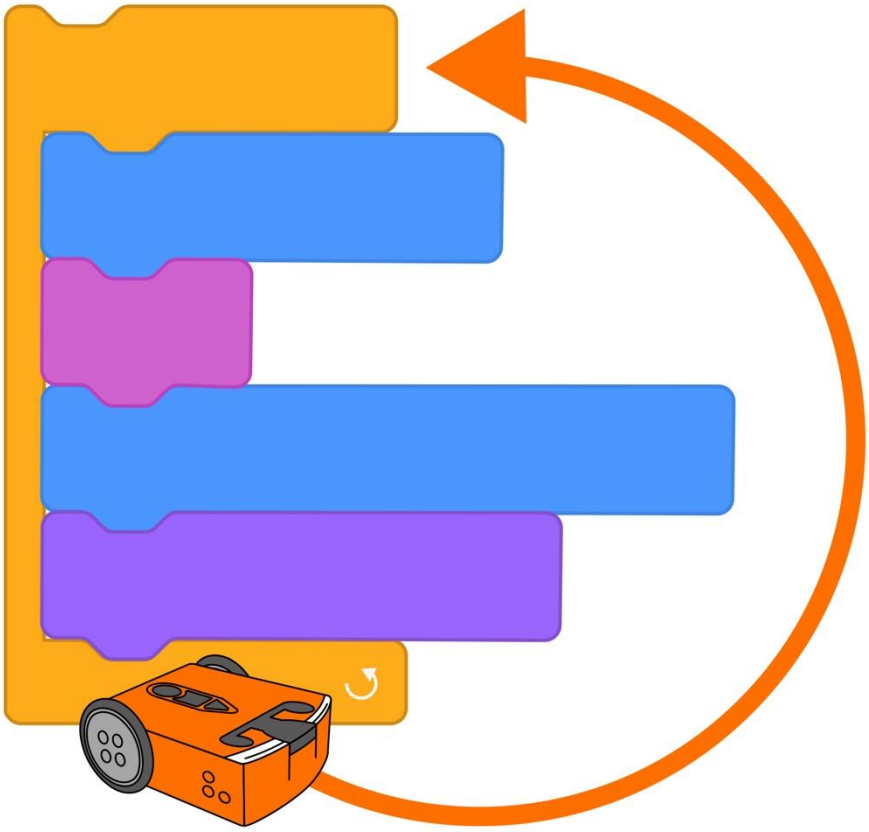
Activity sheet U2-6: Digital display 7



Activity sheet U2-7: Digital display 8



Unit 3: Got loops?



Edison

Name_____

U3-1.1 Let's explore repeating steps

To get a computer, like the Edison robot, to do what you want it to do you need to give very specific instructions. You need to write code that says exactly which actions you want to happen in exactly which order you want each action to happen.



Don't forget

When you write a program for your Edison robot in EdScratch, you are telling the robot what to do and in what order to do each thing. Each EdScratch block is one action you are telling the robot to take. The order you connect the blocks in your program tells the robot in what sequence to do each action. Edison will do the actions one at a time, starting with the top block.

Task 1: Drive in a square

Write a program for Edison using EdScratch so that your robot can drive in a square. Your program should only use blocks from the Drive category to control the motor outputs. Download your program and use activity sheet U3-1 to test your program. Make sure your program has Edison end in the exact same spot it started.

1. How many blocks do you have in your program not counting the start block?

   _____

2. Look at the blocks in your program. What do you notice? Is there a pattern to the blocks?

_____

_____

_____

Task 2: Use a loop to drive a square

To get Edison to drive a square, you need to program the robot to drive each side of the square and turn at each corner of the square. You might have noticed that this makes a pattern in the code: drive the side, turn, drive the side, turn, drive the side, turn, drive the side, and turn one last time, back to the starting position.

Lots of programs have repetition, where a bit of code is used over and over. Repeating stuff is one thing that computers are really good at doing. Unlike a person, a computer doesn't get bored doing the same thing exactly the same way again and again.

Imagine you wanted to get Edison to do the same thing 100 times. Would you want to write out that program using 100 repeating blocks? Would you find that boring to write? Do you think you would be able to write the whole program without making a mistake?

Name_____

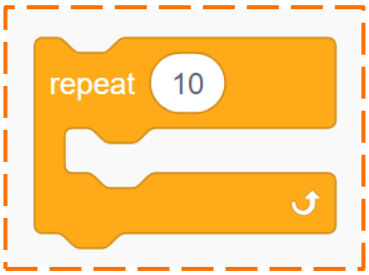
There is an easier and more efficient way to get a computer to repeat commands multiple times. You can get the code to repeat by using something called a loop.



Jargon buster

A loop is a special piece of code that tells a computer to repeat something multiple times. Loops are a type of control structure because loops control other bits of code in a program.

In coding, using loops lets us repeat other bits of code multiple times without having to write each command over and over. In EdScratch, loop blocks are in the Control category in the block pallet. One of the loop blocks in EdScratch is the repeat block:



There are different types of loops. The repeat block is a definite loop.



Jargon buster

A definite loop is a type of loop which will repeat for a set number of times. The repeat block in EdScratch is an example of a definite loop. You tell the loop how many times to repeat using this block's input parameter.

Like all loop blocks in EdScratch, the repeat block wraps around other blocks.



Why is that?

Look at the shape of the repeat block. See how it has a shape a bit like a mouth? Other blocks can sit inside the opening of this block's 'mouth'. Any block that sits inside the repeat block is inside this loop. All blocks inside the loop will be repeated.

Remember, Edison will follow each EdScratch block one at a time. The robot will see the loop block first and know that any blocks inside that loop need to be repeated as many times as the repeat block's input parameter says. The robot will then do the action of each block inside the loop in order. When it gets to the bottom of the blocks in the loop, it will move back to the top of the loop and start again!

Name_____

Try using a repeat block to make a program for Edison to drive a square. You should be able to write a program for Edison which uses only **three blocks** after the start block, including one repeat block. Download your program and use activity sheet U3-1 to test your program. Make sure your program has Edison end in the exact same spot where it started.

3. What value do you need to have in the input parameter in the repeat block to get Edison to drive a square?

   _____

4. Why do you need to have that be the value?

_____

_____

_____

Name_____

U3-1.1a Change it up: Drive a triangle

Even little changes to inputs can make the output of a program completely different. A great example of this is changing the number of repetitions in a loop. Imagine if you wrote a program with a loop that repeats four times, then changed the input so that it repeats five times instead. What would happen when you ran the updated program?

What to do

Write a program for Edison using EdScratch so that your robot can drive in a triangle. Your program needs to use a definite loop control structure, so be sure to include a repeat block. Your program should be as efficient as possible, so try to use as few blocks as you can while still completing the task.

Download your program to your robot and use activity sheet U3-2 to test your program. Make sure your program has Edison end in the exact same spot where it started.

1. How many blocks did you need to use in order to write a successful program (not counting the start block)?

   _____

2. What value do you need to have in the input parameter in the repeat block to get Edison to drive a triangle?

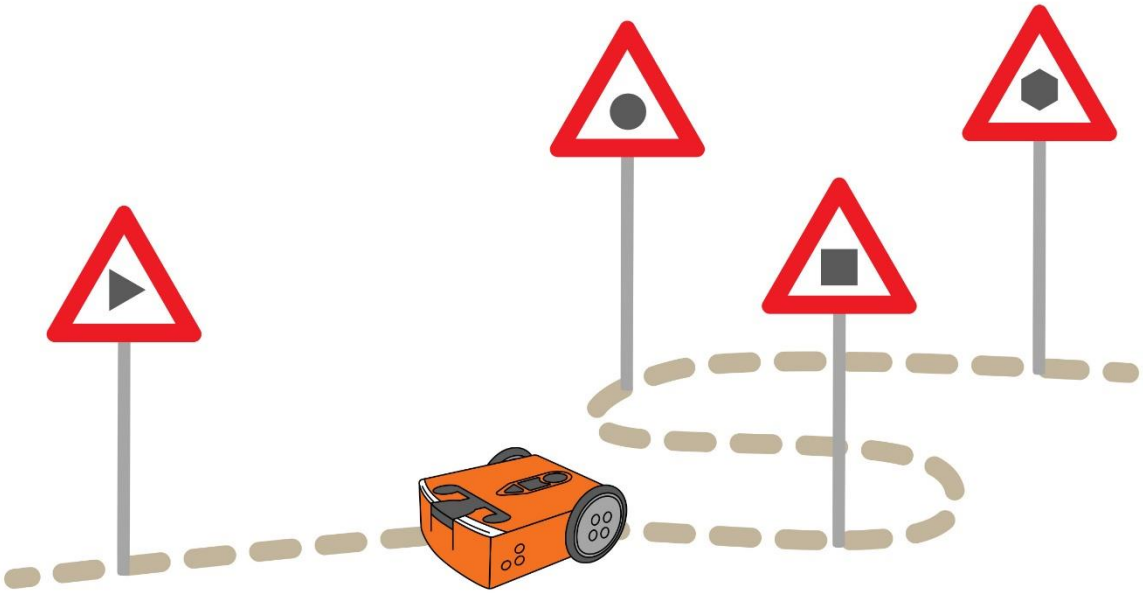
   _____

3. Why do you need to have that be the value?

_____

_____

_____



Name_____

U3-1.1b Change it up: Drive a hexagon

Even little changes to inputs can make the output of a program completely different. A great example of this is changing the number of repetitions in a loop. Imagine if you wrote a program with a loop that repeats four times, then changed the input so that it repeats three times instead. What would happen when you ran the updated program?

What to do

Write a program for Edison using EdScratch so that your robot can drive in a hexagon. Your program needs to use a definite loop control structure, so be sure to include a repeat block. Your program should be as efficient as possible, so try to use as few blocks as you can while still completing the task.

Download your program to your robot and use activity sheet U3-3 to test your program. Make sure your program has Edison end in the exact same spot where it started.

1. How many blocks did you need to write a successful program (not counting the start block)?

   _____

2. What value do you need to have in the input parameter in the repeat block to get Edison to drive a hexagon?

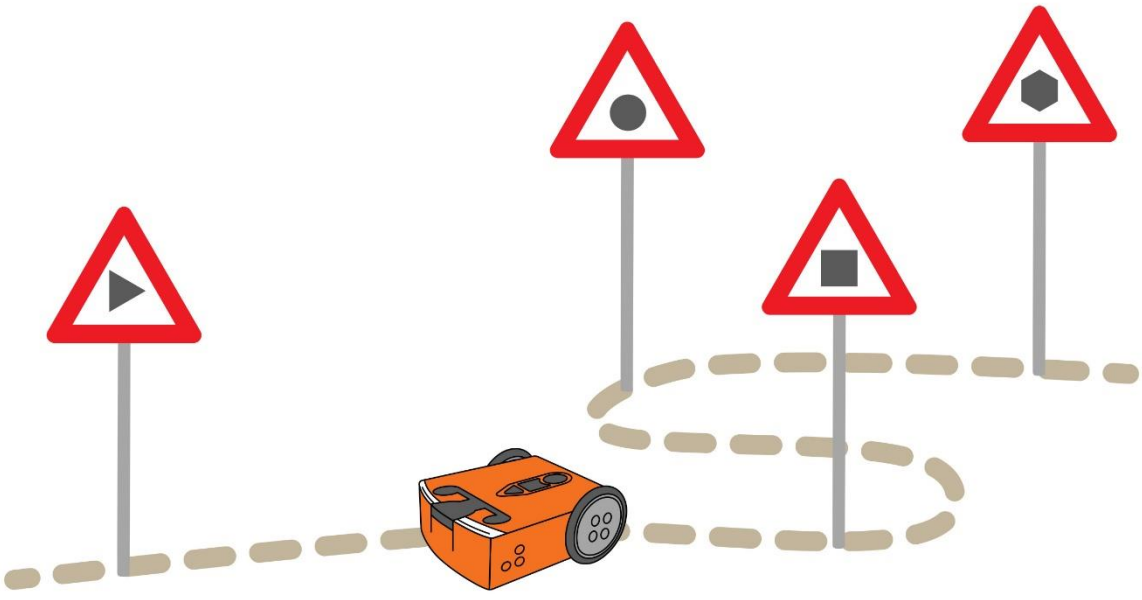
   _____

3. Why do you need to have that be the value?

_____

_____

_____



Name_____

U3-1.1c Challenge up: Choose your shape

Using a definite loop allows you to write a program to get Edison to drive a shape using only a few blocks of code. You can control how many times the program repeats the code commands inside the loop by changing the input of the repeat block.

What do you notice about the number of sides and angles a shape has compared with the input you need in your definite loop? Can you use this pattern to help you write a program to drive any shape?

What to do

Choose a shape which has sides and angles to drive using your Edison robot.

Make a workspace to test your program by either drawing your shape on paper or marking it out on the floor or a desk with coloured tape.



Hint!

You might want to choose a regular shape for this challenge. A regular shape means a shape where all the sides are equal.

Write a program for Edison using EdScratch so that your robot can drive your shape. Your program needs to use a definite loop control structure, so be sure to include a repeat block. Your program should be as efficient as possible, so try to use as few blocks as you can while still completing the task.

Download your program to your robot and test it out using your workspace.

1. What value would you need to have in the input parameter in the repeat block to get Edison to drive a regular (meaning that all sides are equal) 12-sided shape?

   _____

2. There is a pattern between the number of sides and angles a shape has and the number of times you need a loop to repeat in order to drive that shape. Describe how you used this pattern to help you determine the input parameter you needed in the repeat block to get Edison to drive your shape.

_____

_____

_____

_____

_____

Name_____

U3-1.1d Challenge up: Drive a circle

Using a definite loop, like the repeat block, is helpful when you want to write a program for Edison to drive in a shape because shapes have repeating patterns. You have probably noticed a pattern between the number of sides and angles a shape has compared with the input you need to use in a definite loop in a program that gets Edison to drive that shape. Can this pattern help you drive a circle even though a circle has no sides or angles?

What to do

Write a program for Edison using EdScratch so that your robot can drive in a circle. Your Edison needs to drive in the shape of a circle, not just spin in one spot. Your program needs to use a definite loop control structure, so be sure to include a repeat block. Your program should be as efficient as possible, so try to use as few blocks as you can while still completing the task.



Hint!

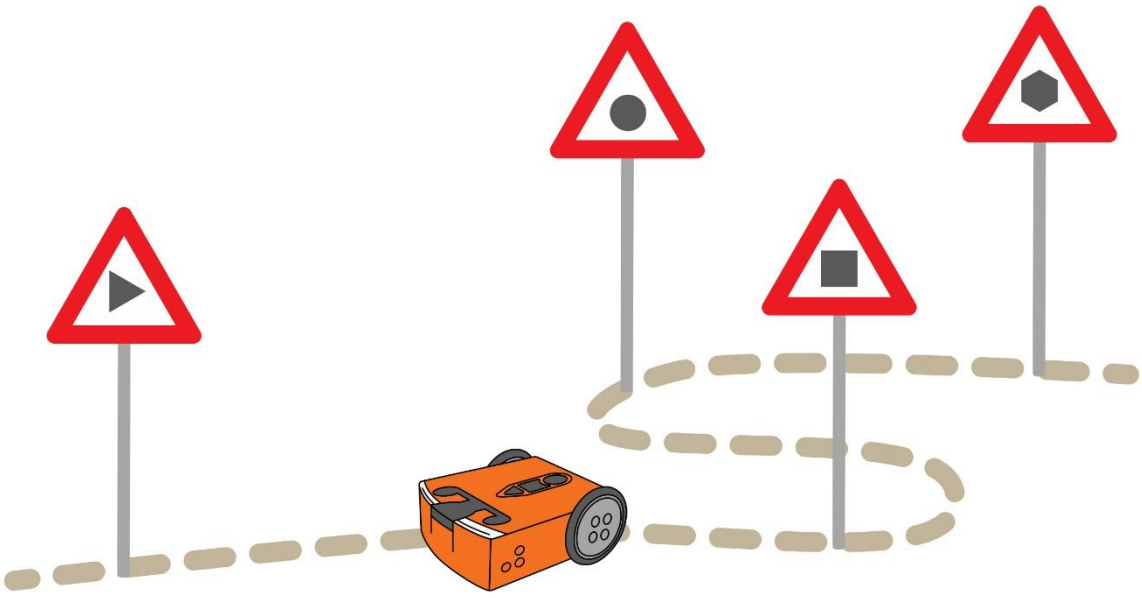
What do you notice about how a shape looks the more sides it has? If you are feeling stuck, try looking at shapes with many sides, such as a decagon and an icosagon. Use the pattern you see to help you write your program.

Download your program to your robot and use activity sheet U3-4 to test your program.

1. What does your program look like? Write your program below. Be sure to include all the input parameters you used.

Name_____

2. Does your robot drive in a perfect circle? If not, can you think of a reason why not?



Name_____

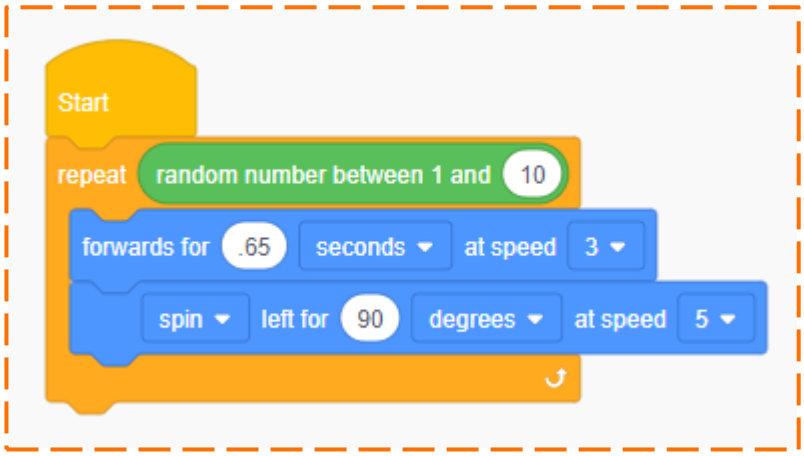
U3-1.1e Change it up: Drive a square?

How many loops does it take to drive in a square? You know that a square has four sides and four angles, so the robot needs to repeat driving and turning four times. That means that if you write a program for Edison to drive in a square using a definite loop, like the repeat block, you need to have the loop repeat four times.

What happens if the loop only repeats three times? How about if it repeats nine times?

What to do

Look at this EdScratch program:



This program is using a special input parameter for the repeat block: the random number block! This block tells Edison to pick a number between 1 and 10 at random. That's how many times the robot will loop the code inside the repeat block.

Write the program in EdScratch. Download your program to your robot and use activity sheet U3-1 to test your program. Try running the program several times to see what happens.

1. What happened when you ran the program? Did the same thing happen every time? Why or why not?

_____

_____

_____

_____

_____

Name_____

U3-1.1f Challenge up: Doodle-bot challenge

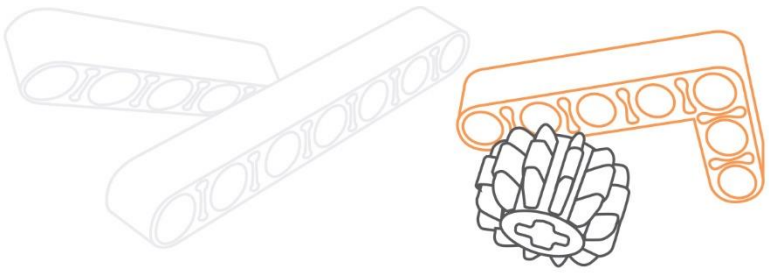
You can program Edison to drive in patterns which make many different shapes. If you attach a pen to your robot, you will be able to get Edison to draw those shapes too!

What to do

Attach a pen to an Edison robot. You can use EdCreate parts or any other materials you like.

Write a program in EdScratch so that when you run the program, your robot will draw a shape. Once you have built your pen attachment and programmed your robot, try running the program to see your shape!

1. Write about what you experienced in this challenge. What problems did you encounter? What did you do to overcome those problems? How did you attach the pen? What shape did you try to draw? Did it work?



Name_____

U3-1.2 Let's explore loops and sequence

Loops are a very useful control structure in coding. Loops can help make programs more efficient by letting you repeat commands without needing to write the same blocks of code multiple times.

When you use loops in programs, you still need to think carefully about sequence. This is especially true when you make programs which have some code inside of a loop and some code outside of the loop.

Let's try making a program that will let Edison drive a quadrilateral. Your program will need to have some of the code inside of a loop, but some of the code will need to be outside of the loop.



Don't forget

Sequence means going in order, step-by-step.

Try it out!

A quadrilateral is a four-sided shape. Squares are quadrilaterals, but not all quadrilaterals are squares. Look at the quadrilateral on activity sheet U3-5. This quadrilateral has four sides and four angles, but they are not all the same.

You need to write a program for Edison using EdScratch so that your robot can drive the shape of the quadrilateral on activity sheet U3-5. Your program should use blocks from the Drive category to generate the motor outputs you need. Your program also needs to use a loop from the Control category.

You need to work out the best place to start Edison on the activity sheet. Be sure your program also has Edison end in the exact same spot where it started.



Hint!

Think about the sequence of actions you want Edison to take. Remember, when you make an EdScratch program for Edison, the robot will start with the top block and do each action in order, one-by-one.

Write your program in EdScratch. Then download your program and use activity sheet U3-5 to test it out.

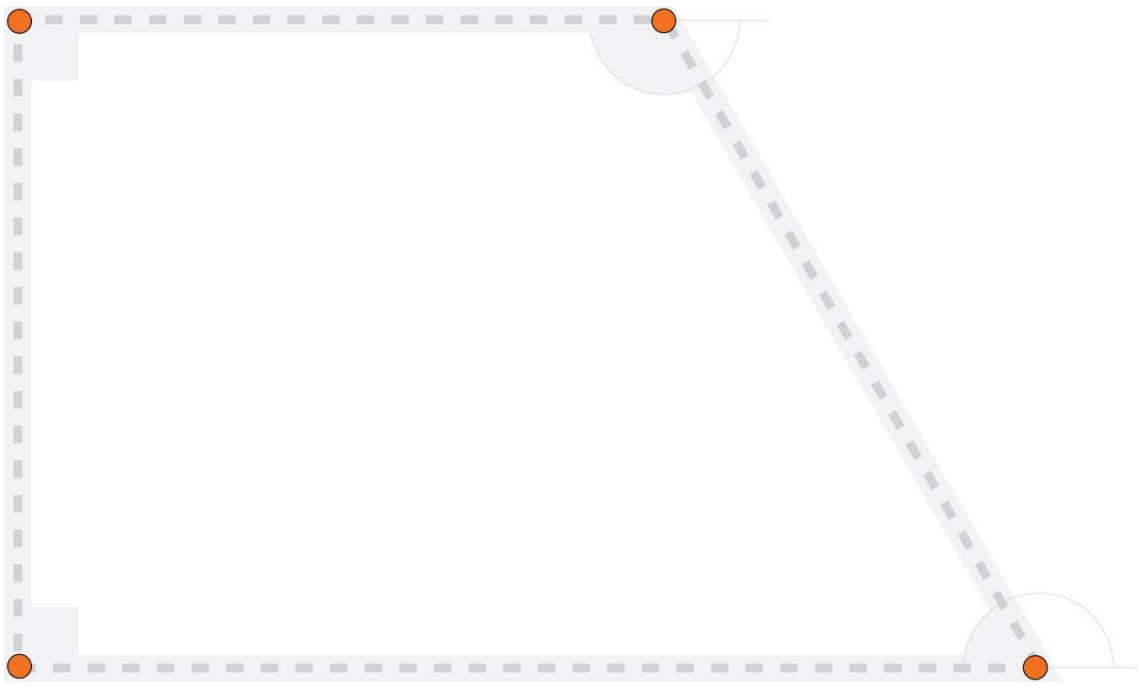


Hint!

You still want to make your program as efficient as possible, so try to use as few blocks of code as you can.

Name_____

1. Where on the quadrilateral did you start Edison? Mark where you started Edison, including which direction you had the robot drive.



2. Look at your start point and your program. How did using this start point affect the sequence of your program?

_____

_____

_____

_____

Name_____

U3-1.3 Let's explore forever loops

Loops let us repeat steps in a program without having to write the same code over and over. When you want a program to do the same thing many times, using a loop is easier than having to write each command again and again. This makes our code more efficient because you can tell the computer to do the same thing using much less code.

In many programs with repeating commands, you know how many times you want the program to loop. If you want to get your robot to drive in a square, for example, you know you need to have Edison drive and turn four times. In that program, you can use a definite loop which repeats four times.

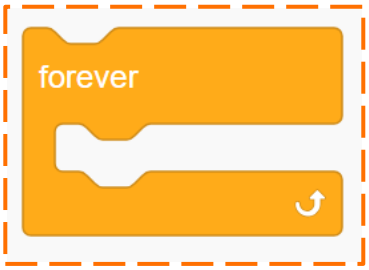
To use a definite loop, you need to know how many times the loop needs to repeat. What if you don't know that? Or, what if you want to make a program that loops forever?



Don't forget

There are different types of loops. A definite loop is a type of loop which will repeat for a set number of times. The repeat block in EdScratch is an example of a definite loop.

To have something repeat forever in EdScratch, you need to use a special loop block in the Control category in the block pallet called the forever block:



The forever block is an **indefinite loop**.



Jargon buster

An **indefinite loop** is a type of loop which will repeat for an undefined number of times. The forever block in EdScratch is an example of an indefinite loop. This loop block tells Edison to keep repeating the code blocks inside this loop forever.

You can think of the forever block in EdScratch as working the same way as the repeat block does, but with the input parameter for the number of loops set to infinity!

The shape of a block in EdScratch can give you some clues about how you use it in the language. Look at the shape of the forever block. Just like all the other loop blocks in EdScratch, the forever block wraps around other blocks. All the blocks that sit inside the loop block will be repeated. What else do you notice about the shape of this block?

Name_____

1. If you write a program using a forever block, do you think you will you be able to add commands for Edison to do after the loop? Why or why not?

_____

_____

_____

_____

Try it out!

Let's turn Edison into an egg timer! Use the forever block to write a program so that Edison will wait a certain number of seconds and then sound an alarm forever.



Hint!

You can always stop a program by pushing the stop (square) button on your Edison robot.

Think about the sequence of things that need to happen for the egg timer program to work. What needs to be inside the loop? What needs to be outside of the loop? Download and test your program with your robot.

2. What does your program look like? Write your program below. Be sure to include the input parameters you used.

Name_____

U3-1.3a Challenge up: Earworm

An earworm is a song that gets stuck in your head for what feels like forever. In this activity, you need to give Edison an earworm by programming the robot with a tune and a forever block!

What to do

Program Edison to play a song or tune over and over using a forever loop block. Write your program in EdScratch, then download it and test it with your robot.



Don't forget

You can always stop a program by pushing the stop (square) button on your Edison robot.



Name_____

U3-1.4 Let's explore stacking and nesting loops

Writing programs using loops lets you be more efficient because you can get a computer to repeat actions without needing to write out the commands multiple times. Using loops also lets you tell a program to do something forever, which you couldn't do if you had to write out every command one by one.

You can also use more than one loop in a program, and you can use the loops in different ways: by **stacking** the loops together or **nesting** loops inside other loops.



Jargon buster

In block-based programming languages like EdScratch, adding blocks together is sometimes called **stacking** blocks and a program is sometimes called a **stack** or a **block stack**. That's why if you use multiple loops together in a program one after another, you can say you are **stacking** the loops.

You can also put a loop block inside another loop block. This is called **nesting** loops.

Why would you use loops in stacks or by nesting them together? Using multiple loops together in this way lets you write programs with repeating patterns. You can even write programs with patterns that repeat inside of other patterns.



Why is that?

Think about an alarm clock on a mobile phone. The alarm can be set to go off in the morning at 7:00 AM. You can set the phone to repeat that alarm every day. When the alarm goes off, it beeps on and off a set number of times. If you snooze the alarm, it stops for a certain amount of time, then comes back on, beeping on and off again for a set number of times.

Can you see how there are repeating patterns inside of other repeating patterns?

This is an example where using stacked and nested loops to write a program would be very helpful. That's because stacking and nesting loops lets you repeat whole sets of commands inside your program.

Stacking and nesting loops have different uses. By stacking loops, we can write programs to get Edison to do different sets of actions multiple times, then move on to a new set of repeated actions. By nesting loops together, however, we can write programs to get Edison to repeat whole patterns multiple times.

Name_____

Task 1: What's going to happen?

Programs that use multiple loops, especially nested loops, might seem a bit confusing at first. To understand what the program is going to do, you need to think about each action that is going to happen in sequence.

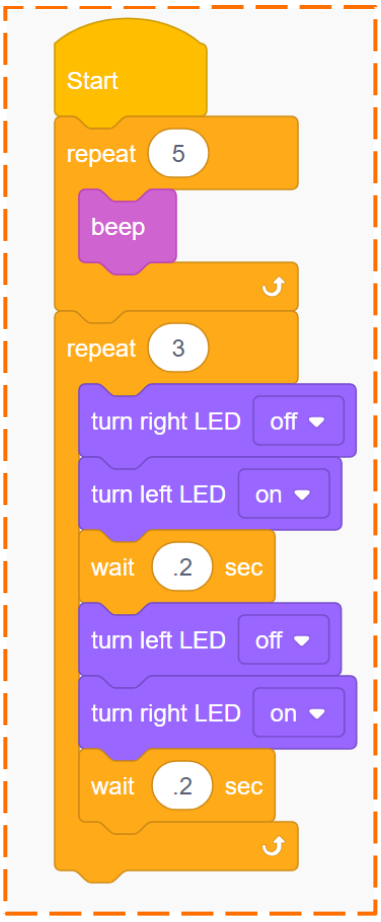


Don't forget

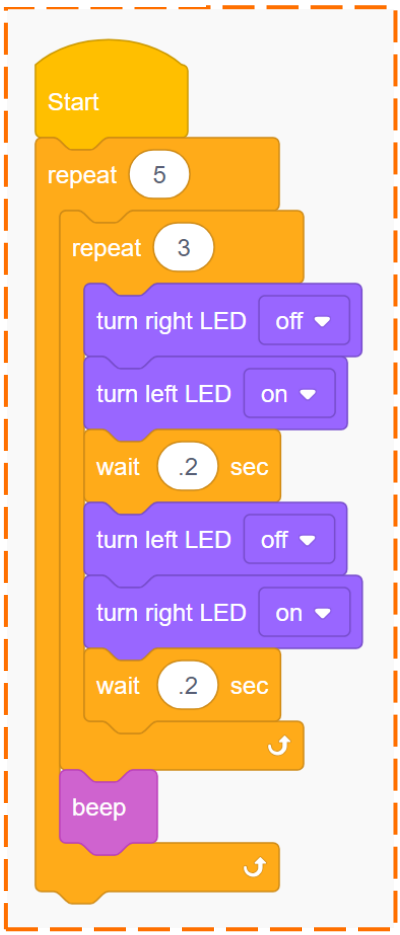
When you make a program for Edison in EdScratch, the robot will start with the top block and do each action one by one. Once it completes a block, it will move to the next block. This is true of all EdScratch programs – whether there are zero loops, one loop or multiple loops!

Look at the following programs and answer the questions about what is going to happen in each program.

Program 1:



Program 2:



Name_____

1. In program 1, how many times will the right LED turn on?

   _____

2. In program 1, which will finish first: all of the beeps or all of the LED flashes?

   _____

3. In program 2, how many times will the right LED turn on?

   _____

4. In program 2, which will finish first: all of the beeps or all of the LED flashes?

   _____



Hint!

Can you follow along with what is happening in each program? If you want to double check your answers, try writing each program in EdScratch, then download it to your Edison robot. Run the program to see what happens.

Task 2: Drive the pattern

Just like a loop lets you repeat a pattern multiple times, nesting loops allows you to repeat multiple patterns! For this activity, you need to use activity sheet U3-6.

5. Look at the pattern on activity sheet U3-6. How would you describe the pattern?

_____

_____

_____

You need to write a program so that Edison will drive the pattern on activity sheet U3-6. Your program can have Edison go across the same line more than once, but the robot must touch all the lines.

Name_____

6. How do you think you can use a nested loop to help you write an efficient program for your Edison robot to drive the pattern on the activity sheet?

_____

_____

_____

Try writing an EdScratch program so that you get your Edison robot to drive the pattern on activity sheet U3-6. Test your idea for using a nested loop to see if it works.



Hint!

You can write a program that completes the activity sheet using just five EdScratch blocks!

Name_____

U3-1.4a Change it up: Edison the designer

Lots of things that run using computer programs have repeating patterns. There are also many programs that have patterns that repeat inside of other patterns. These programs often use nested loops to repeat whole sets of commands inside a program.

What to do

Try using loops to write a program for your robot which makes Edison drive a pattern. If that design has a pattern with a repeating pattern inside of it, try using a nested loop.

Look at activity sheet U3-7 and choose one of the designs to use. For this activity, you will need to create a workspace to test your program. Make a workspace that is large enough to test your program with Edison. You could draw the pattern onto a large sheet of paper or mark it out using dark coloured tape on the floor. Copy out the design onto your workspace. Then write a program in EdScratch that gets Edison to drive that design.



Hint!

Stuck? Try breaking down the pattern into smaller sections and writing code to get Edison to drive each part of the pattern. Link all of the chunks together to get Edison to drive the whole pattern. This can help you to find places where the code repeats. Make your program more efficient by replacing repeating code with loops.

If there is a pattern inside a pattern, be sure to try a nested loop!

Mini challenge!

If you want, you can also design your own pattern to use for this activity. Make sure your design has a pattern repeating inside another pattern. Test your design by writing a program that gets Edison to drive your pattern.

Does your pattern need nested loops?

Name_____

U3-1.4b Challenge up: Dance party!

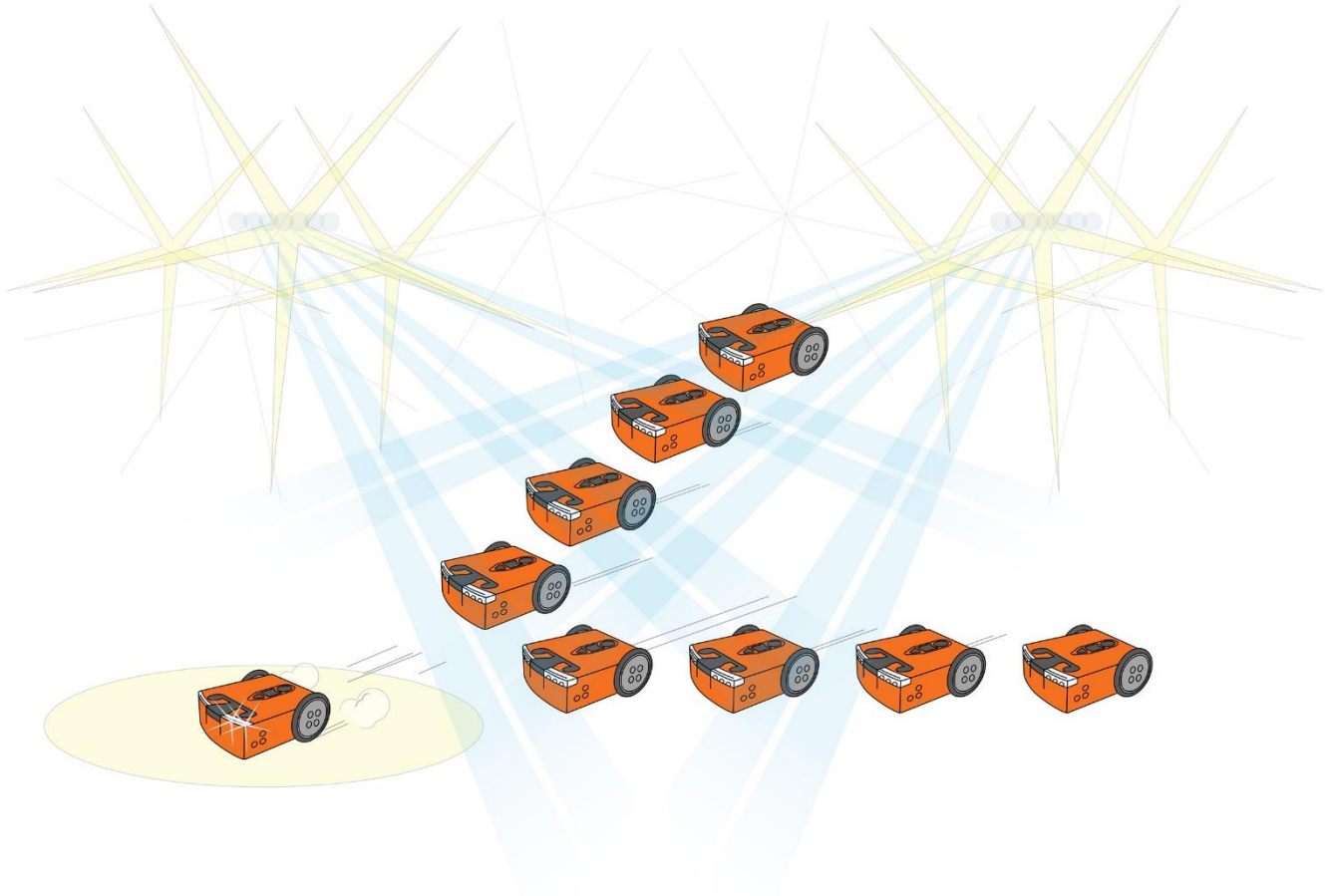
Music and dance often have repeating patterns. The chorus of a song repeats and the notes inside a song often have a pattern as well. Many dances repeat moves too. All this repetition means that if you have a robot dance party, your program probably needs loops!

What to do

Work together to program multiple Edison robots to have a dance party. You can either have all of the Edison robots dance along to a song you play from another device or have one Edison play the song while the other robots dance.

Use any of Edison's outputs (using blocks from the Drive, LEDs and Sound categories) along with loop blocks from the Control category to make a dance program. Will all of the robots do the same thing at the same time? Is one robot the star with the other robots acting as backup dancers?

It's up to you and your team!



Name_____

U3-2.1 Let's explore interrupting the main program

Different computer programming languages have different syntaxes, or rules, which make them look and feel a bit different from each other. No matter the syntax, however, all computer languages work using the same underlying logic. This is why all computer programs behave in similar ways and follow core programming logic, like sequence.



Don't forget

Logic is the organised way of doing things that makes sense to a computer. Logic determines the flow of a program.

Syntax is the rules of how a programming language works.

In EdScratch, the logical flow of a program is to start with the top block and complete each action one block at a time. Programs with loops also follow sequence. When the program sees a loop block, it executes the commands inside that loop in order. When it gets to the bottom of the loop, it goes back to the top of the loop and starts again. Even though loop blocks make programs look a bit different, these programs still follow the logical flow of top-to-bottom sequence.

There is a way to interrupt this sequential flow. You can disrupt a computer program's flow by using an **interrupt**.



Jargon buster

An **interrupt** is a special bit of code that stops the flow of the main code. It's called an interrupt because it *interrupts* the main code. An interrupt is usually used to pause the main code in order to run a **subroutine**.

A **subroutine** is a distinct set of code that is separate from the main program. You can think of a subroutine as a mini program.

To understand how interrupts work, we need to understand what is being interrupted.

What is the main program?

In EdScratch, the main program is whatever is attached to the yellow start block.

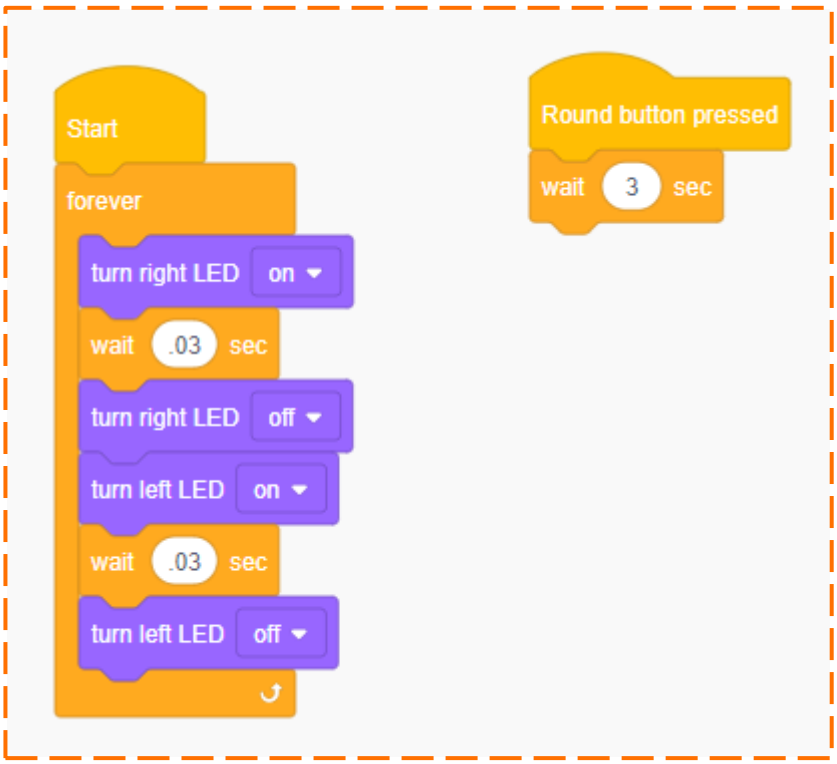


Whenever you write a program for Edison, you need to have at least one block in the main program attached to the start block. When you run a program with your Edison robot by pressing the play (triangle) button, this main program runs block-by-block in sequential order until it reaches the end of the program.

An interrupt can disrupt this flow.

Name_____

Look at the following program:



This program has two parts: the main program and the subroutine.

1. What does the main program tell Edison to do? *Hint:* only the blocks attached to the start block are in the main program.

_____

_____

_____

In addition to the main program, there is also a subroutine. What causes the main program to be interrupted and run the subroutine?

Subroutines will only run when a specific **event** happens.



Jargon buster

In programming, 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.

In EdScratch, you need to use a block from the Events category at the start of any subroutine.

Name_____

2. Look at the blocks in the Events category in EdScratch. What do you notice about the shape of these blocks?

_____

_____

The Event blocks are interrupts that tell the program to look out for that particular event. If the event happens, the Event block interrupts the main program immediately and runs the subroutine. Once the subroutine code is complete, the program returns to wherever it left off in the main program.



Why is that?

We use interrupts in programming because interrupts allow a program to react to an event at any time while the program is running. By using interrupts, you don't have to predict when an event will occur. Without interrupts, you would have to know exactly when something was going to happen, even when that event is out of your control!

Try it out!

Write a program in EdScratch that contains both the main program and the subroutine just as they appear in the picture from earlier in this activity. Download the program to your Edison robot. Press the play (triangle) button on your robot. This will start the main program, causing Edison's LEDs to flash on and off. Now press the round button on the robot. This interrupts the main program and runs the subroutine. This subroutine tells Edison to wait 3 seconds, then return to the main program.

You can use this program to turn your Edison robot into a decider bot!

Think of a question that you can answer with a 'yes' or a 'no'. The decider bot will help you answer that question. If the right LED is lit up when you press the round button, the answer to your question is 'yes', but if the left LED is lit up, then the answer is 'no'.



Why is that?

Remember, an interrupt pauses the main program instantly, so it is possible that neither LED will be lit up when the subroutine runs. If the main program has turned off the right LED but hasn't yet turned on the left LED when the interrupt occurs, then both LEDs will be off. This is a bit like flipping a coin and having it land on its edge – rare, but it can happen!

Name_____

U3-2.1a Change it up: Try a clap instead

Remember Edison's sound sensor? This is the special bit of tech that's both a buzzer and a sound sensor. This is the bit of Edison that makes noise, like beeps or musical notes, but can also detect sounds, like a clap.

You can use Edison's sound sensor to trigger an interrupt in a program. That way, you can make a decider bot that responds to the sound of a clap!

What to do

Write a program to turn Edison into a decider bot. Your main program should have Edison flash its two LEDs on and off forever. You also need a subroutine that will interrupt the main program if the robot detects a clap. Your subroutine should tell Edison to wait for a few sounds so that you can see which LED is on and get your answer.



Don't forget

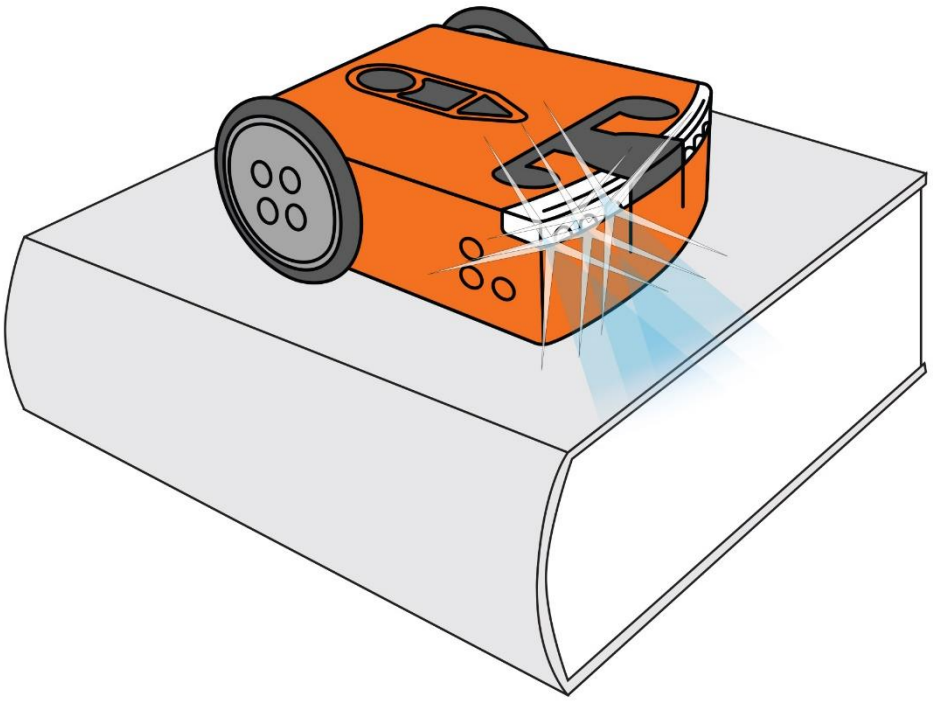
In EdScratch, you need to use a block from the Events category to act as an interrupt and be the first block at the start of any subroutine.

Download your program to your Edison robot and test it out.



Hint!

You can use the decider bot program from activity U3-2.1 as a base for your program.



Name_____

U3-2.1b Challenge up: Cheater bot

Using a decider bot is a fair way to choose between two options. The LEDs flash so quickly that it's very hard to 'cheat' and get the robot to answer the way you want. You can build a decider bot with a 'cheat' however… but you need to add a second subroutine to do it!

What to do

Write a program to turn Edison into a decider bot with a secret cheat. Your main program should have Edison flash its LEDs on and off forever. You also need two subroutines: one subroutine that is fair and one that cheats. The 'fair' subroutine needs to interrupt the main program if the robot detects a button being pressed, and tell Edison to wait for a few sounds so that you can see which LED is on to get your answer.



Don't forget

In EdScratch, you need to use a block from the Events category to act as an interrupt and be the first block at the start of any subroutine.

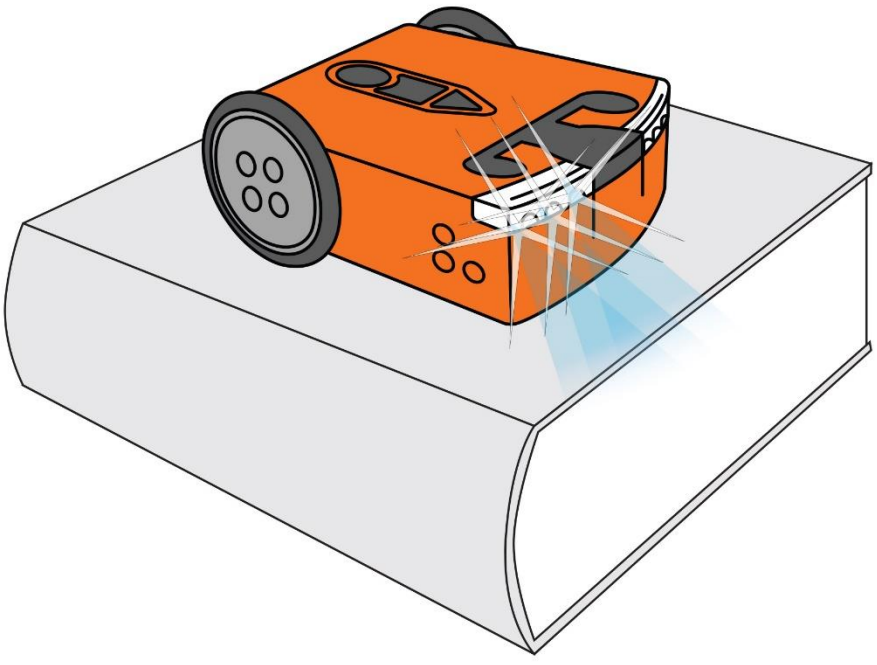
The second subroutine should also interrupt the main program if the robot detects a different button has been pressed. Instead of just waiting, however, you need to design that subroutine to give you a set answer.

Download your program to your Edison robot and test it out.



Hint!

You can use the decider bot program from activity U3-2.1 as a base for your program.



Name_____

U3-2.1c Challenge up: Pick one

A decider bot selects an answer from two different options. When you use Edison as a decider bot using its LEDs, you have to remember which choice the right LED represents and which choice the left LED represents. Instead of remembering, why not create a way for Edison to light up the answer!

What to do

Write a program to turn Edison into a decider bot. Your main program should have Edison flash its LEDs on and off forever. You also need a subroutine that will interrupt the main program if the robot detects a button being pressed and tells Edison to wait for a few seconds so that you can see which LED is on and get your answer.

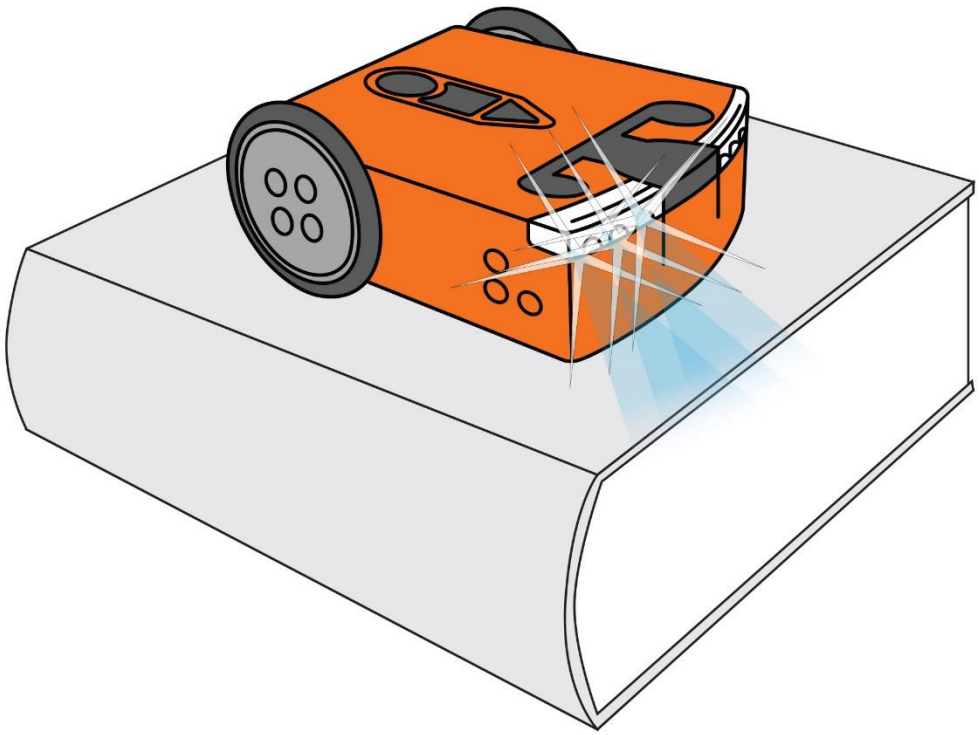
You also need to create some sort of physical set-up so that the decider bot's choice lights up the answer. That way you can write down two choices, and Edison will light one up!

Download your program to your Edison robot and test it out using your creation.



Hint!

You can use the decider bot program from activity U3-2.1 as a base for your program.



Name_____

U3-2.2 Let's explore comments in coding

Part of learning how to code is learning to speak another language: a computer programming language! The more coding you do in a programming language, the easier it is to understand programs written in that language. You can look at a program, follow each command in order, and start to figure out what the program will do when you run it.

There is also a tool to help make it easier for us to read programs called **comments**.

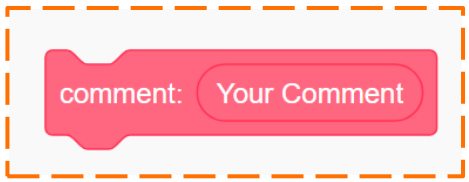


Jargon buster

In programming, **comments** are notes that the programmer adds to help keep track of things. Comments are messages to document what's happening in the program and clarify things so that people can understand the program.

What comments look like depends on the programming language. Sometimes comments can look a bit like code, but comments are not actually code. When a computer runs a program that has comments in it, the computer will ignore the comments. Comments are just for people!

In EdScratch, you can add a comment to your program by using the special block in the Comments category. Here is what the comment block looks like:



The comment block looks similar to other blocks in EdScratch, but it works a bit differently. See where it says 'Your Comment' in the block? That's where you can write in your note. Remember, this block isn't code for Edison. When Edison sees a comment block in a program, it simply skips the block and moves on to the next command. You can think of the message you write in a comment block as that block's input parameter, but instead of being information for Edison, it is a note for a person.

How do you use comments?

In many ways, how you use comments in your program is up to you. Comments do not need to follow syntax, so there isn't a specific way you must write your comments. You can phrase things in your comments however you think makes the most sense. You can also add comments to your code wherever you think you need a note to help explain what comes next.



Why is that?

Because comments are just written for people, you don't have to worry about the computer understanding what you mean. That's why comments don't need to be written in the syntax of the programming language.

Name_____

Adding comments to a program makes it easier for other people to read your program, but the person that is most likely to read your comments is you in the future! This is because comments are a helpful tool for debugging your programs.

By adding a comment, you can organise your thinking and keep track of what it is you are trying to do. That way, if something in your program doesn't work the way you intended, it is easier to go back and see where the issue might be.

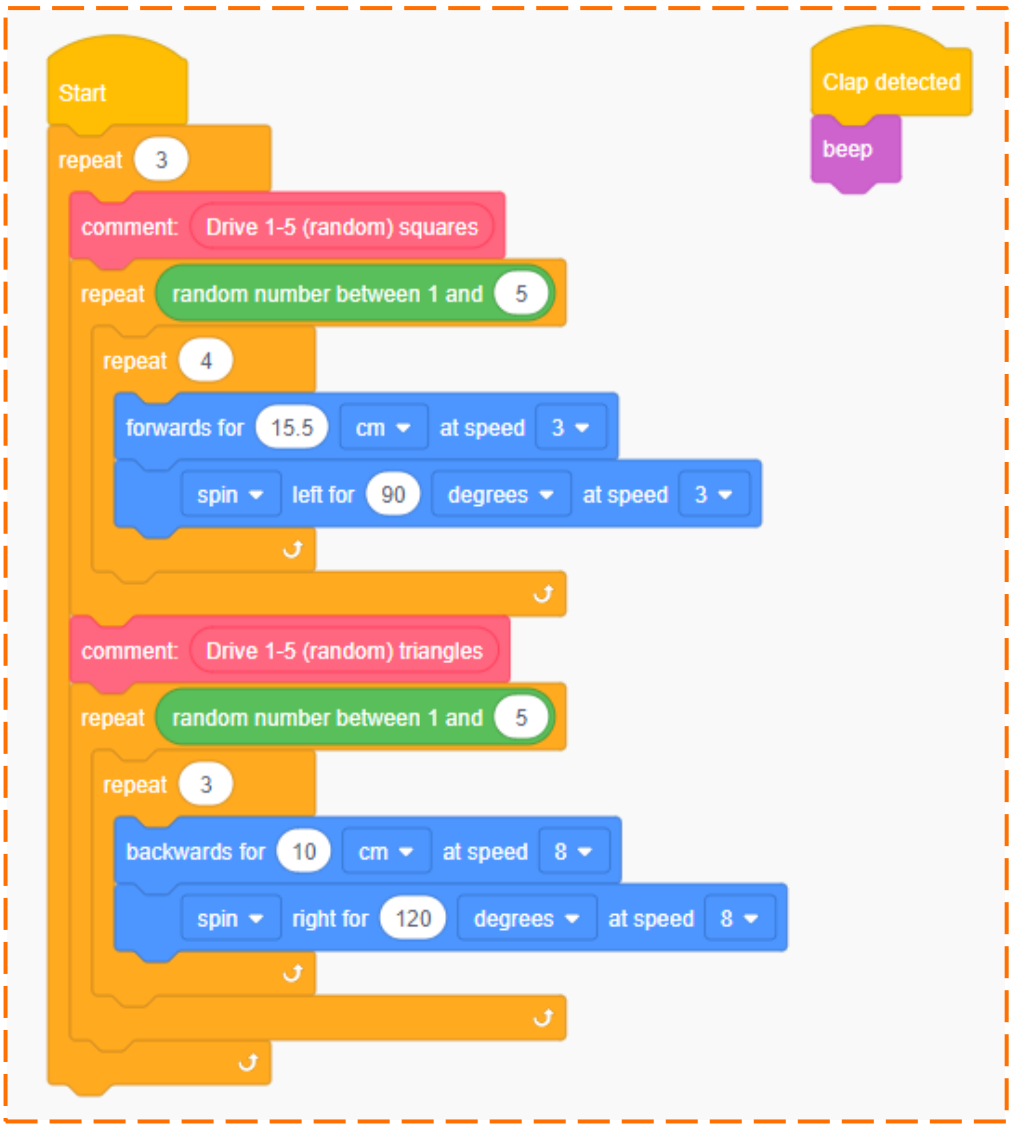


Don't forget

Finding and fixing problems in a computer program is called debugging.

Try it out!

Look at this program:



The programmer has added some comments to the code to help make the program easier to read. Use this picture to answer the following questions.

Name_____

1. What do you think Edison will do if you program the robot with the code in the picture?

2. Do the comments make it easier for you to understand what the programmer wants the program to do? Why or why not?

Now try programming your Edison with the program in the picture.

3. Did the program work the way you expected? Describe anything that happened that you didn't expect.

When the programmer ran this program, something unexpected happened:

"I only wanted Edison to beep when I clapped, but the robot keeps beeping repeatedly the whole time the program is running. Why is that?"

4. Why is Edison beeping? *Hint:* Is there a clue in the EdScratch environment?

Name_____

U3-2.2a Challenge up: Create and comment

Using comments is a good way to organise your thinking as you write code. Adding comments lets you leave a little message to yourself or someone else about what the code is doing. This is really helpful if you need to come back to a program later on, especially if you need to do some debugging. Reading the comments in a program is a quick way to know what is meant to be going on!



Don't forget

The person that is most likely to read your comments is you in the future!

What to do

In this activity, you need to design a program in EdScratch to run with your Edison robot. What your program does is up to you, but it needs to include the following things:

- ☐ a main program
- ☐ at least one subroutine that is triggered by either a button press event or a clap event
- ☐ at least one loop
- ☐ at least two types of outputs

Design and write your program in EdScratch. Add comments as you write your program to help keep track of what you are trying to do and to help someone else read your program.

1. What do you want your program to do? Describe what your program should do when you run it in Edison.

_____

_____

_____

_____

2. Where in your program did you include comments? What did you use them to say? Write down at least one example of a comment you included in your program. Explain where you put the comment block, why you put it there, and what the comment says.

_____

_____

_____

_____

_____

Name_____

Test your program with your Edison robot. Does it work like you expected? Is there anything happening you don't want to happen? Is your program doing everything you want it to do and is each action happening in the order you want?

3. Describe any issue you had when you first tested your program. What did you do to fix the issue?

_____

_____

_____

_____

Once you have your program working just how you want, go back and look at your comments. Do they all still make sense? Do you need to add any new comments or get rid of any of the ones you put in originally? Refine your comments so they work with your final program.

4. What does your finished program look like? Write your program below. Be sure to include the input parameters you used and your comments.

Name_____

U3-2.2b Challenge up: Share your comments

Comments help make it easier for other people to read your code and understand what your program is meant to do. That is, so long as other people understand what your comments mean!

A lot of the time, people work together on coding projects, and comments make it easier to collaborate. Because there aren't strict rules about how to write comments, however, different people write comments in different ways. The result is that sometimes comments are more confusing than the code!

What to do

You will need at least one partner for this activity.

Design and write a program in EdScratch. Keep your program a secret from your partner!

What your program does is up to you. Be sure to include comments to help keep track of what it is you are trying to do and to help someone else read your program.



Hint!

You can use activity U3-2.2a as inspiration for what to include in your program.

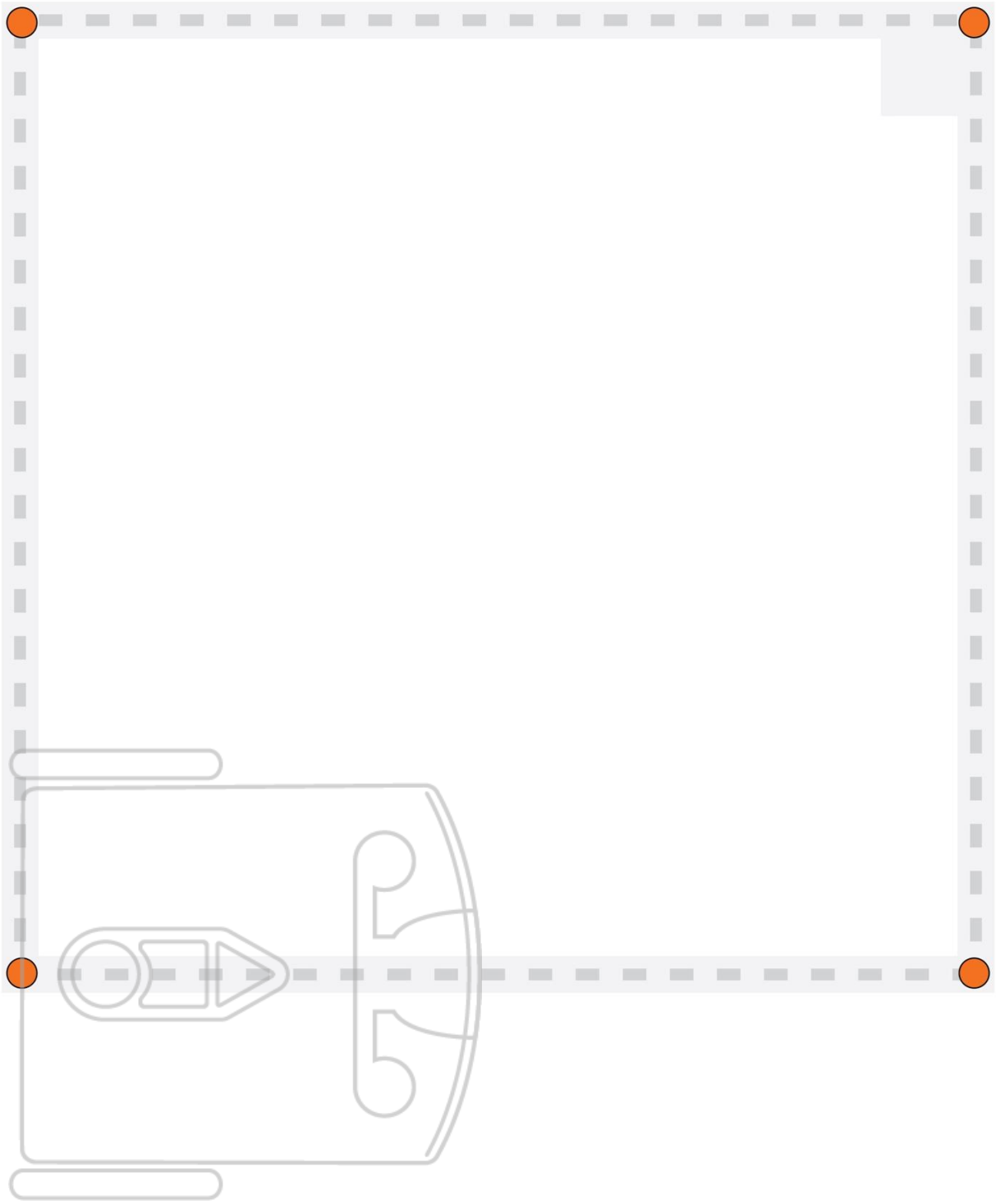
Once both you and your partner have finished writing your own programs, swap with each other.

Without running the program in Edison, can you understand your partner's program by reading their code and comments? Can your partner understand your program and comments?

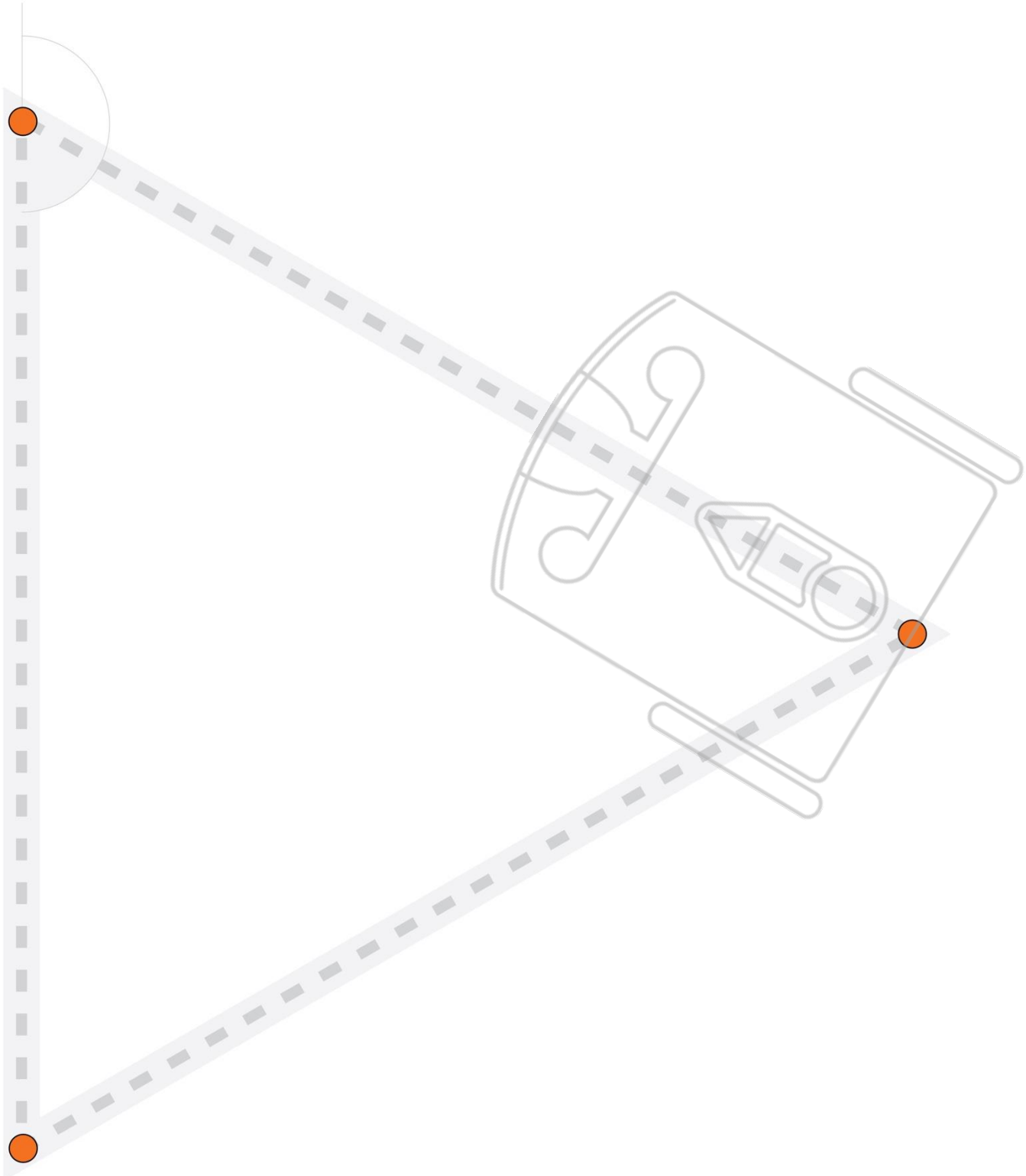
1. Pretend that you are in charge of making the rules for how everyone should use comments in EdScratch. What rules should everyone follow when they use comments so that the comments are helpful and easy to understand? Work together with your partner to decide the new rules for using comments.

Name_____

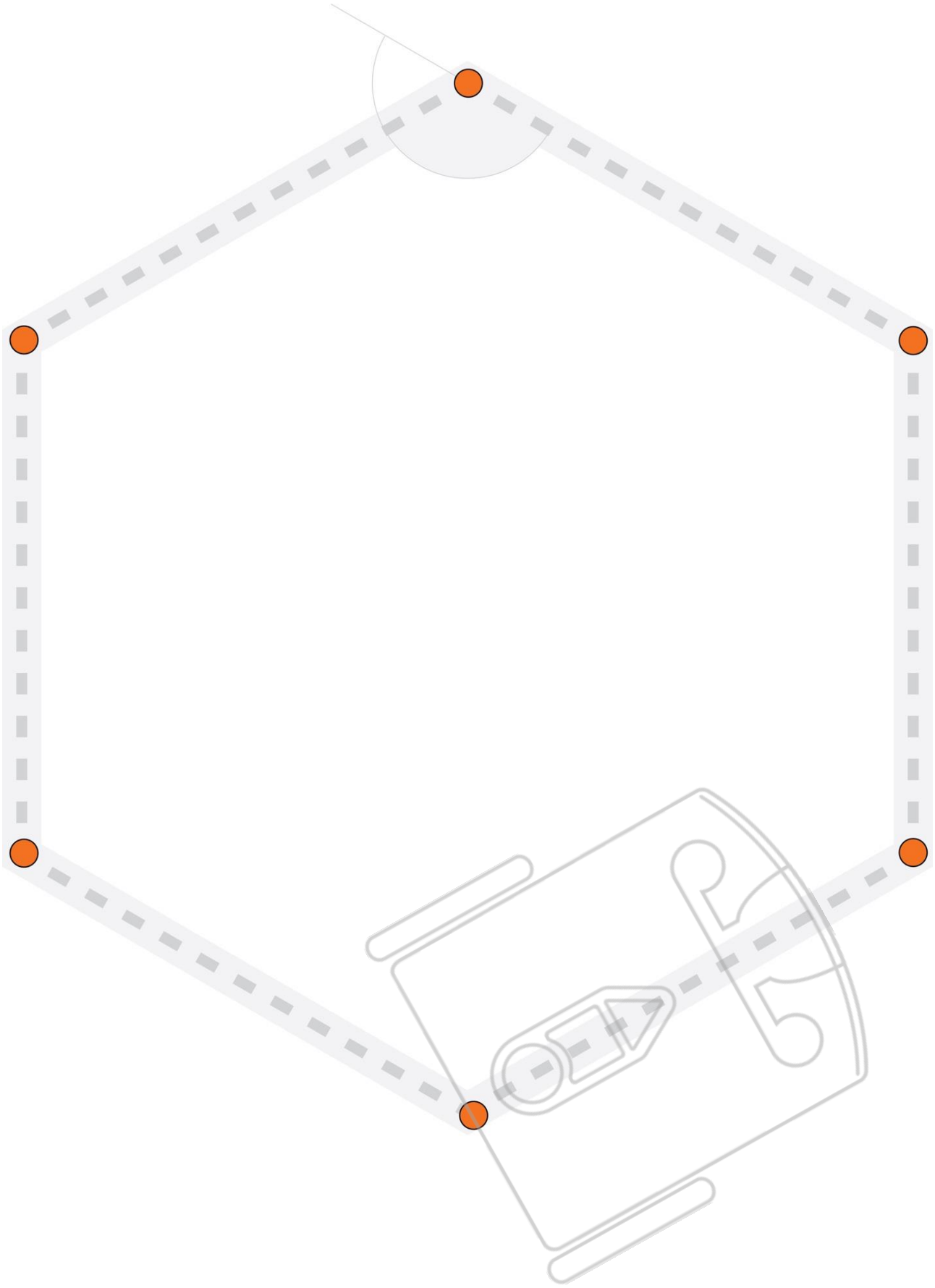
Activity sheet U3-1: Drive a square



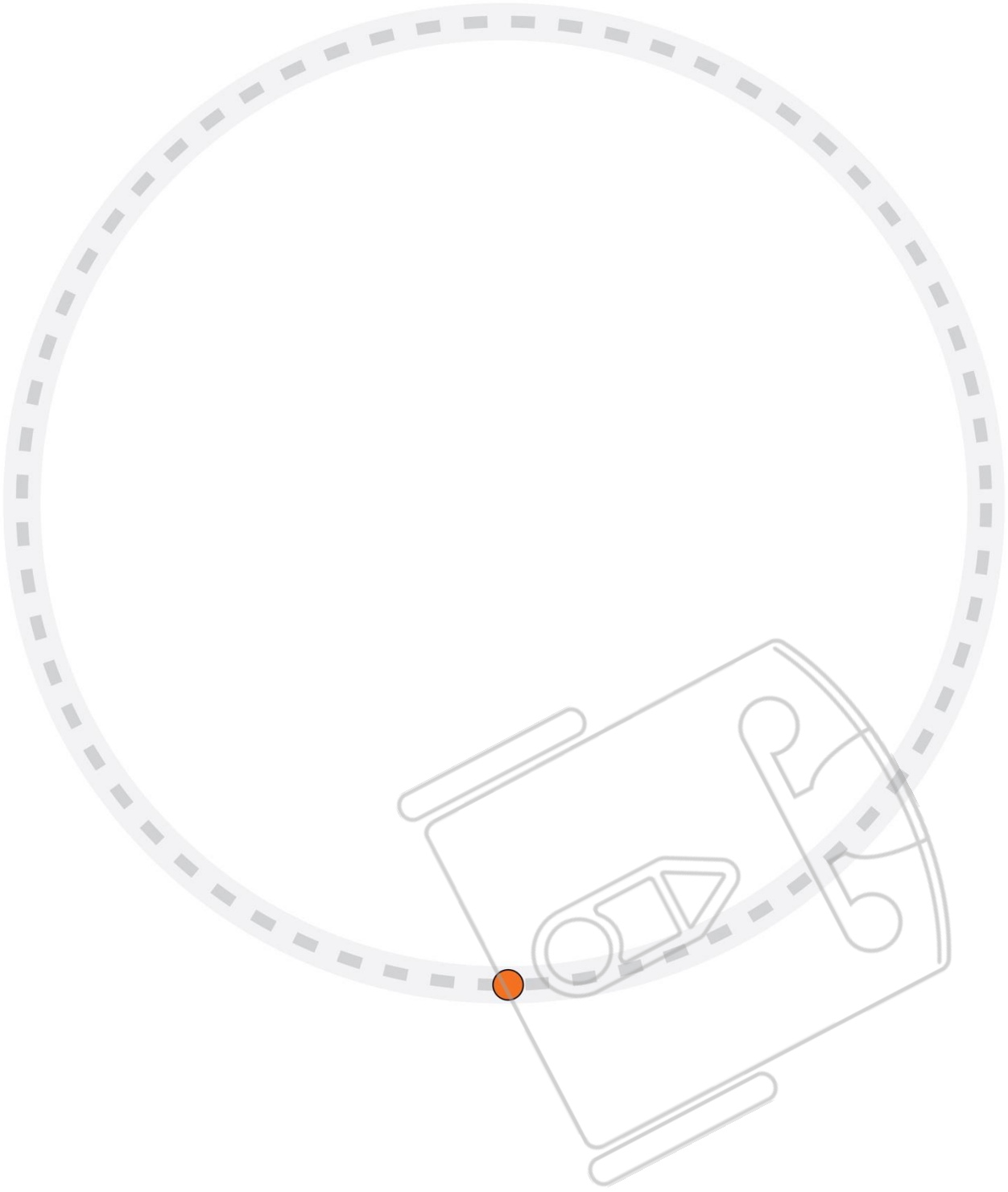
Activity sheet U3-2: Drive a triangle



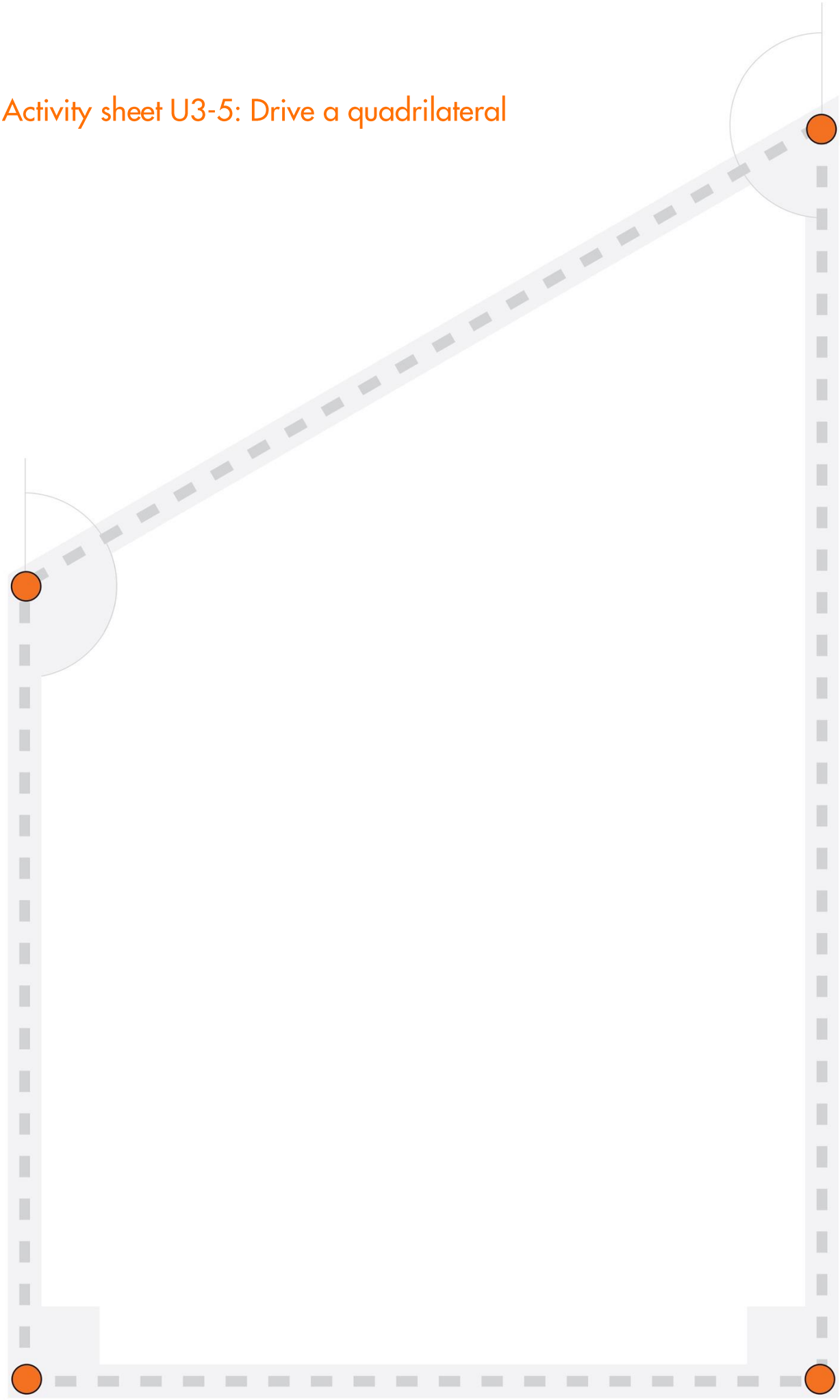
Activity sheet U3-3: Drive a hexagon



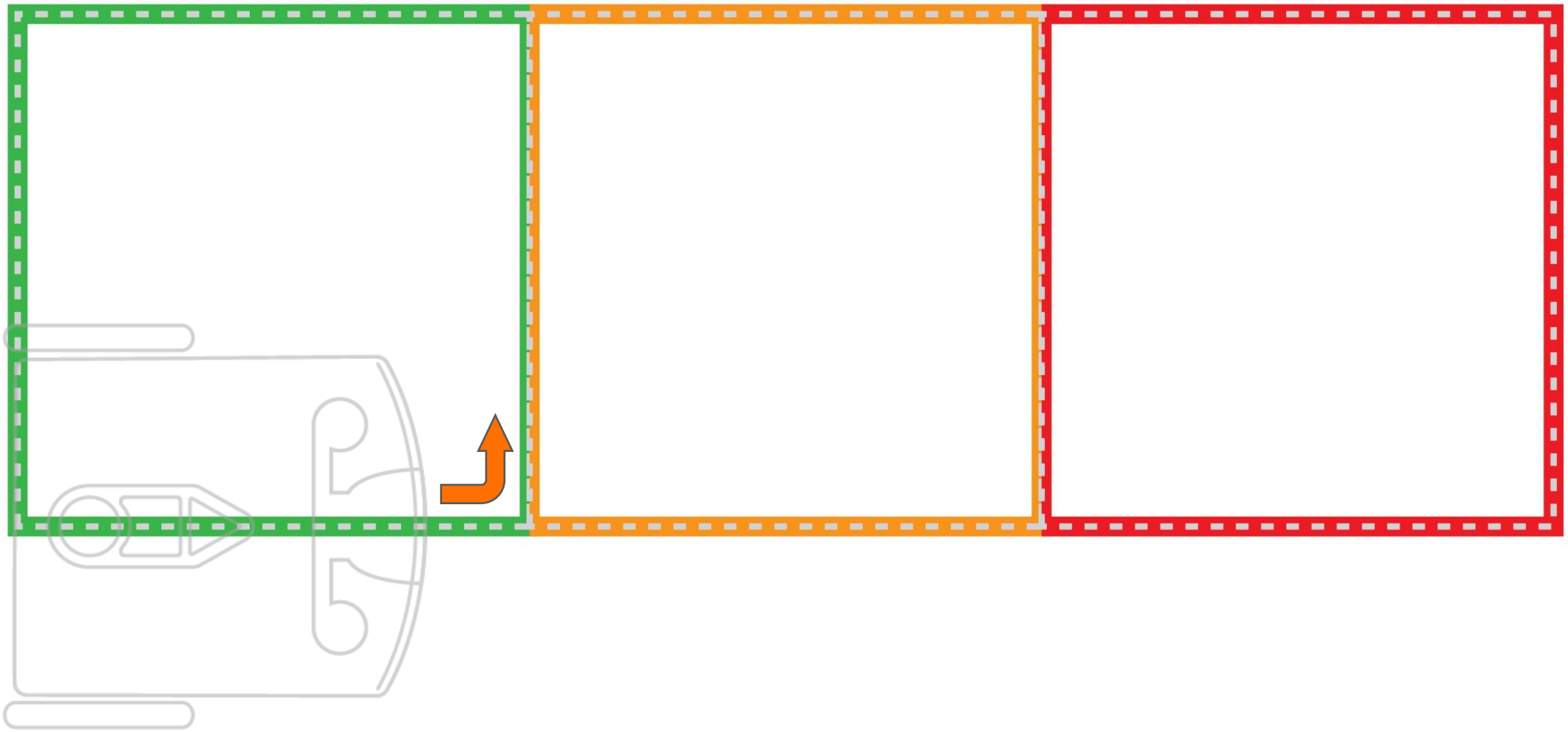
Activity sheet U3-4: Drive a circle



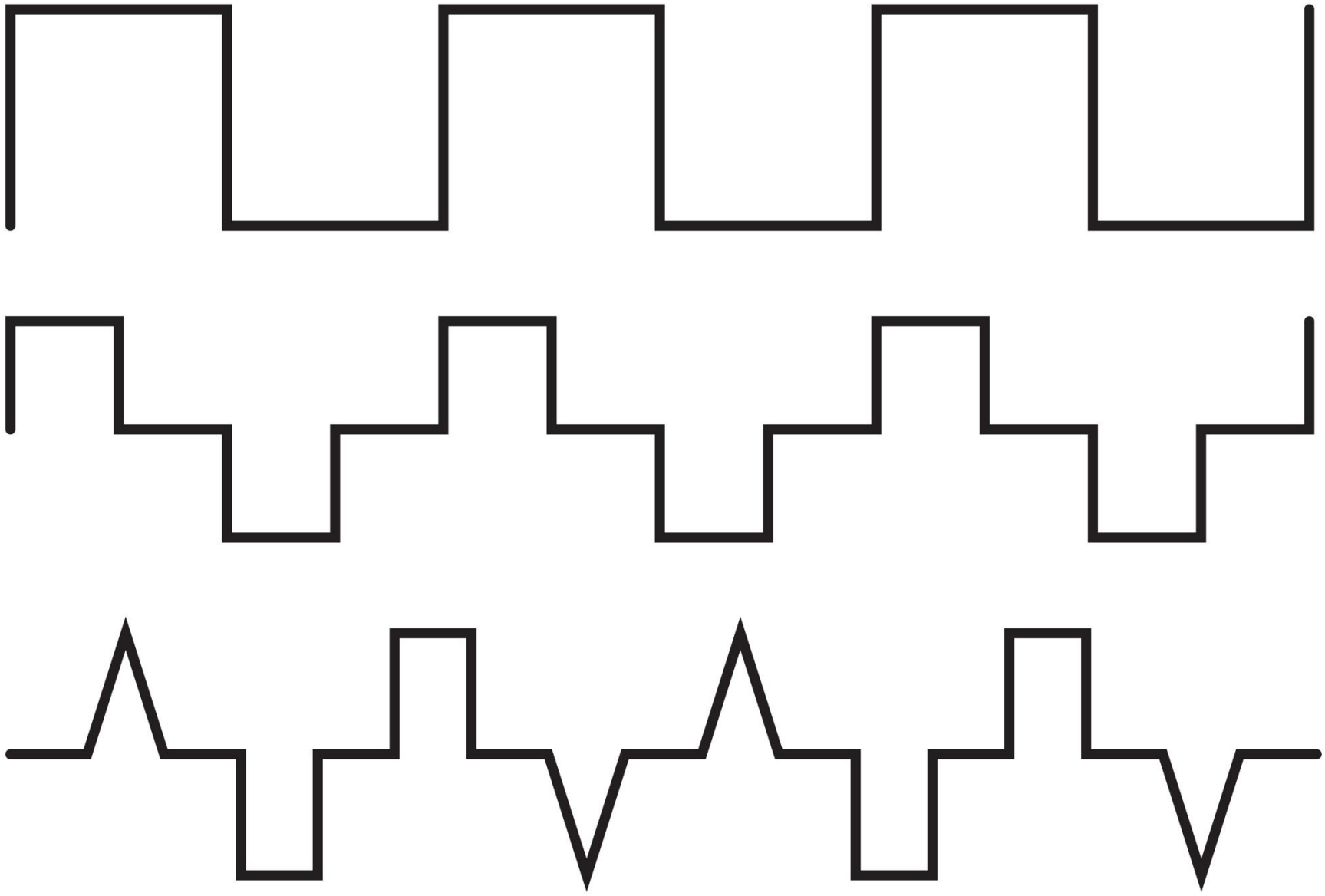
Activity sheet U3-5: Drive a quadrilateral



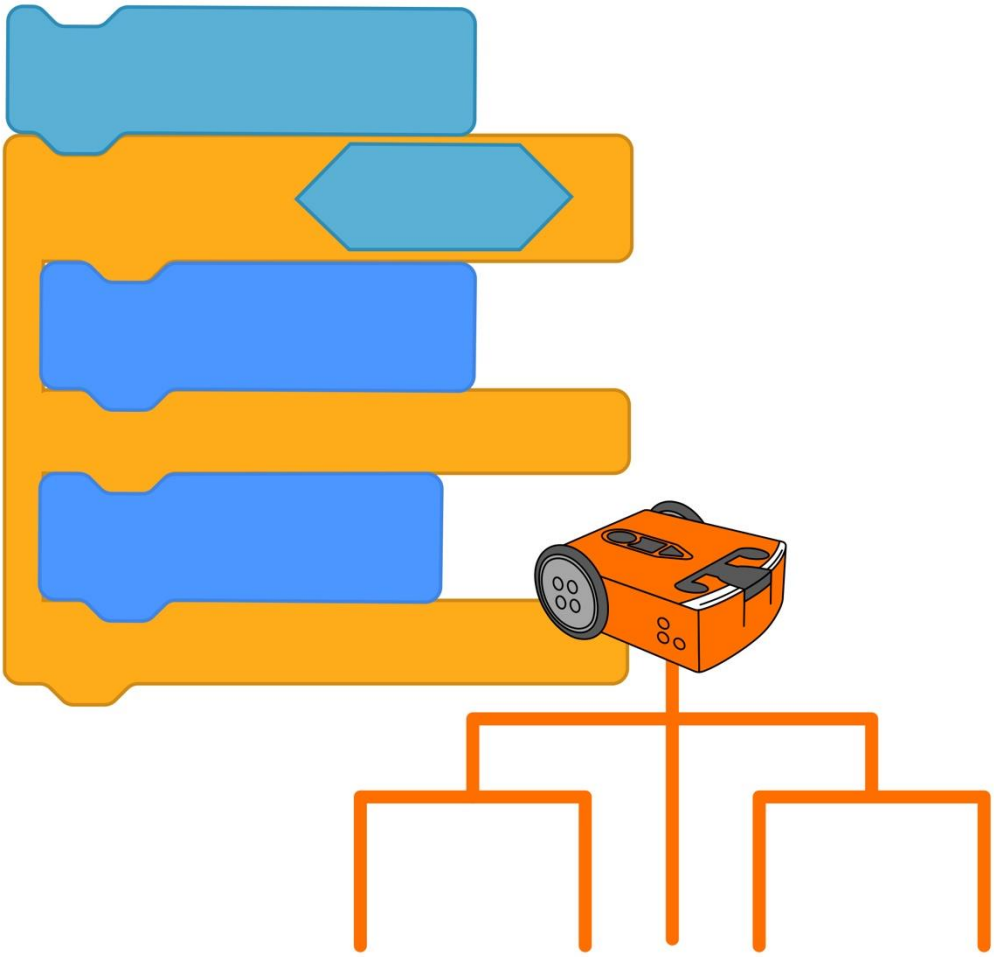
Activity sheet U3-6: Repeating squares



Activity sheet U3-7: Driving designs



Unit 4: What if…



Edison

Name_____

U4-1.1 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**.



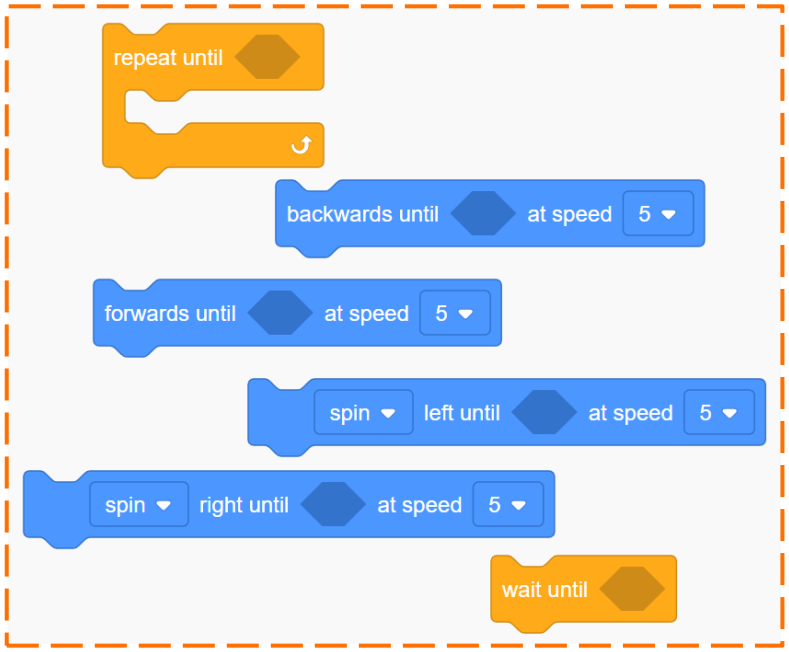
Jargon buster

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?

Name_____

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.

1. 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?

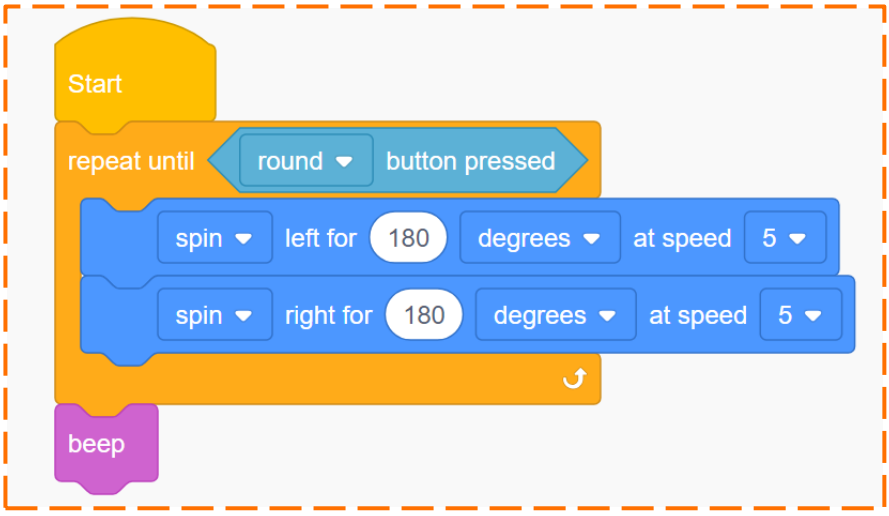
_____

_____

_____

Task 1: Repeat until…

Look at this EdScratch program:



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

Name_____



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.

2. 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.

_____

_____

_____

Task 2: 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

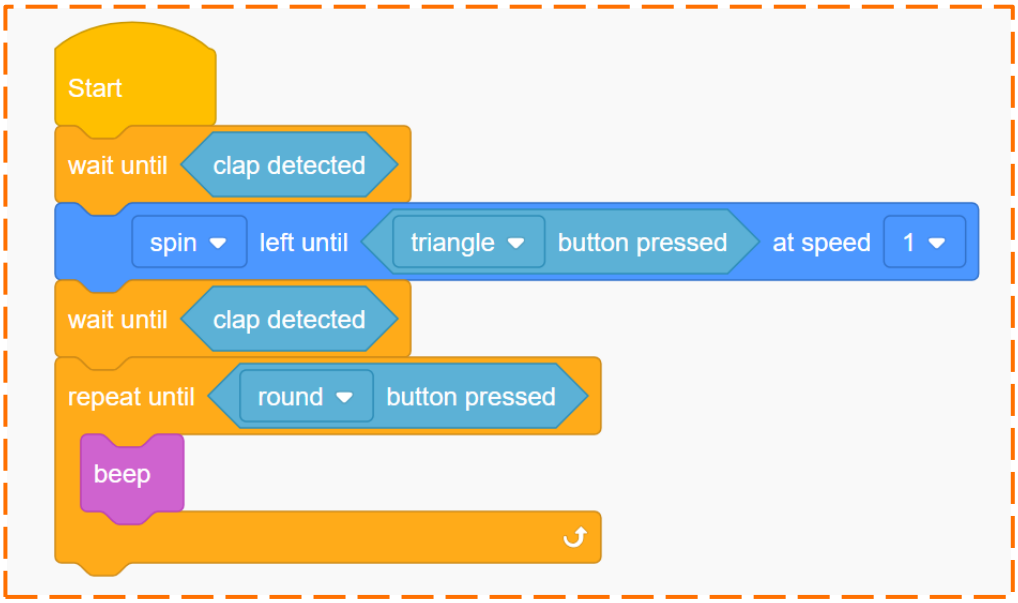
Remember that in programming, an event is something that happens outside of the program code that affects how the program runs.

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.

Name_____

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?

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

   _____

4. 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?

_____

_____

_____

_____

_____

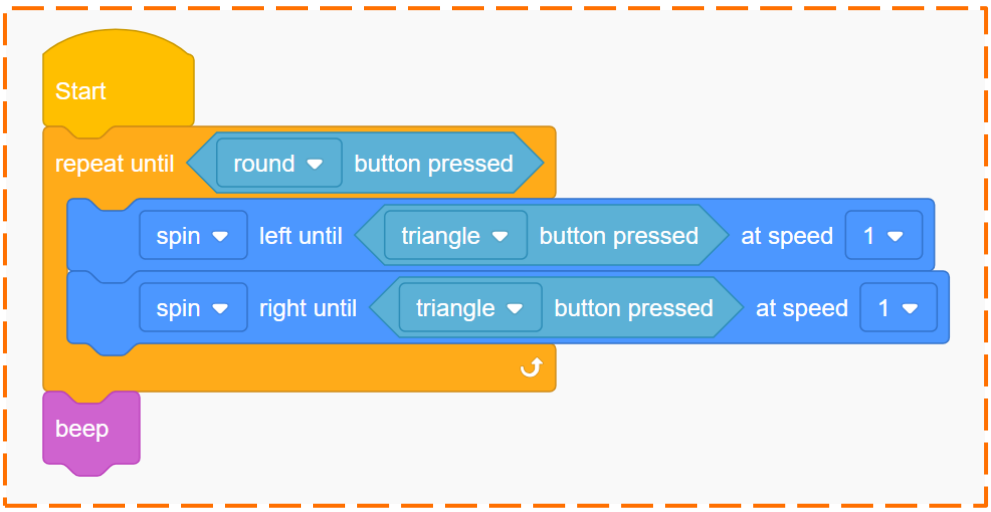
_____

Name_____

U4-1.1a 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.

Name_____

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.

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

_____

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

_____

_____

3. 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 #1:

"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 #2:

"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?

Name_____

What's going on here?

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

Problem #1:

"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!

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

_____

_____

_____

_____

_____

Name_____

Now, let's look at the second problem:

Problem #2:

"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!

5. 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_____

U4-1.2 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:

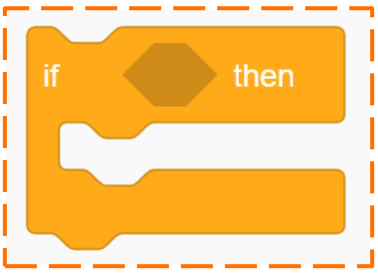
- IF it is cold outside, THEN I put on a jacket before I leave the house.
- IF I am hungry after school, THEN I eat a snack.

1. Think about a conditional you encounter in your daily life. Write it using the 'if__, then___' formula.

   IF _____

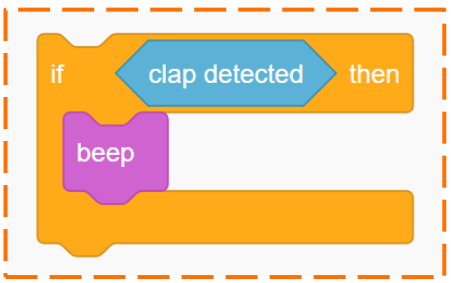
   THEN _____

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**.



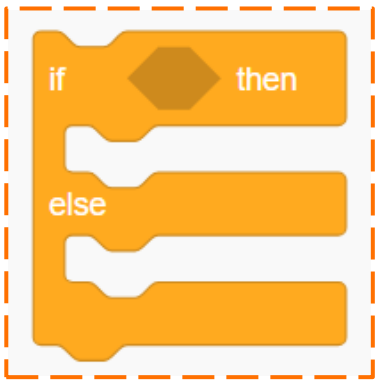
Jargon buster

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.

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 activity sheet U4-1. This activity sheet has a special treasure map that can only be solved with if-else statements.

Name_____

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



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'

2. The Orb of Marshmallow

Begin at the start spot.

Repeat 3 times:

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

Where is the Orb of Marshmallow? _____

3. The Cloak of Pancakes

Begin at the start spot.

Repeat 3 times:

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

Where is the Cloak of Pancakes? _____

4. The Omelette of Space Eggs

Begin at the start spot.

Repeat 2 times:

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

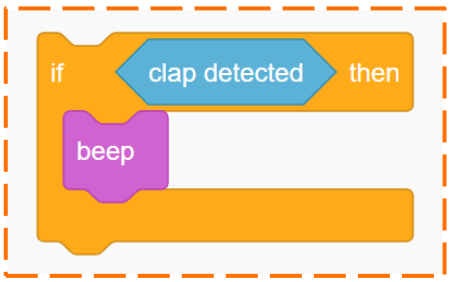
IF the number < 7, THEN go left. ELSE go right.

Where is the Omelette of Space Eggs? _____

Name_____

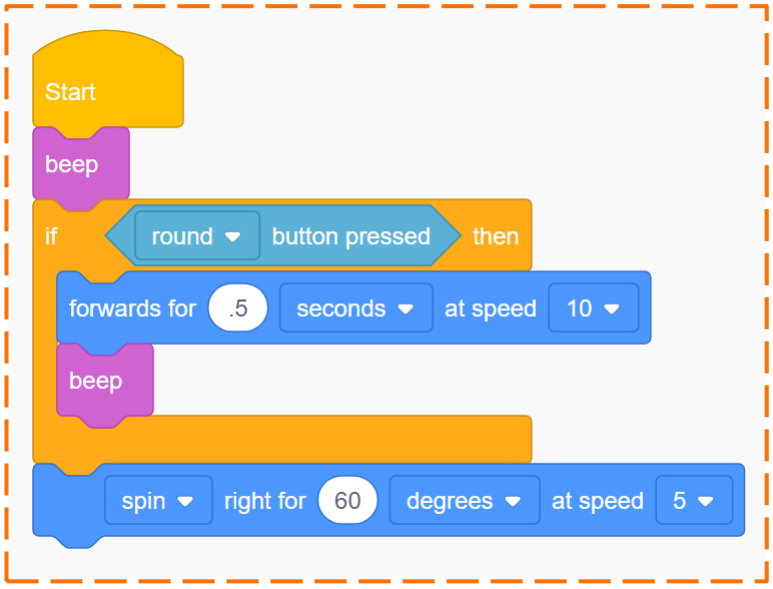
U4-1.3 Let's explore if statements and sequence

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:



1. 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.

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

_____

_____

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

_____

_____

_____



Why is that?

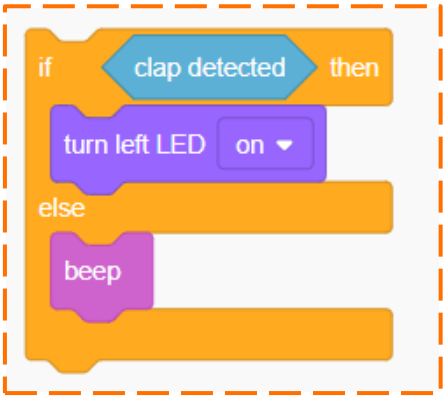
Remember, all programs move through the code step-by-step in sequential order. When the program gets to an if block, it checks to see if the condition has been met. If it has, the program runs the code inside the block. If the condition has not been met, then the program skips the code in the if block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

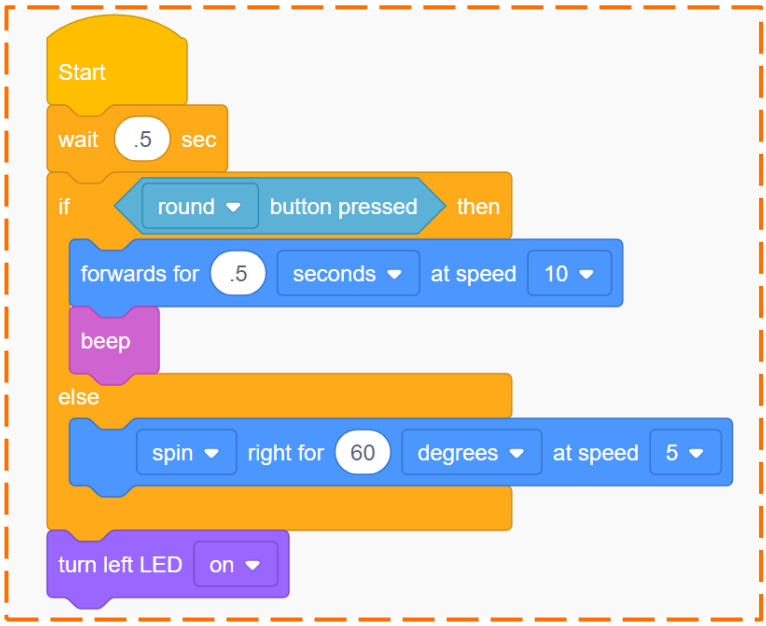
If the condition is met, the robot will do the code inside the 'if' part of the block. If the condition is not met, the robot will do the code inside the 'else' part of the block:



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.

Name_____

Look at this EdScratch program:



4. 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?

_____

_____

_____

_____

5. 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_____

U4-1.4 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!

Task 1: 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.

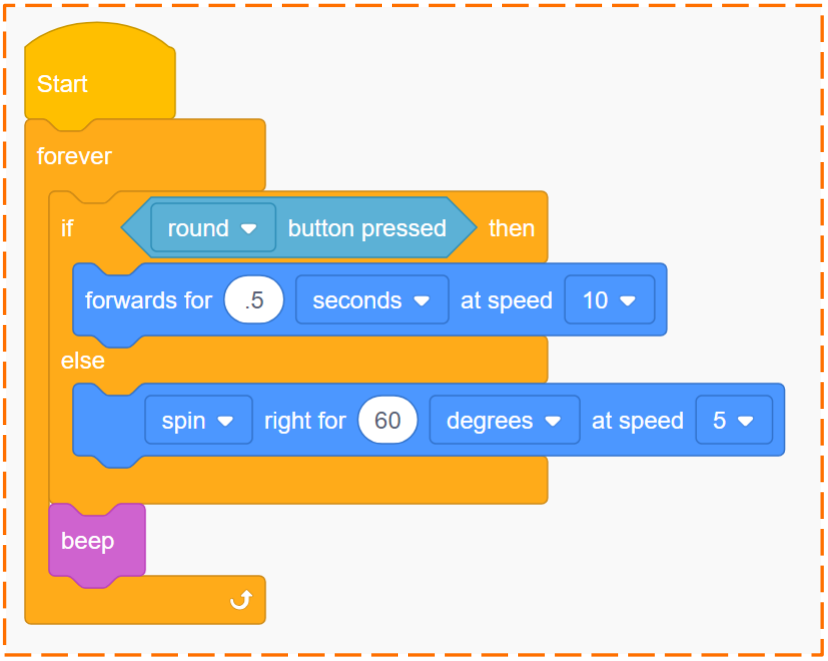
If a block contains conditional code, the program first checks if that condition has been met. The result of the condition check determines what actions the program does next.

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:



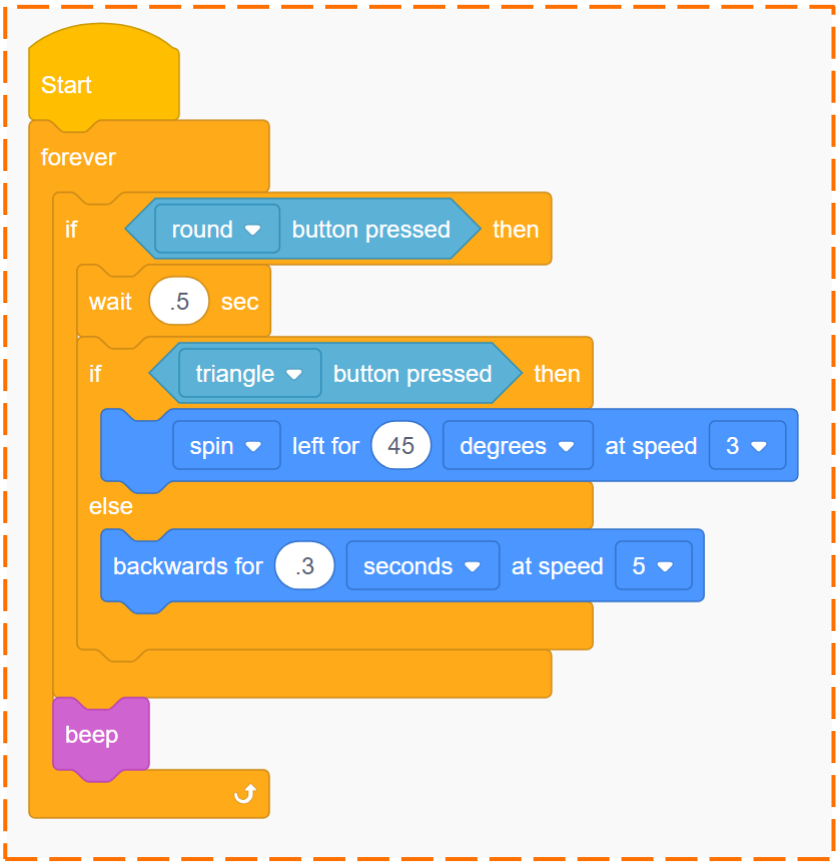
1. If you run program 1 but never press the round button, what will happen? Why?

_____
_____
_____
_____
_____
_____
_____

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

_____
_____
_____
_____
_____
_____
_____

Program 2:



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

_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____

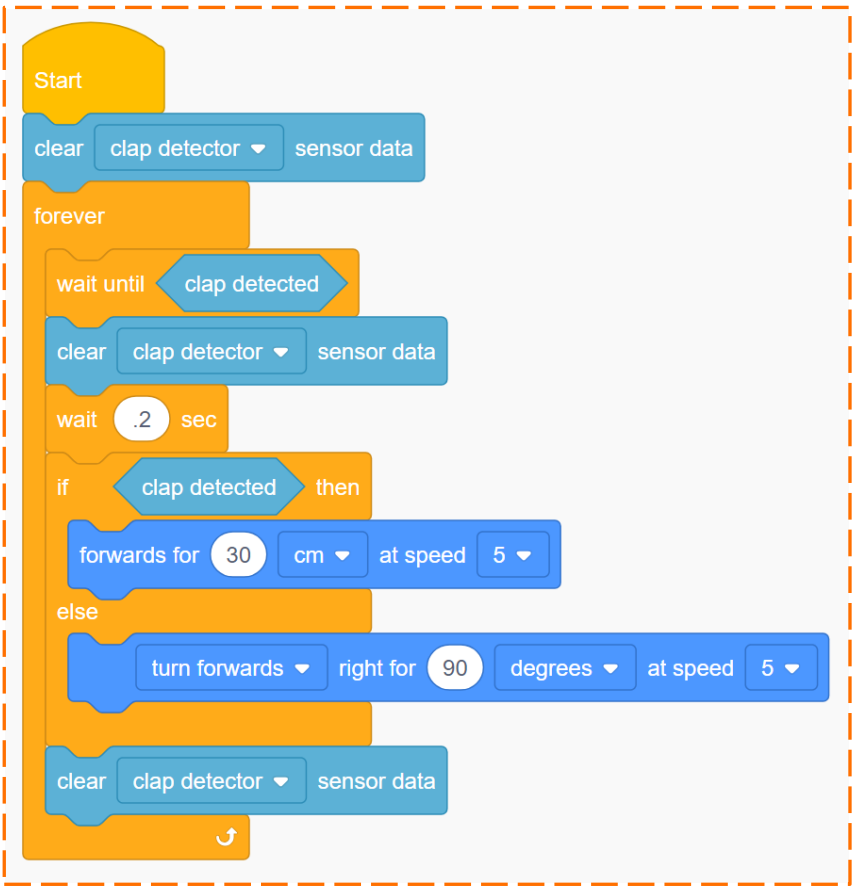
Choose one of the two programs to write in EdScratch. Download the program and test it out in your Edison robot. Experiment to see how these programs with nested 'if' statements work.

Name_____

Task 2: 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.



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_____

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

Write the clap-controlled driving program in EdScratch.



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!

Download the program to your Edison robot and run it. Experiment to see how the program makes Edison respond to claps.

4. 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?

_____

_____

_____

_____

_____

_____

Name_____

U4-1.4a Challenge up: Build a pulley

A pulley is a device which consists of a wheel with a grooved rim over which a rope or chain is pulled in order to lift heavy objects. You can use Edison to create a programmable pulley and write a program in EdScratch to control how the pulley operates!

What to do

Build a pulley system using an Edison robot. You can use EdCreate parts or any other materials you like. You will also need to create a program to operate the pulley using EdScratch.

Your pulley system should use Edison's motor outputs to control the pulley. The motor should move the pulley in one direction (up or down) if the round button is pushed and in the other direction if the triangle button is pushed.

Write your program in EdScratch and test it out with your pulley design.

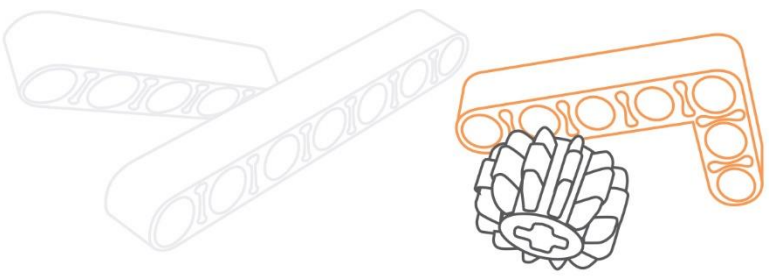
You may need to change your design, your program or both to get your pulley to work. That's okay! Experiment to see what works.



Hint!

Try stacking or nesting 'if' statements in your pulley program to get the pulley to respond to the different button pushes.

1. How did it go? Write about what happened in your pulley project. What went wrong? How did you overcome any problems you encountered? What was the best part of the project? Why was it the best part?



Name_____

U4-2.1 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.

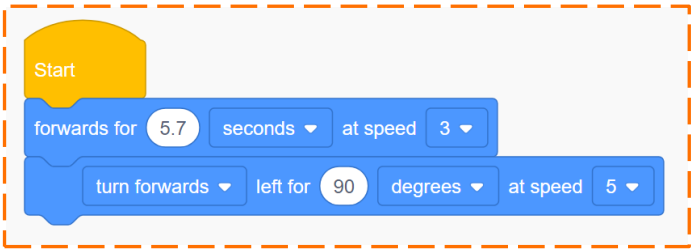
When you plan out your program using pseudocode, you don't need to write everything out in detail or worry about exactly how it will end up looking when you code it. You just need to write a simple version of your plan that makes it easy to follow the flow of the program.

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

```
drive forwards
turn left
```

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.

Name_____

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.

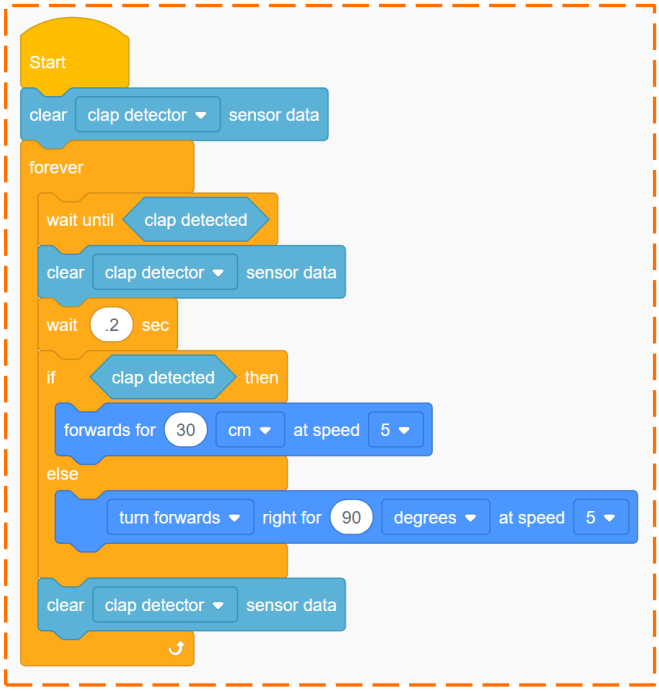
Using pseudocode is especially helpful to plan programs that use control structures like loops or conditionals. You should indent actions that are inside of loops or conditionals to show that this code is inside of the loop or 'if' statement. Organising your pseudocode in this way makes it a lot easier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

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

Pseudocode:

```
clear clap data
forever
        wait until clap
        clear clap data
        wait
        if clap
                drive forward
        else
                turn right
        clear clap data
```

EdScratch program:



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

Try it out!

Let's try reading some pseudocode instructions. Use activity sheet U4-2 and follow the pseudocode instructions to find the answers to the questions.



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.

Name_____

1. 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
```

2. Follow the pseudocode. Where will the program end? _____

```
Start on H facing east
repeat 3 times
        forwards 1
        if living thing
                right 90 degrees
        else
                left 90 degrees
backwards 1
```

3. Follow the pseudocode. Where will the program end? _____

```
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
```



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!

Name_____

U4-2.1a Change it up: Find the answer

When you first use pseudocode, it might seem a bit awkward. Once you get familiar with pseudocode, however, planning out your programs in pseudocode will save you a lot of time!

What to do

For this activity, you will need to work with a partner to practice writing and following pseudocode instructions. Write some pseudocode instructions using activity sheet U4-2 for your partner to follow. The first line of your pseudocode should tell your partner where to start, including which direction to face.



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 the plan so that your partner can follow it to find the answers.

1. Write your pseudocode. Be sure and test it out before you exchange with your partner.

Mini challenge!

Control structures, like loops and conditionals, make programs more powerful and coding more fun! Can you include at least two different control structures in your pseudocode instructions?



Hint!

Look at the Control category in EdScratch for some ideas on what control structures to use.

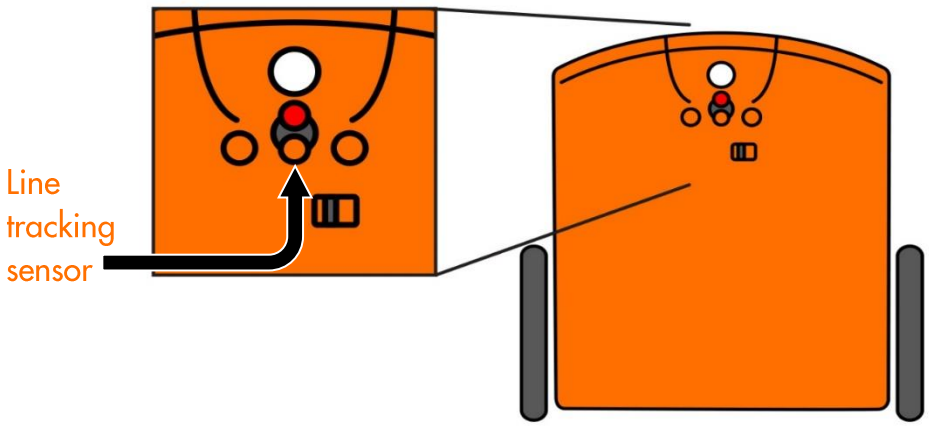
Name_____

U4-2.2 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.

Task 1: 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 activity sheet U4-3 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.

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

_____

_____

_____

Name_____

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?

2. Will Edison see a red surface as reflective or non-reflective? What about a blue surface? Or green? Use activity sheet U4-3 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	

Task 2: 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

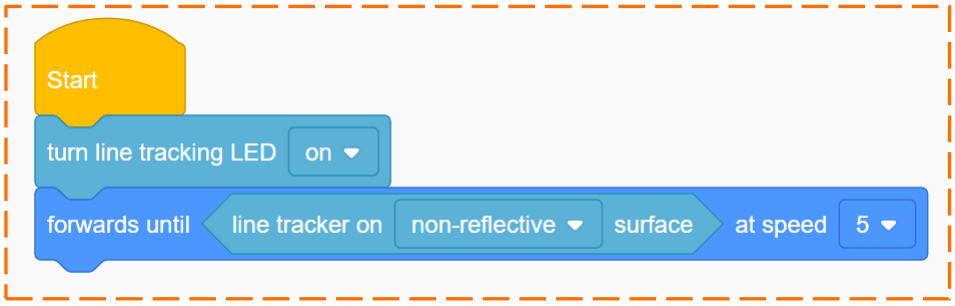
Inputs are the information and instructions that you give a computer. When you write a program for your Edison robot, you are telling the robot what you want it to do by giving it inputs. Edison's microchip then processes the information to tell the robot what to output as part of the input-process-output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

Name_____

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 activity sheet U4-3 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 activity sheet U4-3 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!

Name_____

U4-2.2a 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.

1. Write your pseudocode.

_____

_____

_____

_____

_____

_____

Use your pseudocode as a guide to help write your program in EdScratch. Download it to your Edison robot and test it using activity sheet U4-4.



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_____

U4-2.3 Let's explore algorithms

Edison's line tracking sensor can detect if the surface below the robot is reflective or non-reflective. You can use this sensor to get Edison to behave in different ways, such as driving until it detects a black line, then stopping. You can also use this sensor to program Edison to follow a black line, even if you don't know what that black line looks like. To do this, you first need to create an **algorithm**.



Jargon buster

An **algorithm** is a broad set of instructions to solve a set of problems. An algorithm lays out a process or a set of rules to be followed in order to solve any problem in the set.

Computer programs often use algorithms, but programs and algorithms are not the same thing. Remember, a computer program is a collection of instructions that tell a computer to perform a specific task. An algorithm lays out the logic for how to solve a whole set of problems, not just one specific task. You can write a computer program that uses an algorithm, but not all computer programs are algorithms.

Algorithms are really helpful because using an algorithm lets a person, or a computer, solve a whole set of problems, even if you don't know every detail.



Why is that?

Let's say you want to teach your friends how to make fruit pies. If you know that all of your friends have apples, you can just write down one recipe for apple pie.

Not all of your friends might have apples, however. What if one of your friends has blueberries, another has cherries, and a third has apples? They cannot all follow the apple pie recipe. You would need to write a separate recipe for each different fruit.

What if you don't know what fruit each of your friends has? How could you teach them to make fruit pies?

No matter what fruit they have, all of your friends need to follow the same basic instructions: make the dough, fill the pie with the fruit, then bake the pie.

This new set of instructions is an example of an algorithm.

In computer programming, we often want to create instructions for a computer to follow in order to solve a whole set of problems. By using an algorithm, we can write a program that will let the computer solve any problem in the set. Without an algorithm, we would need to write a new program for every single problem individually.

Name_____

Task 1: Follow a black line

You know that you can use Edison's line tracking sensor to detect black (non-reflective) and white (reflective) surfaces. We can create an algorithm that uses this sensor to get Edison to follow any black line.



Why is that?

Let's say you draw a black line for Edison to follow. To get Edison to follow your line, you could write a program which makes Edison drive the exact path of the line. If you make a new line, however, you will need to write a whole new program for that new line.

Instead, you can create an algorithm.

This algorithm will solve a set of problems: 'follow any black line'. Any specific line you make for Edison to follow is a new problem inside this set.

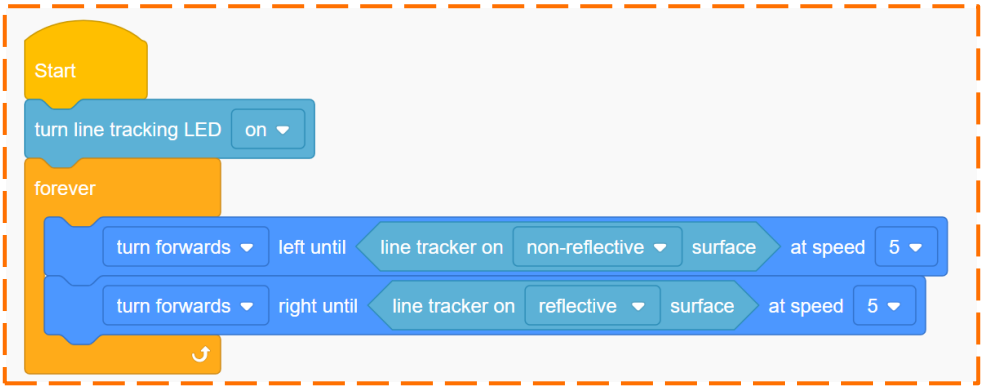
Using the algorithm to guide the logic, you can then write a program which will work for all of the problems in the set. This way, a whole new program isn't needed for each new problem.

You can plan out an algorithm using pseudocode, just like you do when you plan out a program. Here is an algorithm in pseudocode that will allow Edison to follow any black line:

```
turn line tracker on
loop forever
        drive forwards left until the robot detects a black surface
        drive forwards right until the robot detects a white surface
```

This algorithm says that the robot should drive forwards to the left until the line tracking sensor is on a non-reflective (black) surface. Then the robot should drive forwards to the right until the line tracking sensor is on a reflective (white) surface. The robot should keep doing this behaviour forever.

This algorithm can be used to write a program in EdScratch:



Can you see how the algorithm's logic is inside the program?

Name_____

Write the line-following program in EdScratch and download it to your Edison robot. Use activity sheet U4-4 to test the program. Remember to start Edison with the line tracker on a white surface, not directly on the black line.

1. How does the robot move when you run the program? Look at the program and think about the logic in the algorithm. Why does the robot move that way?

_____

_____

_____

_____

Task 2: Follow a different black line

The program you wrote uses an algorithm that is designed to solve any problem in the set of problems: 'follow any black line'. That means this same program should let Edison follow any black line you make!

Make your own line to test. Use a black marker on white paper or make a line using black tape on the floor or a desk. Run the program in Edison to test out your line. Can Edison follow your line?

2. Was Edison able to follow your line? If you had any problems, describe them. What do you think caused the problems?

_____

_____

_____

Name_____

U4-2.3a Challenge up: There's more than one way to follow a line

To get your Edison robot to follow any line, you need to use an algorithm.



Don't forget

An algorithm is a broad set of instructions to solve a set of problems. An algorithm lays out the logic for how to solve a whole set of problems, not just one specific task.

Think about the logic in the algorithm that will allow Edison to follow any black line. At its simplest, what is it saying?

The broad instructions to solve the 'follow any black line' set of problems are: using the line tracker, while moving forward, go one way if on black and the other way if on white. Here's what this very basic algorithm looks like in pseudocode:

```
use the line tracker
forever
        if the robot detects black, drive forward one way (left or right)
        if the robot detects white, drive forward the other way (right or left)
```

If you were planning out an algorithm to then code, you might write this same logic slightly differently, and you would probably add in some of the details, like in this example:

```
turn line tracker on
loop forever
        drive forwards left until the robot detects a black surface
        drive forwards right until the robot detects a white surface
```

Because the logic is the same in both examples, any programs you make using either example will follow the same logic. Even if the programs look different, they will still solve the 'follow any black line' set of problems.

Just how many different ways could you write a program that will solve the 'follow any black line' set of problems? More than you might think!

What to do

Your challenge is to write at least two different programs that use the basic logic of the 'follow any black line' algorithm.

Name_____

You will need to plan each program using pseudocode, then code it in EdScratch.



Hint!

Pseudocode helps us plan, comments help us debug. Adding comments as you code will help you check your logic and keep track of your thinking, so you can find and fix any issues you encounter when you test your program.

Download each of your programs one at a time. Test each program using activity sheet U4-4 or make your own line to use as a test space. Both of your programs need to use Edison's line tracking sensor and should be able to follow any black line.



Hint!

Think about how you can apply the logic of the algorithm in EdScratch. Look at different conditionals, including until blocks, if blocks and if-else blocks. Don't forget about the Events category too!

What do your two programs look like? Either write pseudocode, download and share your program files or write an explanation of each of your programs.

Program 1:

Program 2:

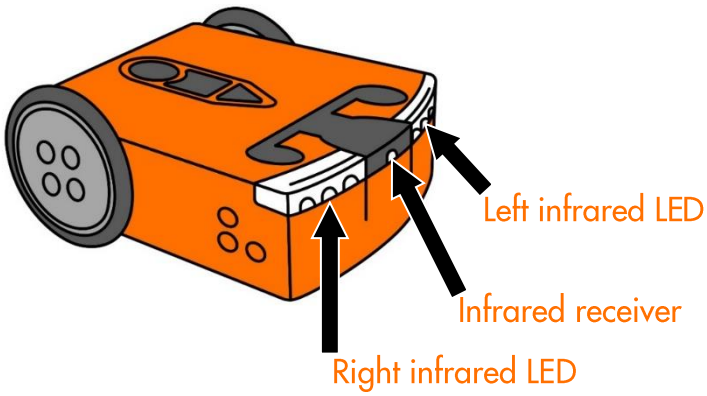
Name_____

U4-2.4 Let's explore Edison's obstacle detection

Edison robots have different sensors that can detect different things. One of these sensors is the infrared light sensor which we can use to detect obstacles.

Task 1: Meet Edison's infrared light sensor

Edison's infrared light sensor is the sensor that lets Edison emit and detect infrared light. The sensor is made up of three parts all located across the front of Edison: two infrared LEDs (one on the right and one on the left) plus an infrared receiver in the middle. Look at your Edison robot. Do you see the different parts of the sensor?



Just like Edison's two red LEDs, the two infrared LEDs can emit light. However, because it's infrared, you won't be able to see it.



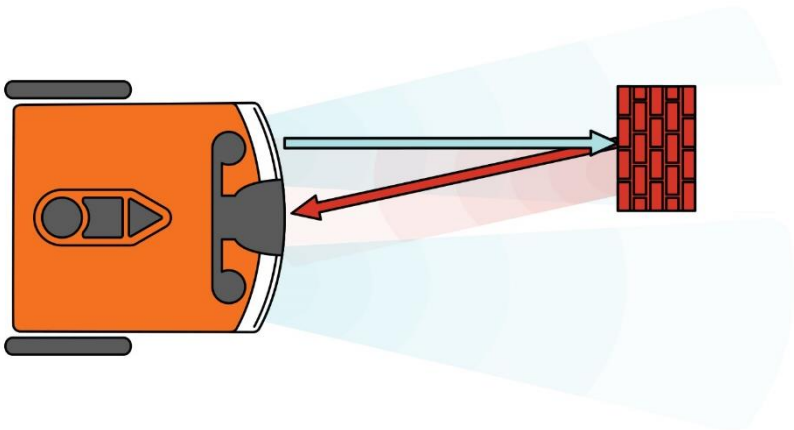
Why is that?

There is a wide range of light. People can see some of this range, but not all of it. Infrared light (which is also called IR light) is not visible to humans.

Even though we cannot see it, people use infrared a lot. For example, infrared light is used in TV remote controls. It's how the remote tells the TV to change the channel or turn up the volume!

Unlike us, Edison can detect infrared light using the infrared receiver on the front of the robot. One way we can use Edison's infrared light sensor is to detect obstacles.

Edison can emit infrared light from the two infrared LEDs. If that infrared encounters an obstacle, like a wall, the light is reflected back towards Edison. Edison's infrared receiver detects the reflected light, telling Edison that there is an obstacle.



Depending on where the obstacle is, infrared light will bounce back from the left LED, the right LED or both. The reflected light tells the robot where the obstacle is located.

Name_____

Task 2: Forward until obstacle

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.



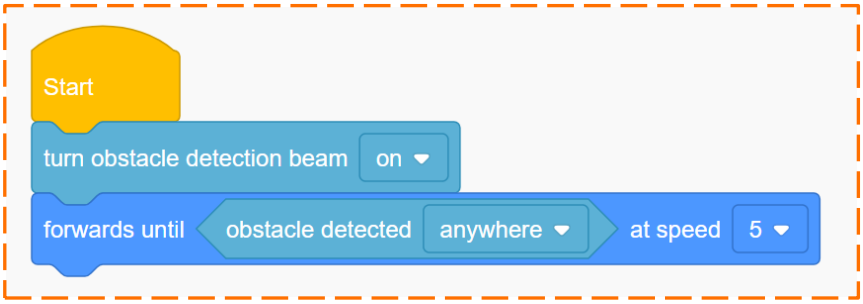
Don't forget

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 infrared sensor in a program which tells Edison to drive until it detects an obstacle. 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 on the obstacle detection beam.



Why is that?

Some of Edison's sensors are always on and checking for events. The infrared receiver is actually always on, but the infrared LEDs are not.

For Edison to detect obstacles, you need to include code in your program to turn on the obstacle detection beam. This turns on the infrared LEDs and tells the infrared receiver to look for reflected infrared light bouncing back to the robot. Just turning the obstacle detection beam on isn't enough, however. You also need code to tell the sensor what event to check for (in other words, where it should be looking for an obstacle) and what to do if that event is detected.

Name_____

Write the program in EdScratch. Download the program and test it using your Edison robot. You will need an obstacle to use in your test as well.

Some obstacles will work better with Edison than others. If an obstacle is too small or doesn't reflect enough infrared light, Edison cannot detect it. Select an object that is opaque but not too dark (don't choose black objects) and at least as tall as Edison. For this program, the wall of the room would be a good obstacle.

Test the program to see how it works.



Hint!

If your robot isn't detecting your obstacle, try a different object. If it still isn't detecting the obstacle, you might need to calibrate the obstacle detection. Ask your teacher for the special barcode to calibrate obstacle detection.

Task 3: Detect and avoid

Instead of just stopping when it detects an obstacle, you can get Edison to avoid any obstacle it detects and keep driving. To do this, you need to create an algorithm to solve the set of problems: 'detect and avoid any obstacle'.

1. Use pseudocode to write a detect-and-avoid algorithm.

Once you have your algorithm, write an EdScratch program for Edison that uses the logic from that algorithm. Download your program and test it out using Edison and some obstacles.

2. Even when professional computer programmers write programs, they run into problems. Describe one problem you had creating your detect-and-avoid algorithm or program. What did you do to solve the problem?

Name_____

U4-2.4a Change it up: Faster, faster, smash?

Edison's obstacle detection works by sending out infrared light from the robot's two infrared LEDs. If there is an obstacle, the light bounces off of the obstacle and reflects back to Edison where the robot's infrared receiver detects it. We can program Edison to detect the reflected IR light and react to that event.

Robots can process information incredibly fast, but the task of receiving inputs, processing the information and generating an output still takes some amount of time. There are multiple steps in the Edison robot's obstacle detection process: Edison emits IR light, the light encounters an object, the light bounces off of the object, the robot's infrared receiver detects the reflected light, and, finally, the robot reacts to that event. Just how long does this obstacle detection process take?

What to do

Write a program to get Edison to drive forward until the robot detects an object. Edison should stop driving as soon as it detects the object so that the robot doesn't hit the obstacle.



Don't forget

Whenever you want to use the infrared sensor to detect obstacles, you need to turn on the obstacle detection beam.

Download your program and test it using Edison and an object that Edison should be able to detect. Then adjust the speed input parameter in your program. Test different speeds with Edison to see what happens with faster and slower speeds.

1. What happens when you run your obstacle detection program with a very fast speed?

_____

_____

2. In programming, we sometimes have to balance different features to achieve the best result. This is known as a trade-off. Describe the trade-off between Edison's speed and the robot's ability to detect obstacles.

_____

_____

_____

_____

Name_____

U4-2.4b Challenge up: If line, go right. If obstacle, go left

Edison robots have different sensors that can detect different things. You can use multiple sensors in a single program, getting the robot to react to different types of events.

What to do

Write a program so that your Edison robot moves through a grid, checking each new section of the grid for any non-reflective (black) surfaces or obstacles. Your program should tell Edison that if the robot detects a black surface, it needs to turn right, but if it detects an obstacle, the robot should turn left instead. Test your program out using activity sheet U4-5. See if you can write a program so that your Edison robot starts on the outline, then uses its sensors to get to the parking zone goal.



Hint!

- Sensors like the obstacle detection beam and line tracker need to be turned on to work. Your program also needs to tell Edison what event the sensors should look for and how to react to that event.
- Some of Edison's sensors, including obstacle detection, generate and store data when they detect specific events. It is good coding practice to clear the sensor data, especially 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 wise to clear the sensor data at the start of a program, just in case the robot has old data stored from a previous program.
- Pseudocode helps us plan, comments help us debug. Use them to help you plan and test your program!

Mini challenge!

Can you make your own 'grid maze' for Edison to solve using its sensors? Design a layout that has a start, a goal, and both black surfaces and obstacles to detect. Write a program that uses the robot's sensors to get Edison to the goal.

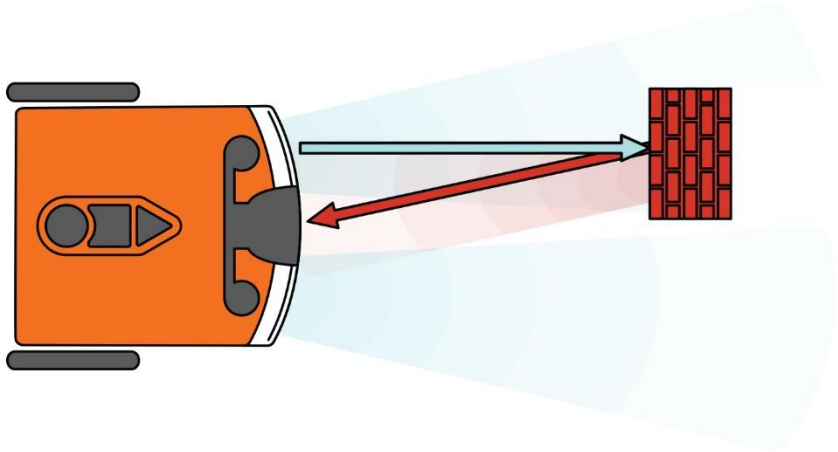
Name_____

U4-2.4c Change it up: Where is the obstacle?

Edison can emit infrared light from the two infrared LEDs on the right and left of the robot. If that infrared light encounters an obstacle, like a wall or your hand, the light is reflected back towards Edison. Edison's infrared receiver detects the reflected light, telling Edison that there is an obstacle.

Depending on where the obstacle is, infrared light will bounce back from the left LED, the right LED, or both. The reflected light tells the robot where the obstacle is located.

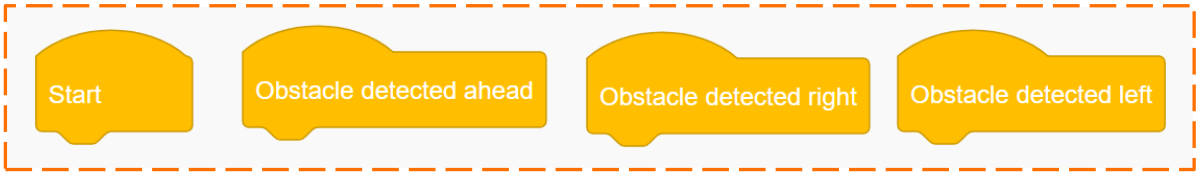
You can program Edison to react with different outputs depending on where the robot detects the obstacle.



What to do

Create a program in EdScratch to get Edison to detect obstacles but respond with a different output depending on where the obstacle is located: on the right, on the left, or straight ahead.

In this challenge, you need to use blocks from the Events category. Your program should contain all of the blocks in this picture:



What the robot outputs for each of the different events is up to you.



Hint!

You probably don't want to use Edison's motors as outputs in this program. Why? If the robot moves, it will change position relative to the obstacle. That could become really confusing!

What outputs could you use instead? What could you do to make your outputs help communicate where the robot detected the obstacle?

Download and test your program by putting objects to the right, then the left and then straight in front of Edison.

Name_____

U4-2.4d Challenge up: 3D maze

You can use Edison's obstacle detection beam to program the robot to detect and respond to obstacles. Can you use obstacle detection to get Edison to drive autonomously through a maze?

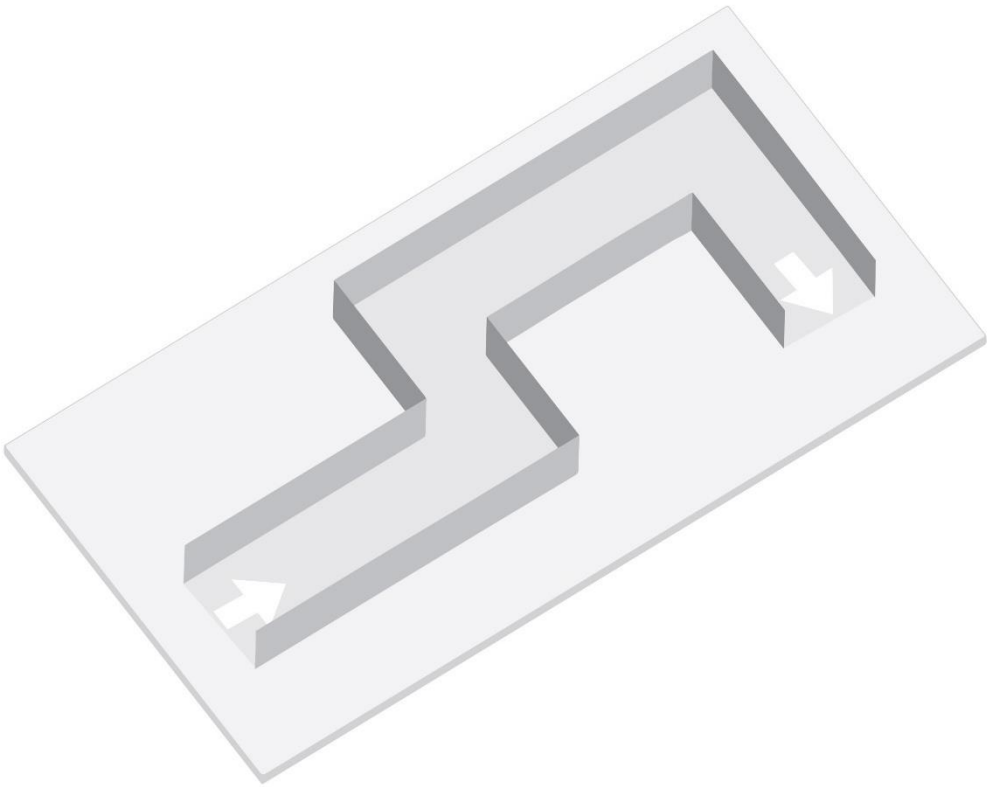
What to do

In this challenge, you will need to build a 3D maze for Edison to navigate using obstacle detection. The walls of your maze will need to be high enough so that Edison can detect and react to them. You also need to write a program to get Edison through the maze using obstacle detection.



Hint!

- Some of Edison's sensors, including obstacle detection, generate and store data when they detect specific events. It is good coding practice to clear the sensor data, especially when you use sensor events in conditionals nested inside loops, plus at the start of a program.
- Remember that Edison can detect where an obstacle is located: to the right, to the left, or straight ahead. This may help you create a successful program.
- Pseudocode helps us plan, comments help us debug. Use them to help you plan and test your program!



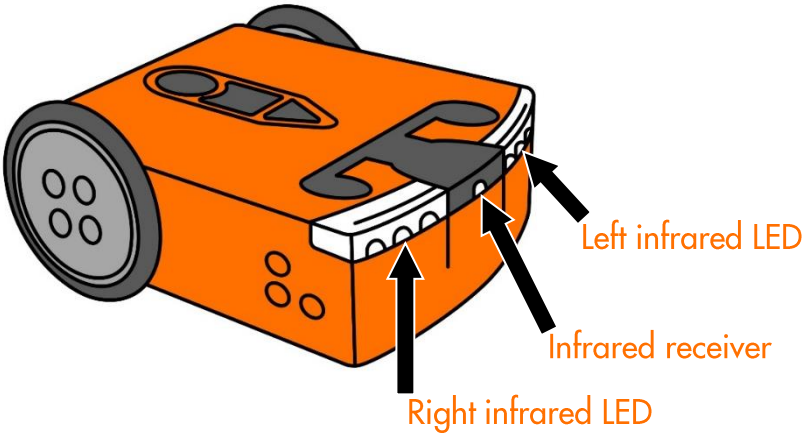
Name_____

U4-2.5 Let's explore messaging with Edison

Edison robots have an infrared light sensor which can be used to detect obstacles. We can also use this sensor in another way: to send and receive infrared messages.

Task 1: Edison and infrared messages

Edison's infrared (IR) light sensor is the sensor that lets Edison emit and detect infrared light. The sensor is made up of three parts all located across the front of Edison: two infrared LEDs (one on the right and one on the left) plus an infrared receiver in the middle.



In obstacle detection, we use the infrared LEDs to emit infrared light and the infrared receiver to check for any of that light that has been reflected off of objects back to Edison. We can also use the infrared receiver to detect infrared light from other sources, like a TV or DVD remote control, or another Edison robot. Using the infrared receiver this way lets you send or receive messages using your robot.



Why is that?

TV remote controls use infrared light to send 'messages' to the television set. These messages aren't messages like you might send to a friend, however. The message is a pre-set signal that activates a specific function. For example, one message tells the TV to turn up the volume and a different message tells the TV to turn the volume down.

Each button on a remote sends a different message using infrared light!

We can also use infrared light to send and receive messages using Edison robots. One robot can send out an infrared message using its two IR LEDs which another robot's IR receiver can detect.



Don't forget

Infrared is sometimes abbreviated as IR. In EdScratch, IR message blocks are blocks that relate to Edison's infrared messaging using the infrared LEDs and infrared receiver.

Name_____

Task 2: Message received

To use messaging with Edison robots, you always need at least two robots, one to send out the IR messages and one to detect and react to the IR messages.

For this activity, you need a partner or a group. One robot is going to send out a message. All the other robots need to wait until they detect a message. Once the receiving robots detect a message, each robot should react by dancing!

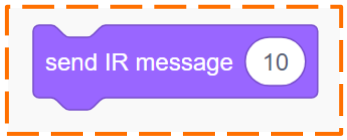


Why is that?

You can use Edison's infrared sensor in a program which tells Edison to detect an IR message event. You also need another Edison robot to run a program which sends out an IR message, however, or your first robot will have nothing to detect!

The 'sending IR message' program

To send an infrared message with your Edison robot in EdScratch, you need to use this block:



1. The send IR message block is in the LEDs category of blocks in EdScratch. Why is that?

_____

_____

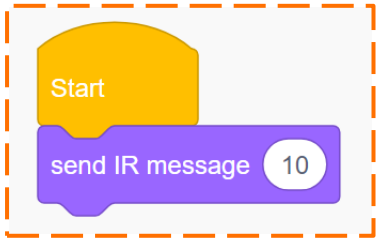
The send IR message block has an input parameter which you can set to send a specific message.



Why is that?

Just like a remote control can send different messages to a television, you can send different messages from your Edison robot. In EdScratch, you can change the message by changing the value of the input parameter in the send IR message block. The send IR message block has an input range of 0 to 255. In other words, Edison can send and receive 256 different 'messages'. Imagine if a remote control could send that many messages. That would be a LOT of buttons!

Sometimes we want to create programs telling Edison to react a specific way to a specific message. Other times, like in this activity, we just want Edison to react if any IR message is detected. The robot that is sending out the IR message for the other robots to detect needs to use this base program:

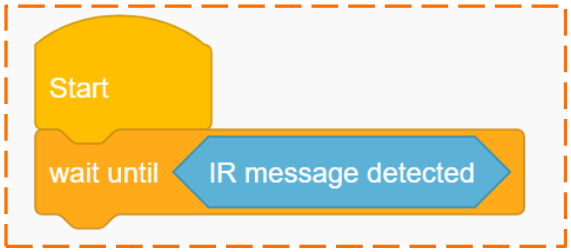


Name_____

For this activity, you can set the input parameter in the send IR message block to whatever value you like. You also need to write more code for this robot to do after it sends out the IR message. Use at least two different types of outputs including blocks from the Drive category to make your robot dance after it has sent out the IR message.

The 'receiving IR message' program

All of the robots which are going to detect the IR message need to have the same base program:



This program uses a block from the Sensing category in EdScratch with the wait until control block. You don't need a block to turn on the IR message detection.



Why is that?

Some of Edison's sensors are always on and checking for events. The infrared receiver is one of these 'always on' sensors, so you don't need code to turn it on. You do need to tell it what kind of IR event to detect, however, and what to do if that event is detected.

In the base program example, the code tells Edison to look for an IR message detected event. This event will be triggered if the robot detects an IR message, no matter which message it detects. That's why it doesn't matter what value the 'sending IR message' program uses!

Use the base program and write more code to tell the robot what to do once it receives an IR message. Use at least two different types of outputs including blocks from the Drive category to make your robot dance once it receives the IR message.

Once all of the robots have their programs downloaded, it's time to party!



Hint!

Press the play (triangle) button on all of the receiving robots first, then on the sending robot. If you run the sending robot's program before the receiving robots are ready, the receiving robots will miss the message!

Name_____

U4-2.5a Change it up: Remote-controlled flag machine

Do you remember Edison's special barcodes? Edison comes with some programs already loaded which we can get the robot to access by using different barcodes. There is also a set of barcodes we can use along with Edison's infrared sensor to pair the robot with a TV or DVD remote control. We can then write an EdScratch program telling Edison what to do when it detects a specific remote control code.



Why is that?

The programmable TV remote codes are a special type of barcode. These barcodes let Edison store a specific code that the robot can reference later.

Unlike other Edison barcodes, these barcodes don't activate a program in Edison. Instead, the barcodes tell Edison to store the next remote-control button press that the robot detects as a specific remote-control code.

To use a programmable TV remote code, you first need to scan the barcode with your Edison and pair it to a button of your choice on a remote control. For example, you might scan remote code #1 and match that to the 'volume up' button on your remote. Once paired, whenever you push that button on the remote, Edison will reference the IR signal as 'remote code #1'.

You can then write a program in EdScratch using that same code as an input. Your EdScratch program will need to tell Edison what remote codes to detect and what to do if the robot detects a specific code.

By using remote codes as events in EdScratch programs, you can build robotic creations that you can control with a TV remote control! Let's try building a flag machine using Edison's motor outputs which you can control using a TV remote.

What to do

The goal of this activity is to build and program a flag machine that you can use to help make studying for a test more fun.

Your creation should let you control a two-sided flag which has the word 'Yes' on one side and the word 'No' on the other side of the flag. You will need to write a program in EdScratch that will let you show the 'Yes' side of the flag when you press one button on a TV remote and the 'No' side when you press a different button. When your flag machine is finished, work with a partner and take turns quizzing each other. Use your flag machine to signal if your partner answered your question correctly or not!

You need to design how your flag machine will work using Edison's outputs and write a program in EdScratch to control it using remote codes.

Name_____

The first thing to do is pair your robot to a remote control. Look at the programmable TV remote codes on activity sheet U4-6. For this activity, you will probably need to use two of these remote codes.

To pair your robot with one of these codes, you need to follow these steps:

1. Place Edison facing the barcode on the **right side** of the barcode.
2. Press the record (round) button **three times.**
3. Wait while Edison drives forward and scans the barcode.
4. Choose a button on your remote control that you want to match to that remote code. Point the remote at your robot and press your selected button. Be sure and use different remote-control buttons for each different remote code.



Hint!

You need Edison to be able to drive in order to scan a barcode, so be sure to scan whichever barcodes you need before you start building your robotic creation!

You also need to write a program in EdScratch that tells Edison what to do if it detects the different remote codes. Your program should tell Edison to show one side of the flag if you press the first remote-control button your paired and the other side of the flag if you press the other button.

Once you have built and programmed your creation, try it out!



Name_____

U4-2.5b Challenge up: Build and control the EdCrane

Edison has a set of barcodes we can use along with Edison's infrared sensor to pair the robot with a TV or DVD remote control. We can then write an EdScratch program telling Edison what to do when it detects a specific remote control code.



Why is that?

The programmable TV remote codes are a special type of barcode. These barcodes let Edison store a specific code that the robot can reference later.

Unlike other Edison barcodes, these barcodes don't activate a program in Edison. Instead, the barcodes tell Edison to store the next remote-control button press that the robot detects as a specific remote-control code.

To use a programmable TV remote code, you first need to scan the barcode with your Edison and pair it to a button of your choice on a remote control. For example, you might scan remote code #1 and match that to the 'volume up' button on your remote. Once paired, whenever you push that button on the remote, Edison will reference the IR signal as 'remote code #1'.

You can then write a program in EdScratch using that same code as an input. Your EdScratch program will need to tell Edison what remote codes to detect and what to do if the robot detects a specific code.

By using remote codes as events in EdScratch programs, you can build robotic creations that you can control with a TV remote control!

What to do

In this activity, you will build and program the EdCrane. The EdCrane is a remote-controlled crane with a magnetic 'hook' that you can use to lift and move small metal objects.

The first thing to do is pair your robot to a remote control using the programmable TV remote codes on activity sheet U4-6. For this activity, you will need to use four remote codes to tell the EdCrane to do one of four different actions.

1. Plan out your remote codes and remote-control buttons:

EdCrane action	Remote code	Remote control button
Spin the magnetic spool clockwise		
Spin the magnetic spool counter-clockwise		
Spin the crane clockwise		
Spin the crane counter-clockwise		

Name_____

Using your plan as a reference, pair your robot to each code. To pair your robot with one of these codes, you need to follow these steps:

1. Place Edison facing the barcode on the **right side** of the barcode.
2. Press the record (round) button **three times.**
3. Wait while Edison drives forward and scans the barcode.
4. Choose a button on your remote control that you want to match to that remote code. Point the remote at your robot and press your selected button. Be sure and use different remote-control buttons for each different remote code.



Hint!

You need Edison to be able to drive in order to scan a barcode, so be sure to scan whichever barcodes you need before you start building your robotic creation!

Once the robot is paired to the remote, you can build the EdCrane and write a program in EdScratch that tells Edison what to do if it detects each of the four different remote codes.



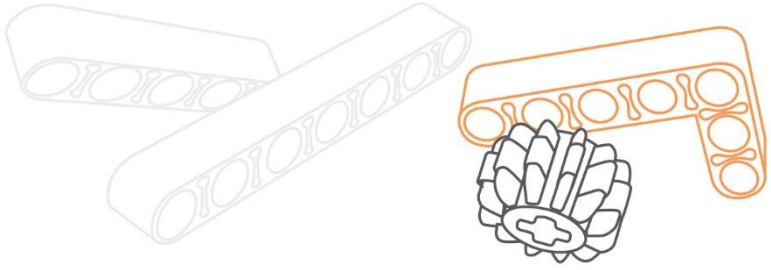
Use this link

Go to meetedison.com/content/EdCreate/EdBuild-EdCrane-instructions.pdf

This link will take you to the step-by-step instructions for building the EdCrane.

Using your plan as a reference, write your EdScratch program so that the EdCrane will react to each remote code with the outputs it needs to complete the desired action.

Once you have built and programmed your creation, try it out!



Name_____

U4-2.5c Challenge up: Firefighting water cannon

Fighting fires, including major disasters like wildfires, is a dangerous but necessary task. When terrible fires break out, the conditions can become extremely dangerous for humans. One way to help fight these fires is by using firefighting robots.

For this challenge, you will need to build and program the EdTank to use its cannon to shoot a jet of water (represented by the rubber band) to help fight a fire.

What to do

To complete this project, you will need to build the EdTank, create your test space, and program your EdTank using EdScratch.

Test-space set-up

You need to create the test-space you will use to run your firefighting water cannon program. Create two 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.

One black line represents the firefighting team's base, and the other represents the 'ash line' beyond which the fire is burning.



Hint!

The firefighting EdTank robots will fire their rubber bands out from the 'ash line'. Make sure that this line is facing somewhere where no one will get hit, such as a wall.

The EdScratch program

You will need to write two different programs, one for the top Edison and one for the bottom Edison in the EdTank.

Your programs need to control the EdTank so that the robot will drive from the firefighting team's base until it encounters the 'ash line' (but no further), shoot out its water payload (in other words, fire the cannon) and then return to the firefighting team's base.



Hint!

- You can use messaging in your programs to have the two Edison robots communicate with each other.
- Pseudocode helps us plan, comments help us debug. Use them to help you plan and test your program!

Name_____

The build

You will need to build the EdTank to use as the firefighting water cannon.

Use this link

Go to meetedison.com/content/EdCreate/EdBuild-EdTank-instructions.pdf

This link will take you to the step-by-step instructions for building the EdTank.

Remember that each of the two robots will need to be programmed with one of your EdScratch programs. Be sure to get the right program into the right robot!

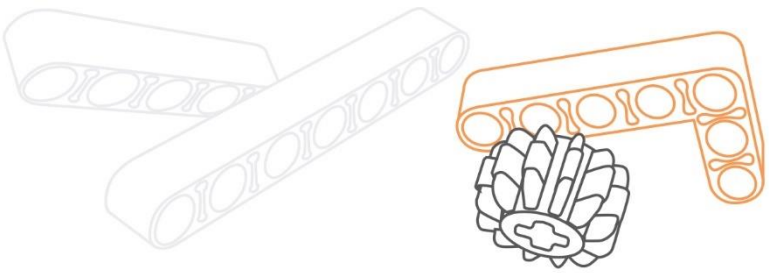
Try it out!

Test out your solution on the test space.

Did your firefighting robot drive up to the 'ash line', fire the water cannon and then come back to the base to be loaded again?

Hint!

If your firefighting robot isn't working quite right, don't give up! Keep experimenting, testing and problem solving until you get your firefighting robot working like a hero!



Name_____

U4-2.5d Challenge up: Semi-automated digger

Remote-controlled construction machines allow people to handle potentially hazardous tasks or environments more safely. In construction projects which require demolition, it is especially important to keep the machine operators away from flying debris. Many tasks on a construction site need to be performed again and again. Pre-setting, or automating, a portion of this type of task limits the amount of active operation required by the controller. Automating helps save time and limits the chances of human errors occurring, all while keeping the operators safer.

For this challenge, you will need to build and program the EdDigger to perform a rubble clean-up task semi-autonomously.

What to do

To complete this project, you will need to build the EdDigger, create your test space, and program your EdDigger using EdScratch.

Test-space set-up

You need to create the test-space you will use to run your rubble clean-up program. Create a 'worksite' with a pile of 'rubble' in one area and a drop-off zone in a different area where the rubble needs to be delivered so it can be hauled away.



Hint!

You can use any small objects to represent the rubble, even extra bits of EdCreate! Just make sure the EdDigger will be able to scope up the objects you choose.

The EdScratch program

You will need to write two different programs, one for the top Edison and one for the bottom Edison in the EdDigger. The robot should only start moving when the remote-control operator presses the right button on a remote control.

Once the remote code is detected, the robot should be able to perform the rubble clean-up task autonomously. Your programs need to control the EdDigger so that the robot will drive from the drop-off zone to the rubble pile, scoop up some rubble, then return to the drop-off zone and deliver the rubble.



Hint!

- You will need to use one programmable TV remote code from activity sheet U4-6.
- You can use messaging in your programs to have the two Edison robots communicate with each other.
- Pseudocode helps us plan, comments help us debug. Use them to help you plan and test your program!

Name_____

The build

You will need to build the EdDigger to use in your rubble clean-up task.

Use this link

Go to meetedison.com/content/EdCreate/EdBuild-EdDigger-instructions.pdf

This link will take you to the step-by-step instructions for building the EdDigger.

Remember that each of the two robots will need to be programmed with one of your EdScratch programs. Be sure to get the right program into the right robot!

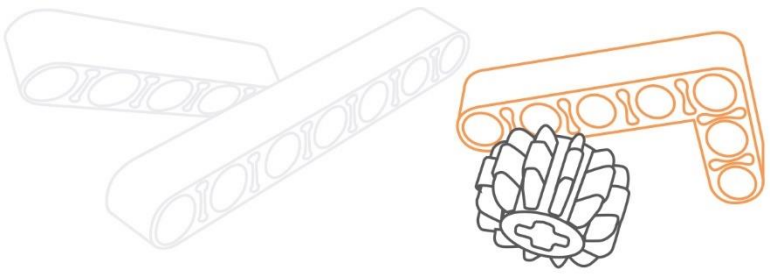
Try it out!

Test out your solution on the test space.

Did your robot wait for a remote code, then drive up to the rubble pile, scoop up some rubble, then bring it back to the drop-off zone and deliver it?

Hint!

If your rubble clean-up digger isn't working quite right, don't give up! Keep experimenting, testing and problem solving until you get your robot working like the #1 employee!



Name_____

U4-2.5e Challenge up: Hazardous material removal

Many types of materials can be hazardous to people, such as biomedical waste, by-products from manufacturing or radioactive material. To keep people safe, these types of materials need to be safely moved and disposed of away from populated areas. Remotely operated vehicles with robotic arms are one way to handle this task.

For this challenge, you will need to build and program the EdRoboClaw to use its articulated arm to carry away some hazardous material, placing it into a pre-designated area.

What to do

To complete this project, you will need to build the EdRoboClaw, create your test space, and program your EdRoboClaw using EdScratch.

Test-space set-up

You need to create the test-space you will use to run your hazardous waste removal program. Create a 'disposal zone' by marking out an area, such as a square, with black lines on a white background surface. The inside of the 'disposal zone' is where the hazardous material will need to be dropped off. You will also need something to represent the hazardous waste.



Hint!

Make sure the robot's arm can pick up and carry whatever is representing the hazardous waste.

The EdScratch program

You will need to write two different programs, one for the top Edison and one for the bottom Edison in the EdRoboClaw.

Your programs need to control the EdRoboClaw so that the robot will pick up the hazardous waste and carry it to the disposal zone where it will drop off the waste. Your robot should not enter the zone, but its arm can reach into the disposal area. Once the material is dropped off, the robot should retreat away from the zone.



Hint!

- You may want to start the robot with the claw already hovering open over the 'waste'.
- You can use messaging in your programs to have the two Edison robots communicate with each other.
- Pseudocode helps us plan, comments help us debug. Use them to help you plan and test your program!

Name_____

The build

You will need to build the EdRoboClaw to use in your hazardous waste removal task.

Use this link

Go to meetedison.com/content/EdCreate/EdBuild-EdRoboClaw-instructions.pdf

This link will take you to the step-by-step instructions for building the EdRoboClaw.

Remember that each of the two robots will need to be programmed with one of your EdScratch programs. Be sure to get the right program into the right robot!

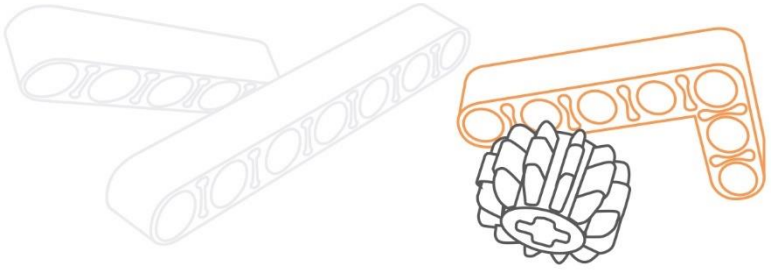
Try it out!

Test out your solution on the test space.

Did your robot pick up the hazardous waste, carry it to the disposal zone, drop the waste off inside the designated area, then retreat away from the disposal zone?

Hint!

If your robot isn't working quite right, don't give up! Keep experimenting, testing and problem solving until you get your robot to remove the hazardous waste and come back to safety!



Name_____

U4-2.5f Challenge up: Homing pigeons

Homing pigeons are a type of domestic pigeon descended from the rock pigeon. In the wild, rock pigeons have an incredible innate ability: they can find their way home even from very long distances. People have bred homing pigeons for this special talent and use the birds to fly back to their homes, carrying messages along with them!

Can you get your Edison robot to act like a homing pigeon?

What to do

To complete this project, you will need to design a way to get Edison to act as much like a homing pigeon as you can, returning to a designated 'home base' from somewhere else. You will need to create a test space to use and a program to run in your Edison robot.

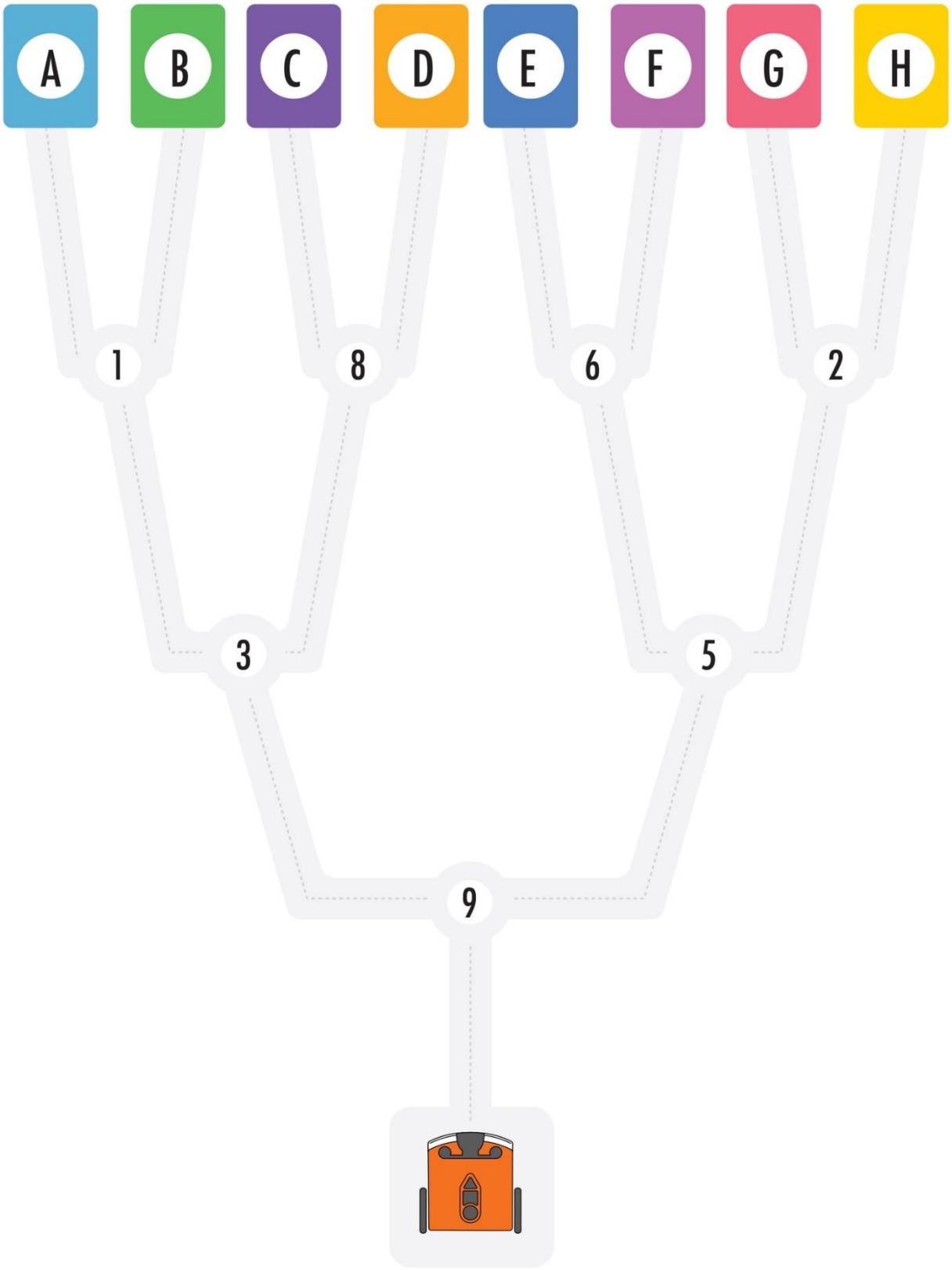
How you design your test space and how your program works is up to you. You can use any of Edison's sensors in any way you like, including line tracking, obstacle detection, and IR messaging.



Hint!

- Sensors like the obstacle detection beam and line tracker need to be turned on to work. The program also needs to tell Edison what event the sensors should look for and how to react to that event.
- Some of Edison's sensors, including obstacle detection, generate and store data when they detect specific events. It is good coding practice to clear the sensor data, especially when you use sensor events in conditionals which are nested inside loops. You don't want the data from a previous loop to affect the next loop! It's also wise to clear the sensor data at the start of a program, just in case the robot had old data stored from a previous program.
- Some of Edison's sensors, like IR messaging and obstacle detection, cannot be used at the same time. However, you can turn sensors on and off again in different spots in your program if you want.
- Use your test space design to your advantage. What could you build that would help Edison find and get back home?
- How could you signal the robots that it is time to start finding their way back home?
- Pseudocode helps us plan, comments help us debug. Use them to help you plan and test your program!

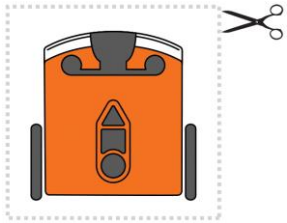
Activity sheet U4-1: If-else maze



Activity sheet U4-2: Pseudocode step-by-step

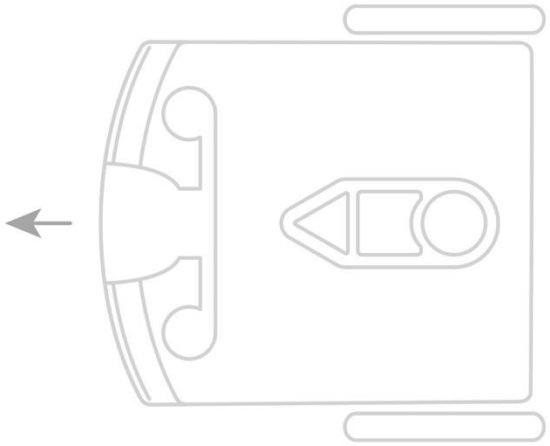
				J		D
	6		I			
F		C	7		9	
		H		4	G	2
3		1				E
B			A	8	5	



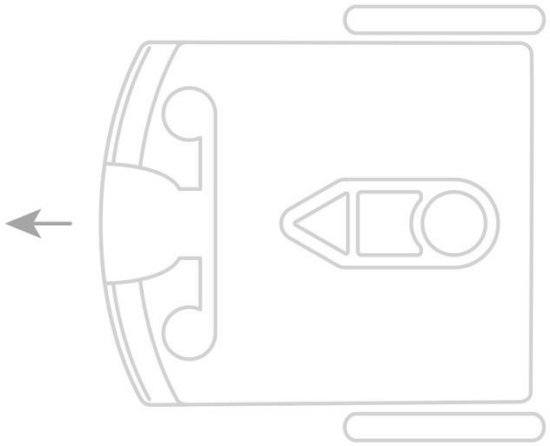


Activity sheet U4-3: Line tracker test zone

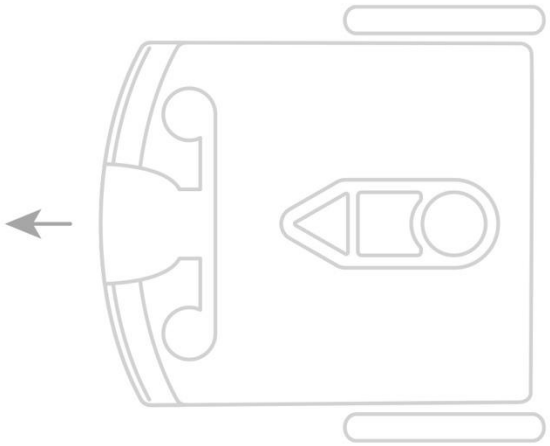




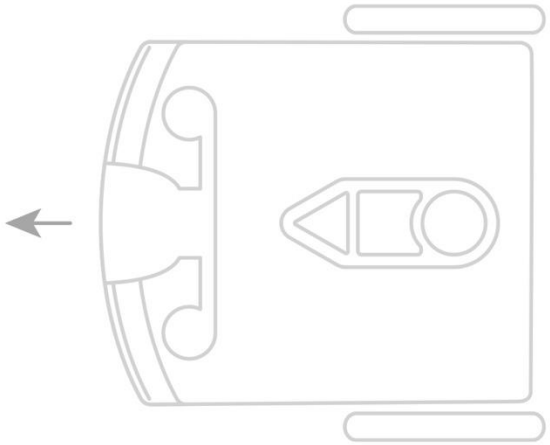




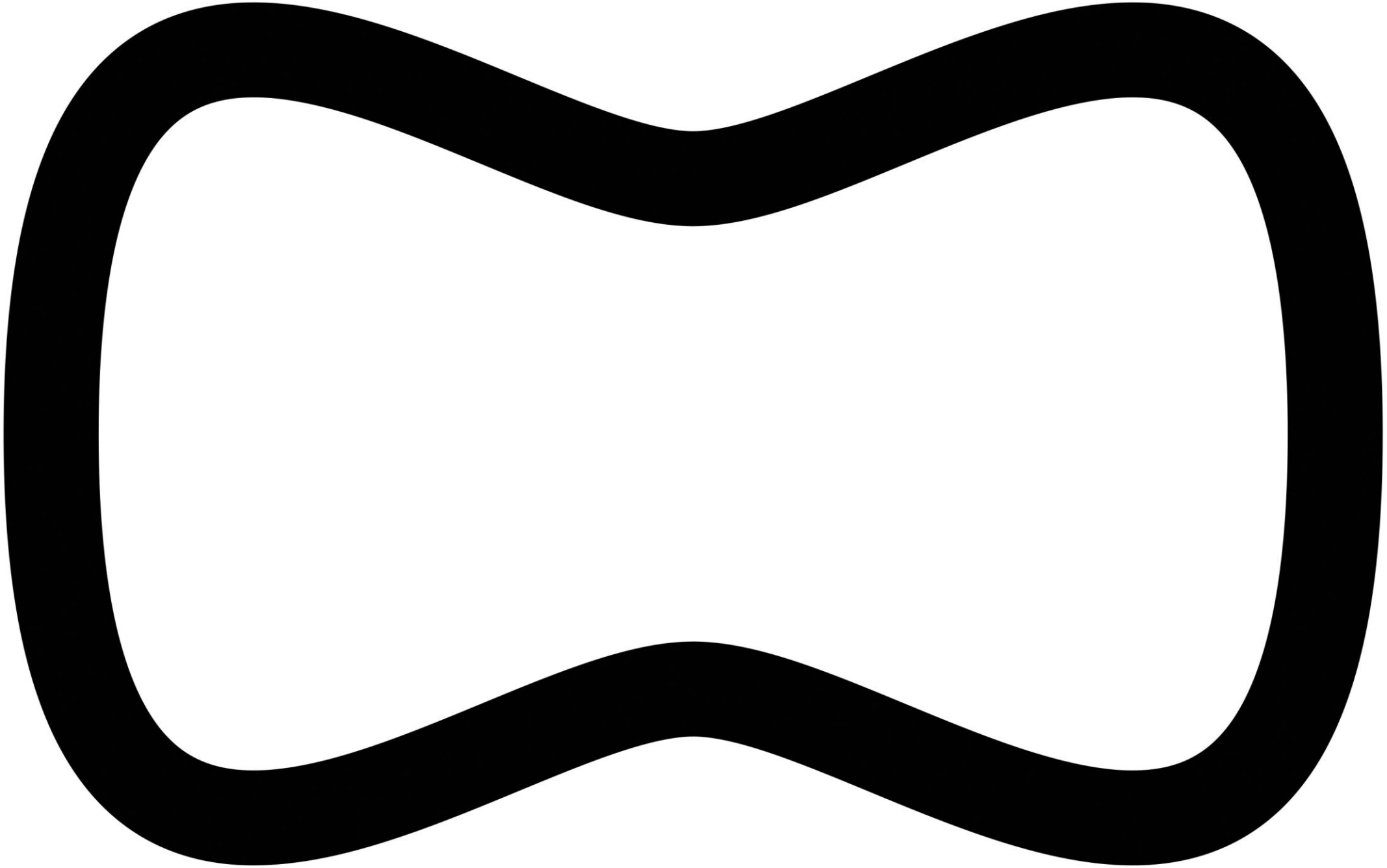




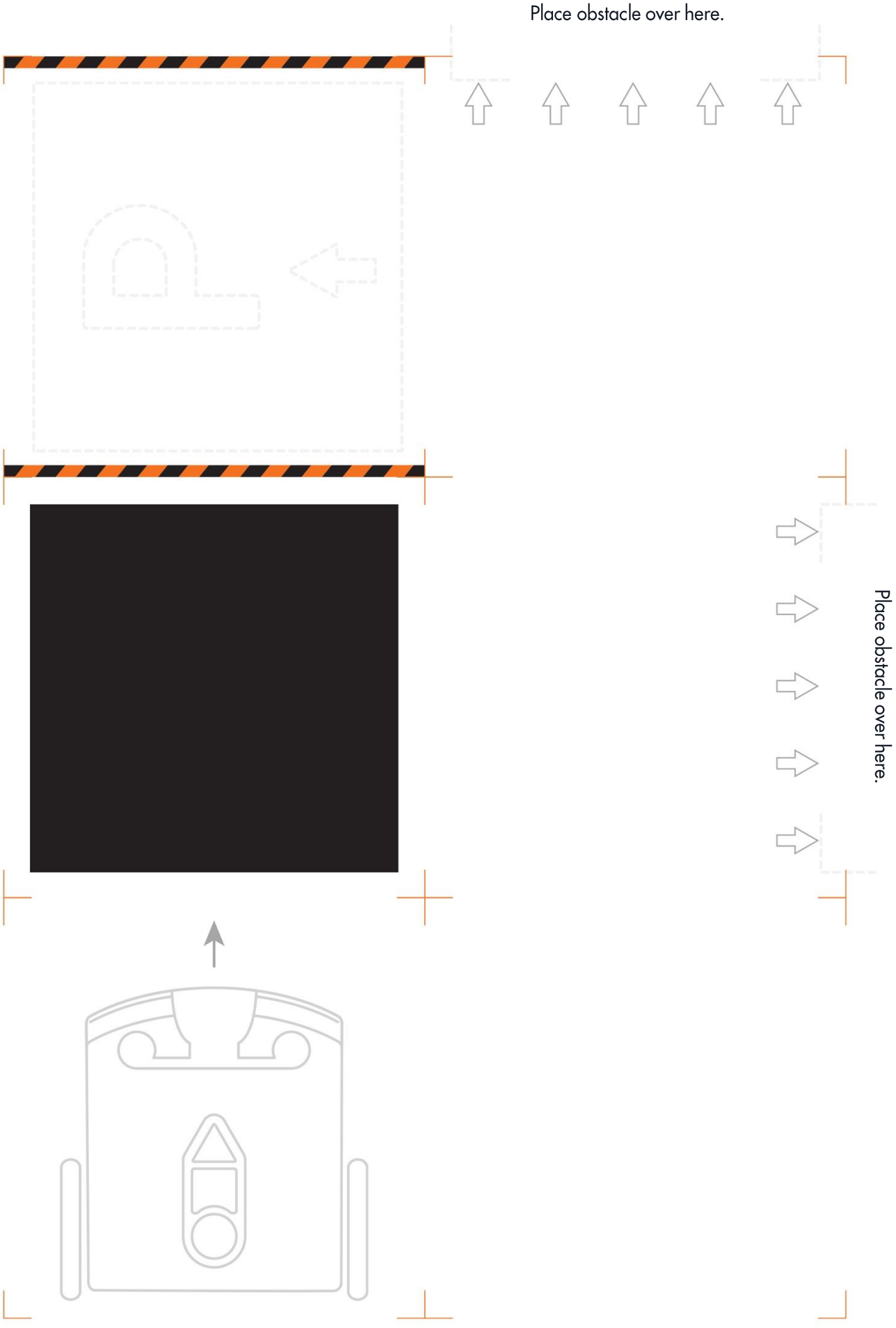




Activity sheet U4-4: Non-reflective border



Activity sheet U4-5: If line, go right



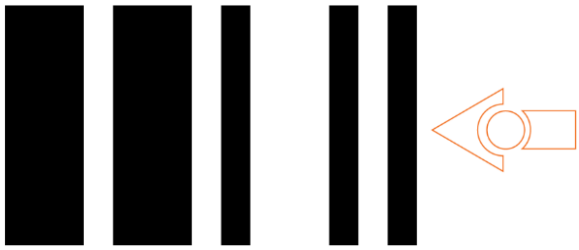
Activity sheet U4-6: Programmable TV remote codes



TV/DVD remote control code # 0



TV/DVD remote control code # 1



TV/DVD remote control code # 2



TV/DVD remote control code # 3



TV/DVD remote control code # 4



TV/DVD remote control code # 5

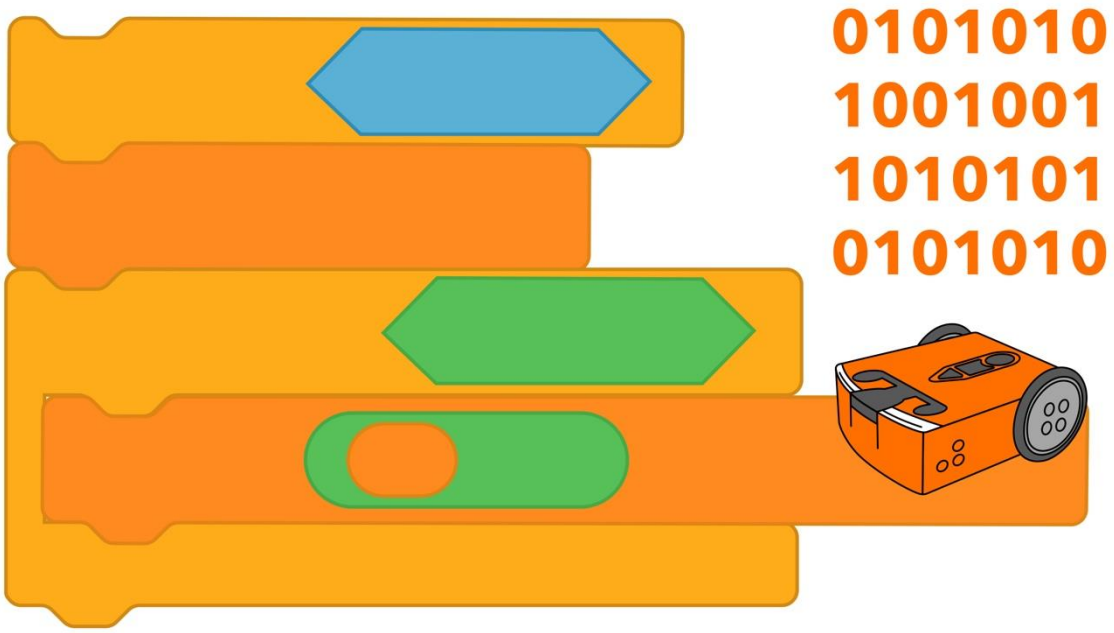


TV/DVD remote control code # 6



TV/DVD remote control code # 7

Unit 5: Versatile variables



Edison

Name_____

U5-1.1 Let's explore 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.



Don't forget

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

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.

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.

Name_____

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**.



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.

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.

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)?

1. (A*2) = B

   Meaning: _____

   Resolves to: _____

2. A >= B

   Meaning: _____

   Resolves to: _____

3. (A+A) ≠ B

   Meaning: _____

   Resolves to: _____

4. (A-1) < (B-3)

   Meaning: _____

   Resolves to: _____

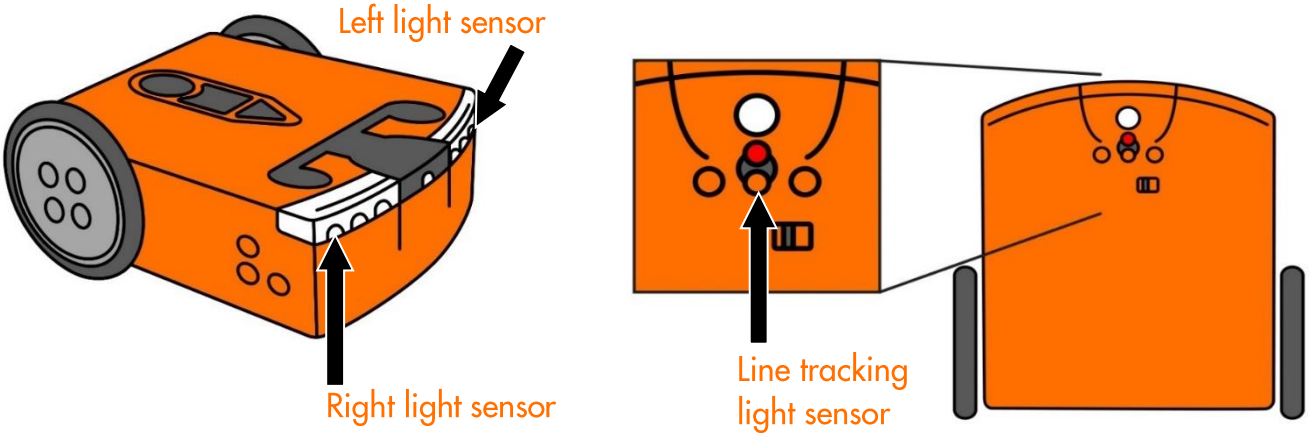
Name_____

U5-1.2 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.

Task 1: 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.

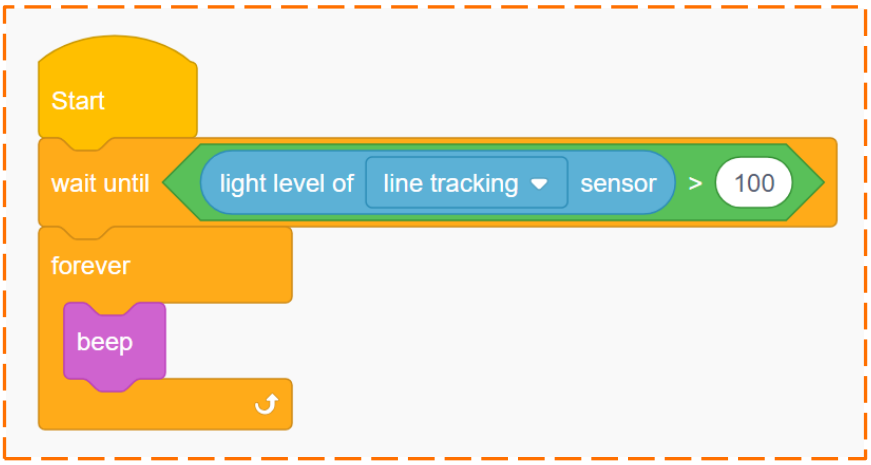
We can use the light reading from one of the light sensors as a value in an EdScratch program.

Name_____

Task 1: 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_____

Task 2: 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?

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

Name_____

U5-1.2a 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

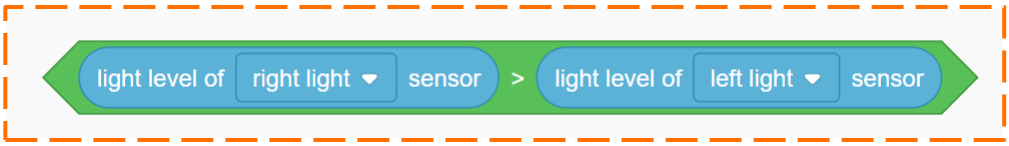
Let's program Edison to behave like a moth, following a bright light. Your program should get Edison to move towards the brightest light it detects. To write this program, you will need to use the light level readings from Edison's top two light sensors.



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.

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 torch, 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 torch. 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!

Name_____

U5-1.2b 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.

1. Write down your pseudocode.



Hint!

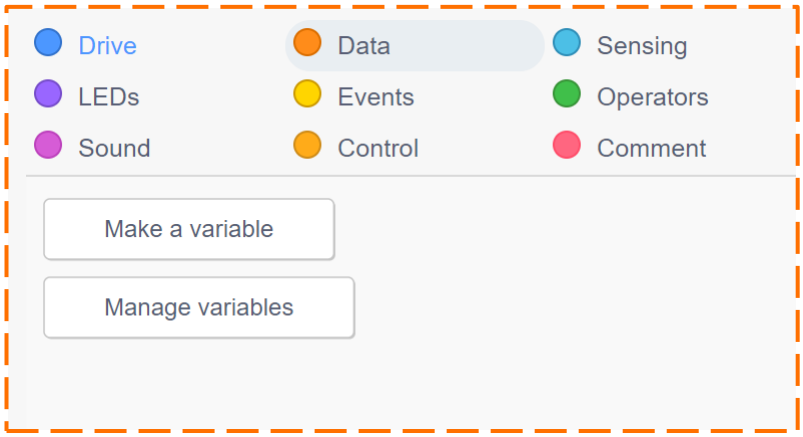
You can compare the light level reading from one of Edison's light sensors to a fixed number. You can also compare the light level readings from one light sensor to the light level reading of a different light sensor. Which approach will work best for what you are trying to do in this program?

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 torch, and test to see if Edison acts like a cockroach, avoiding the light.

Name_____

U5-1.3 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.

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.



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.

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.



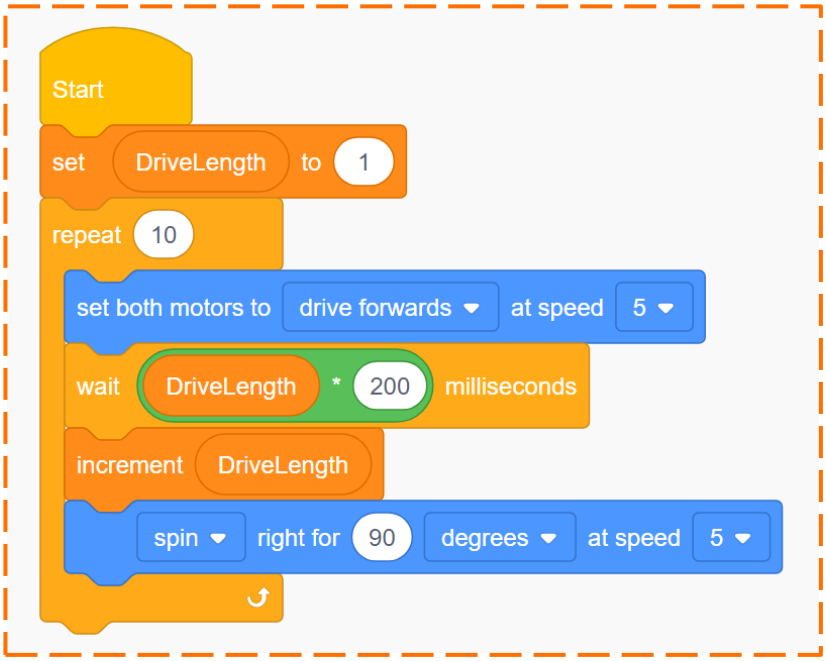
Jargon buster

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

Task 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.

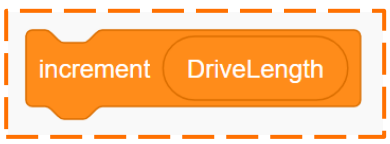
Look at the following program:



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.

Name_____

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



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.

Remember, 1 second = 1000 milliseconds.



Don't forget

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

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.

1. 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			

Task 2: 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.

Name_____

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

_____

_____

3. 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?

_____

_____

_____

_____

Name_____

U5-1.3a 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.



Hint!

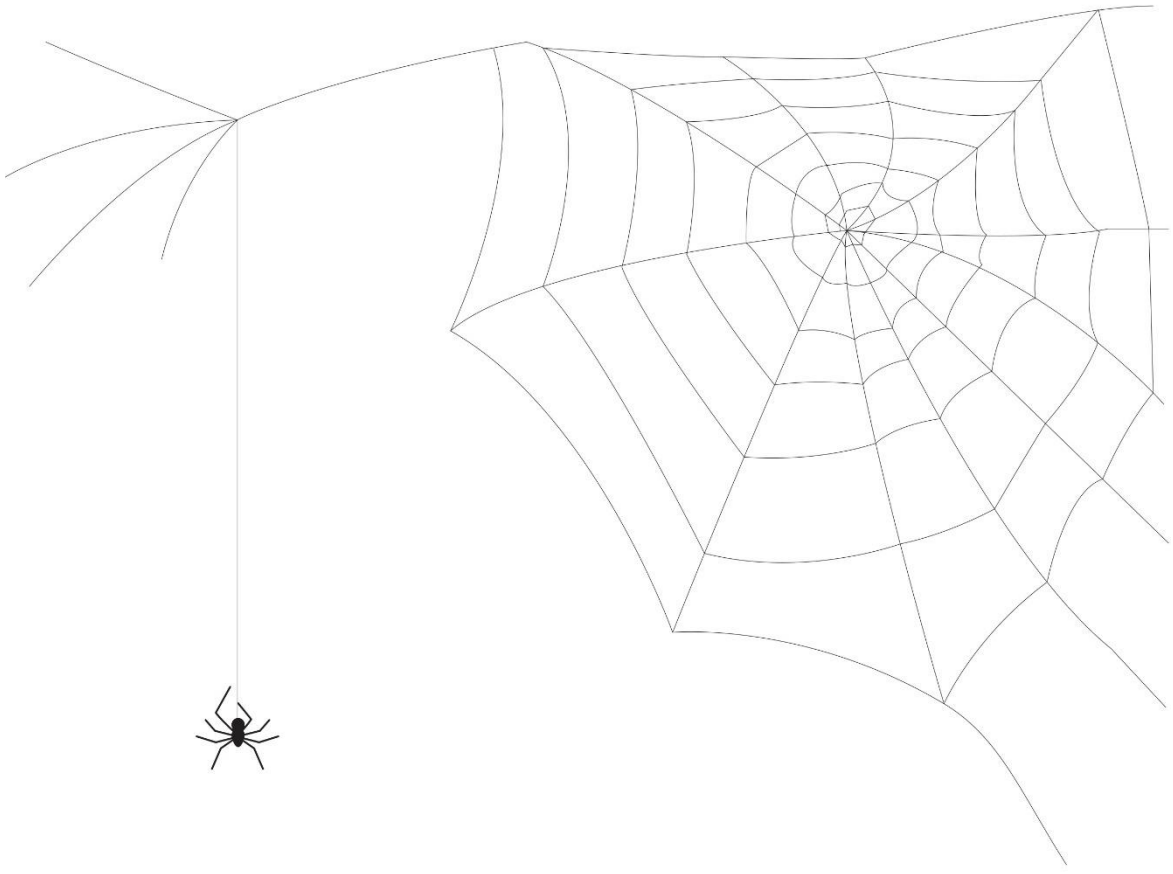
You might want to look at the program in activity U5-1.3 for some inspiration to help you write your program.

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.

Mini challenge!

Can you make Edison leave a 'sticky spider thread' trail behind the robot as it drives? Use something, like some string, to represent a spider's silk. See if you can engineer a solution so that Edison leaves a trail behind it as it drives.

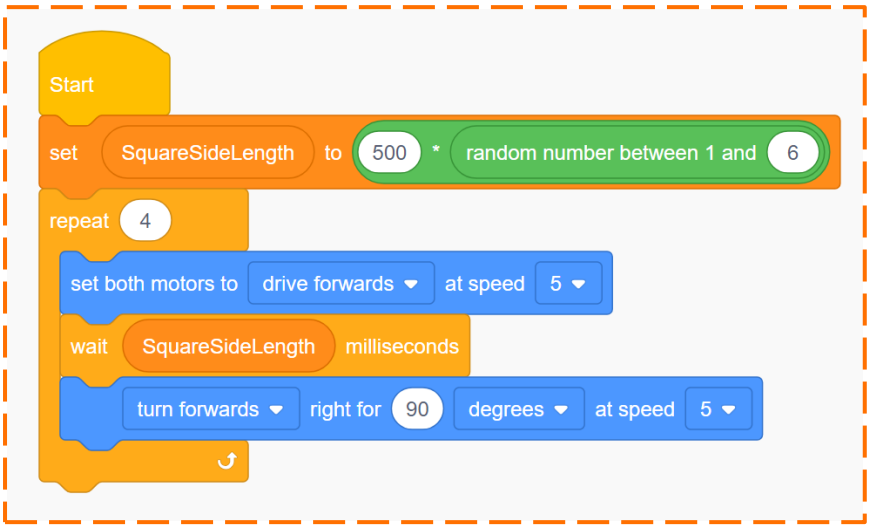


Name_____

U5-1.3b 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.

1. 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_____

U5-1.4 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.



Don't forget

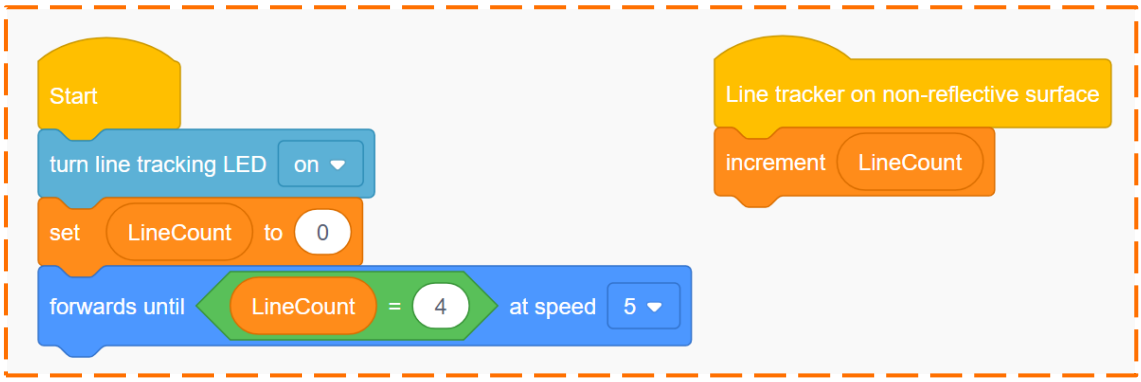
A variable is a bit of memory that is used to store some bit of information in a program.

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

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:



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

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

Name_____

Try it out

Write the program in EdScratch, then download it to your Edison robot. Test the program using activity sheet U5-1 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.

3. 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.

4. 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.

Name_____

U5-1.4a 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.

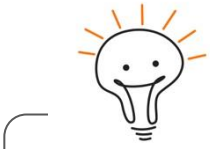
We can program Edison to work like a 400-meter specialist, changing speed as the robot progresses through a course.

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.

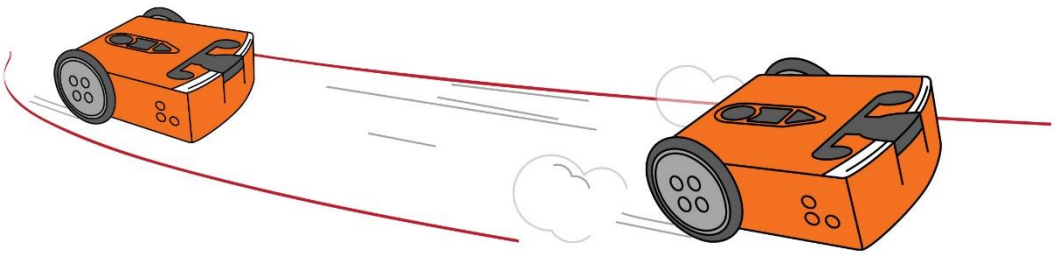
Test your program using activity sheet U5-1. For this activity, the fourth line is the finish line, so you need to be sure the robot crosses this line, rather than just stopping once it detects that line.

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 program in activity U5-1.4 for some inspiration to help you write your program.



Name_____

U5-1.4b Change it up: Edison-controlled flag machine

Edison's infrared (IR) light sensor is the sensor that lets Edison emit and detect infrared light. We can use the IR sensor to send messages to or receive messages from other Edison robots. One robot can send out an infrared message using its two IR LEDs which another robot's IR receiver can detect.



Don't forget

Infrared is sometimes abbreviated as IR. In EdScratch, IR message blocks are blocks that relate to Edison's infrared messaging using the infrared LEDs and infrared receiver.

To send an infrared message with your Edison robot in EdScratch, you need to use the send IR message block, which has an input parameter that allows you to send a specific message. You can change the message by changing the value of the input parameter in the send IR message block. The send IR message block has an input range of 0 to 255. In other words, Edison can send and receive 256 different 'messages'.

By using IR messages with different values, we can create a program telling Edison to react one way if the robot detects one IR message, and a different way if it detects another message. For this to work, we need to use a variable to store the data Edison receives with its IR sensor.

Let's try building and programming a flag machine using Edison's motor outputs which you can control using messaging from another Edison robot.



Don't forget

A variable is a bit of memory that is used to store some bit of information in a program. Because all of Edison's sensors send back data to Edison as values, you can store that value in a variable.

What to do

The goal of this activity is to build and program a flag machine that you can use to help make studying for a test more fun. You will need to work with a partner for this activity and use two Edison robots. One robot will be the flag machine which will have a two-sided flag with the word 'Yes' on one side of the flag and the word 'No' on the other side. This 'flag machine' robot will receive messages. The other robot will be the controller bot that will send out the messages.

There are three parts to this project: designing and building the flag machine, programming the flag machine, and programming the controller bot. When your flag machine is finished, and both robots are programmed, you will be able to work with your partner taking turns quizzing each other. You can use your controller bot to move the flag machine, signalling if your partner answered your question correctly or not!

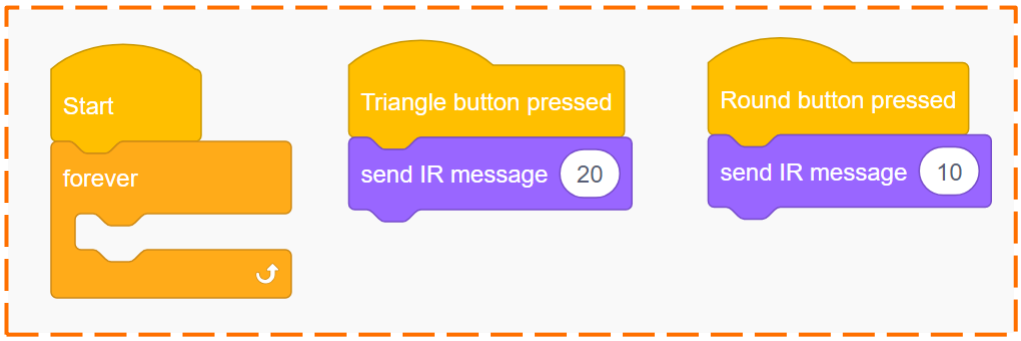
The first thing to do is program your controller bot.

Name_____

The controller bot program

The Edison robot that sends out the IR messages is the controller bot. This robot needs to be able to send out two different messages. One message will tell the flag machine to show the 'Yes' side of the flag. The other message will tell the flag machine to show the 'No' side.

Look at this program:



You can use this as your controller bot program. This program tells the Edison acting as the controller bot to send out an IR message whenever you press the round or triangle button on the controller bot. If you press the triangle button, the controller bot will send out IR message 20. If you press the round button, however, the controller bot will send out IR message 10 instead.

In your controller bot program, you can use whatever values you want between 0 and 255 for your two IR messages. The two messages need to have different values, however, so the receiving robot will be able to tell the messages apart.

Whatever values you send with your controller bot, you also need to use in your flag machine program.

1. Planning out how your two Edison robots will work together is the only way your creation will be able to function. Work with your partner to decide which button on the controller bot will do what. Write down your plan.

Round button:

- When you press the **round** button on the controller bot, it will send IR message _____ and that will tell the flag machine to show the _____ side of the flag.

Triangle button:

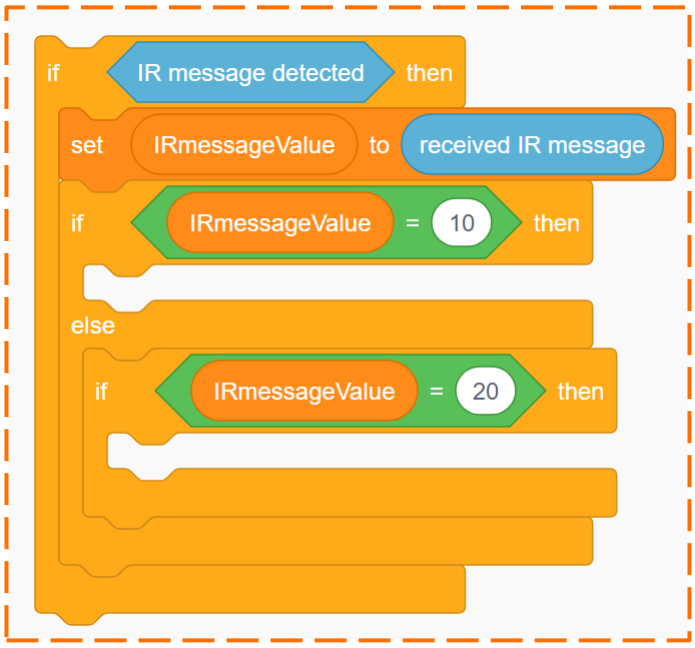
- When you press the **triangle** button on the controller bot, it will send IR message _____ and that will tell the flag machine to show the _____ side of the flag.

You will need to use this plan when you design your flag machine program.

The flag machine and flag machine program

Design your flag machine to use Edison's outputs to move the two-sided flag when it receives the different IR messages. The flag machine should show the 'Yes' side of the flag when it receives one IR message and the 'No' side when it receives the other IR message. Use the plan you created with your controller bot to help you.

Look at the following code:



This isn't a full program, but you can use this code to help you build your flag machine program.

This section of code uses a variable called IRmessageValue. The value of this variable is set to be whatever the value of the received IR message is. When you work with sensor values, you should always store any values you want to use in your program in a variable. You can then have your program use that variable in whatever ways you need anywhere in the program.



Why is that?

Computers don't keep track of values from sensors unless you tell them to. When you tell a computer to store a value in a variable, you are telling the computer to remember that value.

Remember that Edison's IR detector is always on, checking for data over and over. The robot does not keep track of every value it detects forever. It only keeps the value in its working memory until it is called for in a program or replaced by a new reading. By storing the value of the IR message into a variable, you are telling Edison to hold on to that value, so you can do something with it.

The value will stay in the variable until the sensor detects a new IR message and the set block replaces the old value with the new one.

Program the flag machine using a variable to store any IR messages the robot detects. Once you have built and programmed your creation, try it out!

Name_____

U5-1.4c Change it up: Hey Edison, where do I go?

Edison's infrared (IR) light sensor is the sensor that lets Edison emit and detect infrared light. We can use the IR sensor to send messages to or receive messages from other Edison robots. One robot can send out an infrared message using its two IR LEDs which another robot's IR receiver can detect.



Don't forget

Infrared is sometimes abbreviated as IR. In EdScratch, IR message blocks are blocks that relate to Edison's infrared messaging using the infrared LEDs and infrared receiver.

To send an infrared message with your Edison robot in EdScratch, you need to use the send IR message block, which has an input parameter that allows you to send a specific message. You can change the message by changing the value of the input parameter in the send IR message block. The send IR message block has an input range of 0 to 255. In other words, Edison can send and receive 256 different 'messages'.

By using IR messages with different values, we can create a program telling Edison to react in different ways depending on what IR message it detects. For this to work, we need to use a variable to store the data Edison receives with its IR sensor.

Let's try using two Edison robots, one driving and one navigating, to get through a treasure maze. The robots will need to use different IR messages to communicate in order to make it to the correct location.



Don't forget

A variable is a bit of memory that is used to store some bit of information in a program. Because all of Edison's sensors send back data to Edison as values, you can store that value in a variable.

What to do

You will need two Edison robots for this activity: one robot will be the navigator bot, and the other will be the driver bot. The driver bot will move through the treasure maze according to the messages it receives from the navigator bot.

There are three parts to this project: making the treasure maze, programming the driver bot, and programming the navigator bot.

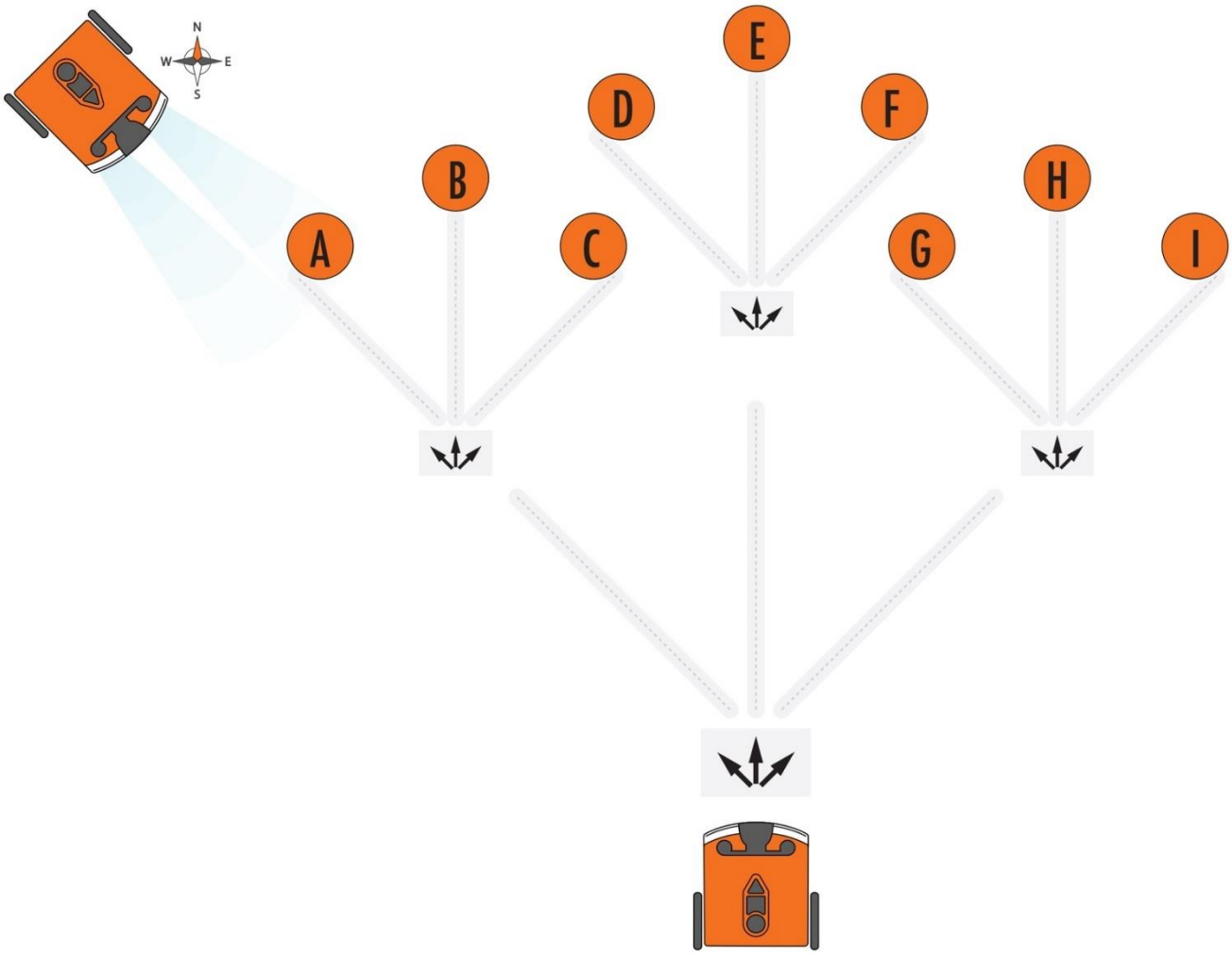
The first thing you need to do is make your treasure map maze which you will use as a test space.

Name_____

The treasure map maze

Make a branching treasure map which has many different possible end locations. Your driver bot will need to navigate this maze to get to the spot that holds the treasure. Your map needs to be big enough for Edison to drive, but not so big that the two robots cannot message back and forth to each other. You will also need a spot for the navigator bot to sit to send out commands.

The basic shape of your treasure map test space needs to look something like this:



All the paths in your map should be the same length as each other. Choose how long the paths will be, how many options come out of each junction, and how many junctions there are before the final treasure spots. Work together to design and decorate your treasure map test space.



Hint!

Use something like letters or numbers to mark the different possible treasure spots. You don't know where the real treasure will be yet!

Name_____

The navigator bot program

The Edison robot that is not going to move, but send out instructions to the driver bot is the navigator bot. You won't be able to write the navigator bot's final program until you know where on the treasure map you are sending the driver bot.

The navigator bot will send out 'commands' to the driver bot using a series of different IR messages. Each different message will tell the driver bot to do a different action. How many different messages your navigator bot needs to be able to send out will depend on your treasure map.

For example, if your map has three pathways coming out of each junction, you will need to be able to send three different IR messages: one to tell the driver bot to take the left path, one to tell the driver bot to take the middle path, and one to tell the driver bot to take the right path.

You will also need one more IR message. This message will be sent by the driver bot to the navigator bot when the driver bot is ready for the next command. This message will be the driver bot's way of saying, "I'm at the next junction. Now, where should I go?"

Each of the IR messages you send needs to have a different value so that the receiving robot will be able to tell them apart. You can use whatever values you want between 0 and 255 for your IR messages. Be sure to match up your messages across both your navigator bot and driver bot programs.

1. How many messages is your navigator bot going to need to be able to send to the driver bot? What value are you going to use for each message (including the driver bot's message to ask for the next command)? What will the driver bot need to do when it receives each separate message from the navigator bot? Work together and design your plan. Make notes to help guide you when you write your EdScratch programs.

_____

_____

_____

_____

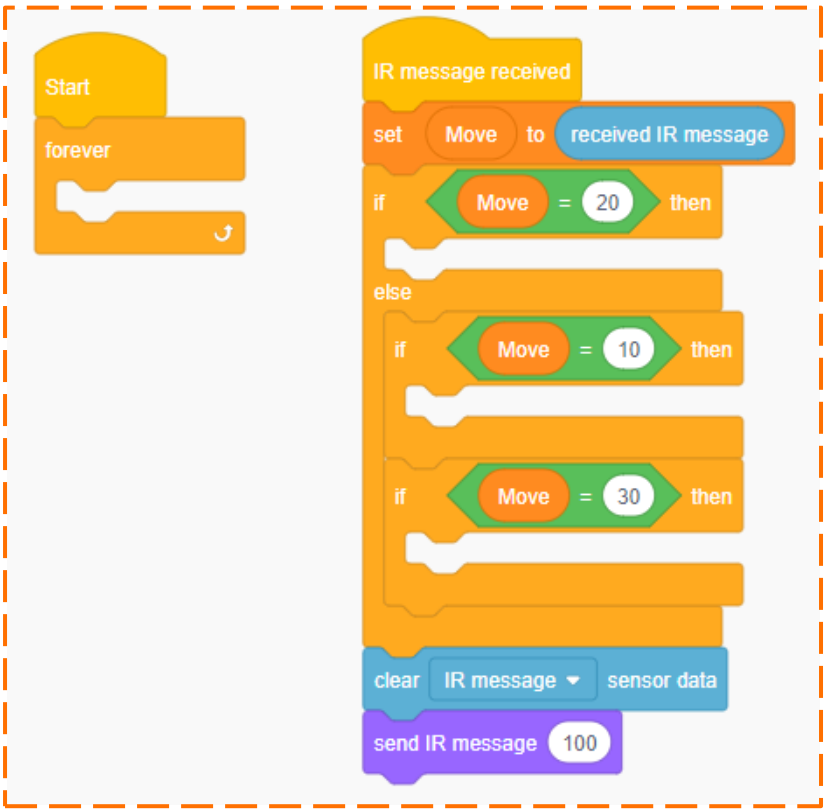
IR message value	Sending Edison (navigator or driver)	Receiving Edison will…

Name_____

The driver bot program

The Edison robot that is going to receive IR messages and drive according to those commands is the driver bot.

Look at the following code:



This isn't a full program, but you can use this code to help you build your driver bot program. This section of code uses a variable called Move. The value of this variable is set to be whatever the value of the received IR message is. When you work with sensor values, you should always store any values you want to use in your program in a variable. You can then have your program use that variable in whatever ways you need anywhere in the program.



Why is that?

Computers don't keep track of values from sensors unless you tell them to. When you tell a computer to store a value in a variable, you are telling it to remember that value.

Remember that Edison's IR detector is always on, checking for data over and over. The robot does not keep track of every value it detects forever. It only keeps the value in its working memory until it is called for in a program or replaced by a new reading. By storing the value of the IR message into a variable, you are telling Edison to hold on to that value, so you can do something with it.

The value will stay in the variable until the sensor detects a new IR message and the set block replaces the old value with the new one.

Name_____

Your driver bot program needs to check for an IR message, then store the value of that message as a variable. Depending on the value of that variable, the driver bot should take a different action to move down the corresponding path on your treasure map. Once the driver bot has arrived at the next junction, the robot needs to signal the navigator bot that it is ready for the next command.

Use your plan to help write the driver bot program. Once the driver bot is programmed correctly, it should be able to follow IR messages from the navigator bot and get to any treasure map location, no matter which one it is.

Try it out!

Once you have your treasure map ready and your driver bot programmed, choose a spot on your map to be the treasure location. Program your navigator bot to send out the IR messages in the correct sequence so that your driver bot will arrive at the treasure. Make sure you have the navigator bot wait for the driver bot's signal in between each command:



Run your programs in your robots to test them out. Did you arrive at the right treasure? If not, try to work out what went wrong. Adjust your programs and try again. Keep testing until you get your programs working.

Once you get to the treasure, choose a new spot on your map to the be the new treasure location and test again.



Don't forget

Once the driver bot is programmed correctly, it should be able to follow IR messages from the navigator bot and get to any treasure map location, no matter which one it is. You should only need to change the sequence of commands in your navigator bot program to send the driver bot from the start to any treasure location.

Name_____

U5-1.4d Change it up: The Edison chorus

Edison's infrared (IR) light sensor is the sensor that lets Edison emit and detect infrared light. We can use the IR sensor to send messages to or receive messages from other Edison robots. One robot can send out an infrared message using its two IR LEDs which another robot's IR receiver can detect.



Don't forget

Infrared is sometimes abbreviated as IR. In EdScratch, IR message blocks are blocks that relate to Edison's infrared messaging using the infrared LEDs and infrared receiver.

To send an infrared message with your Edison robot in EdScratch, you need to use the send IR message block, which has an input parameter that allows you to send a specific message. You can change the message by changing the value of the input parameter in the send IR message block. The send IR message block has an input range of 0 to 255. In other words, Edison can send and receive 256 different 'messages'.

By using IR messages with different values, we can create a program telling Edison to react in different ways depending on what IR message it detects. We can also get different Edison robots to only react to specific IR messages.

For this to work, we need to use a variable to store the data Edison receives with its IR sensor.

Let's try using IR messaging to help coordinate multiple Edison robots to play a song together in a round.



Don't forget

A variable is a bit of memory that is used to store some bit of information in a program. Because all of Edison's sensors send back data to Edison as values, you can store that value in a variable.

What to do

A round is a musical piece where two or more people (or robots!) sing or play the same melody, but each begins at a different time. While every participant is singing or playing a different part of the song, the melody still harmonises them together. In this activity, you will use multiple Edison robots to play a tune in a round.

You will need to work in a group for this activity with at least three Edison robots. One Edison will need to be the conductor bot, and all the other robots will be performer bots. The conductor bot will use IR messages to track the performer bots and tell each performer bot when to start playing music.

There are three parts to this project: choosing and arranging your song, planning your IR messages, and programming all of the robots. The first thing you need to do is choose a song for the robots to play.

Name_____

Choose and arrange your song

Together with your group, you need to choose which song you are going to perform. Many nursery rhyme songs work well in a round, such as *Row, Row, Row Your Boat*. You also need to decide at which points in the song you will have each new performer bot join in.

1. What song are your performer bots going to play and when will each new performer bot join in? Write down your plan to help you when you go to program each performer bot to play your song.

_____

_____

_____

_____

Optional: To program your song into EdScratch, you will need to write out each note. If you want, you can use this space to write down your music to make it easier to program into your performer bots.

Name_____

The IR message plan

One of your Edison robots will be your conductor bot. The conductor bot is not going to play the song, but instead manage all of the performer bots using IR messages. The conductor bot is going to need to both send and receive IR messages.

The conductor bot will need to send out messages to signal each performer bot to start playing one at a time. When a performer bot gets up to the place in the song where the next robot should join in, that performer bot needs to signal the conductor bot using a different IR message. The conductor bot will then know to get the next performer bot playing.

This will make a pattern in the IR messages that your conductor bot sends and receives. The conductor bot needs to send an IR message to start the first performer bot playing, wait for the signal from that performer bot, then send the next IR message to get the next performer playing. The conductor bot then needs to repeat this pattern of waiting for an IR message, checking which performer that message came from, then signalling the next performer over and over until all the performer bots have begun to play.

How many different messages your conductor bot needs to be able to send and receive will depend on how many performer bots are in your chorus. Each of the IR messages you send needs to have a different value so that the receiving robots will be able to tell them apart. You can use whatever values you want between 0 and 255 for your IR messages. Be sure to match up your messages across all of your robots' programs.

2. How many messages is your conductor bot going to need to be able to send and receive? What value are you going to use for each message? Which robot is each message for and what does the receiving robot need to do once it detects that IR message? Work together and design your plan. Make notes to help guide you when you write your EdScratch programs.

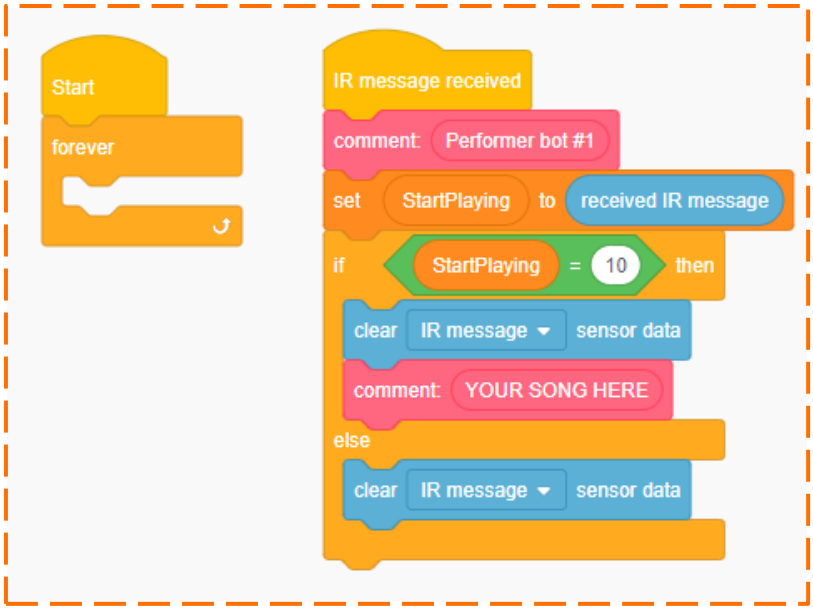
IR message value	Sending Edison (conductor or which performer)	Receiving Edison (conductor or which performer)	Receiving Edison's action

Name_____

Program all the robots

All of the Edison robots that are going to play the song are the performer bots. Each performer bot's program will tell that robot to wait to start playing until it receives an IR message from the conductor bot. Since we don't want all of the robots to start playing at the same time, each performer bot will need to start playing only if it detects its assigned IR message.

Look at the following code:



This isn't a full program, but you can use this code as an example to help you build your performer bots' programs. This section of code uses a variable called StartPlaying. The value of this variable is set to be whatever the value of the received IR message is. When you work with sensor values, you should always store any values you want to use in your program in a variable. You can then have your program use that variable in whatever ways you need anywhere in the program.



Why is that?

Computers don't keep track of values from sensors unless you tell them to. When you tell a computer to store a value in a variable, you are telling it to remember that value.

Remember that Edison's IR detector is always on, checking for data over and over. The robot does not keep track of every value it detects forever. It only keeps the value in its working memory until it is called for in a program or replaced by a new reading. By storing the value of the IR message into a variable, you are telling Edison to hold on to that value, so you can do something with it.

The value will stay in the variable until the sensor detects a new IR message and the set block replaces the old value with the new one.

Name_____

Each of your performer bots' programs needs to check for an IR message, then store the value of that message as a variable. Depending on the value of that variable, the performer bot should either start playing or just clear the data and go back to waiting for its IR message.

Use your plan to help write a program for each of your performer bots.



Hint!

Be sure each performer bot has the right send IR message block in the correct spot in the song!

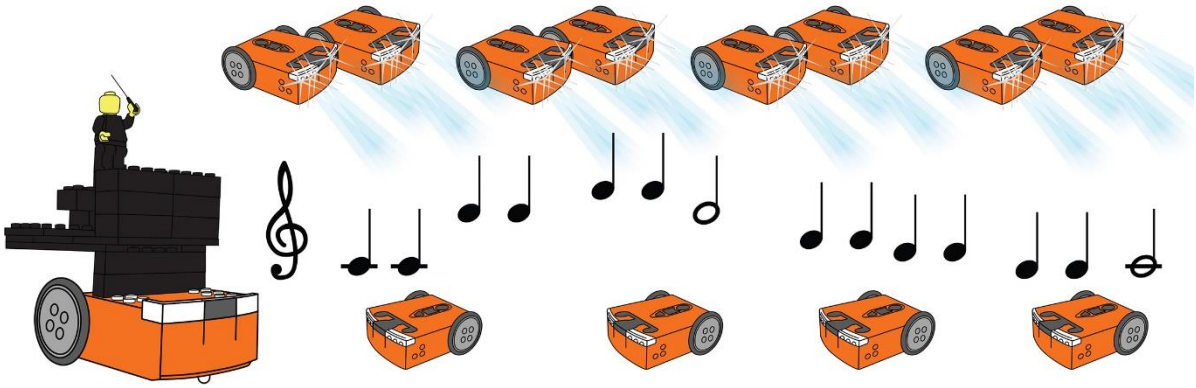
You also need to program the conductor bot using your plan as a guide. Remember, your conductor bot will also need to check for an IR message, then store the value of that message as a variable. Depending on the value of the variable, the conductor bot will need to send out a different IR message to get the next performer bot to start.



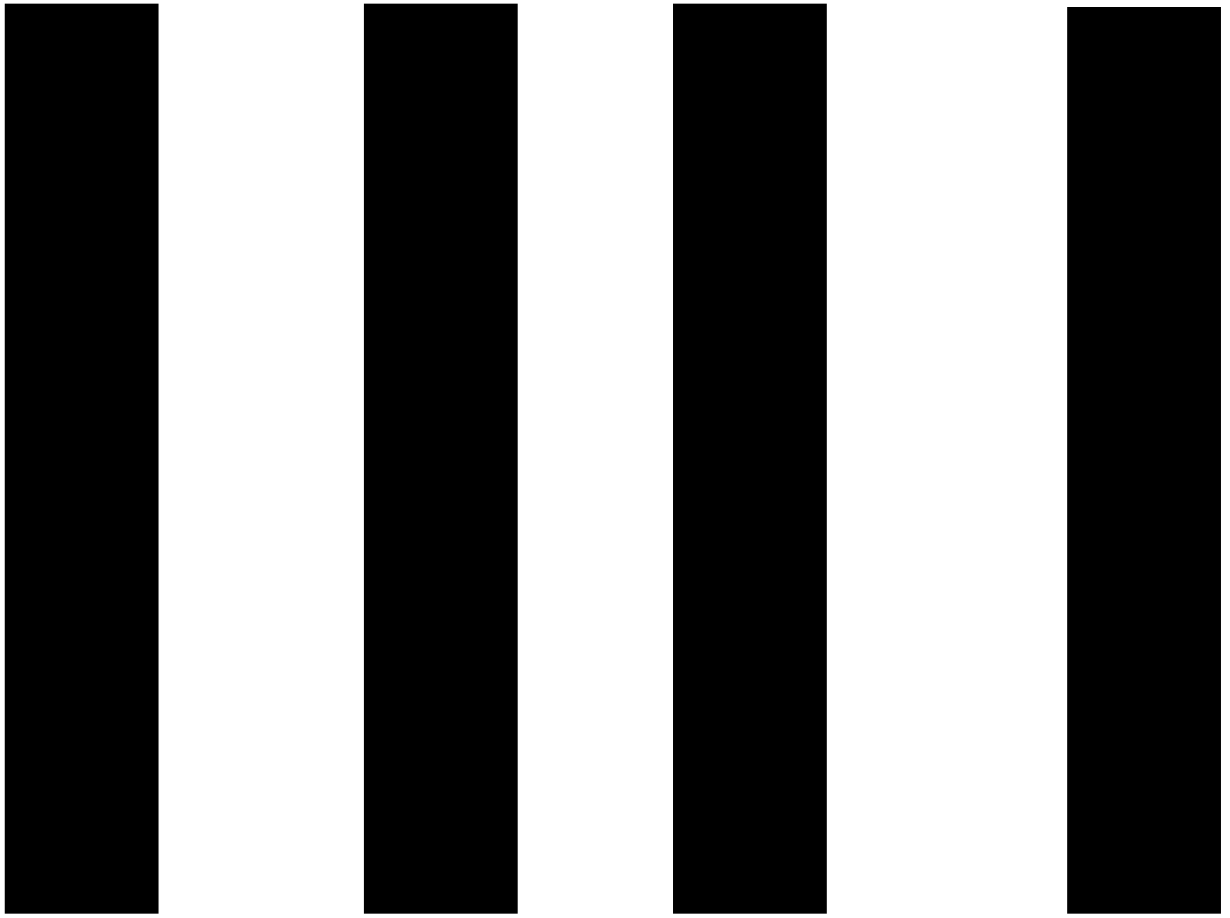
Hint!

Whenever you receive IR data and store that value as a variable, you should then clear the IR data using the clear IR message data block as the next action in your program.

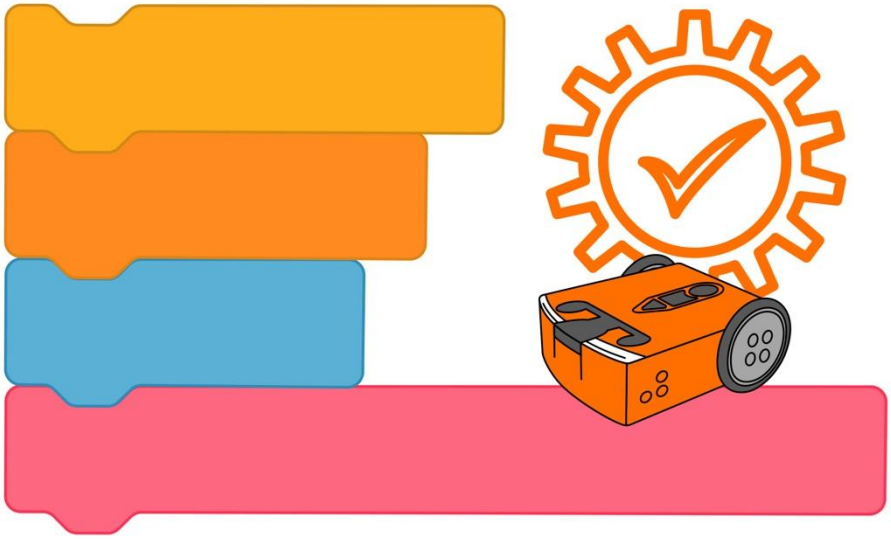
Once all your bots are programmed, test out your programs with your robots. Did each robot play when it was supposed to? If not, try to work out what went wrong. Adjust your programs and try again. Keep testing until you get your programs working.



Activity sheet U5-1: Four lines



Unit 6: Inventor's time!



Edison

Name_____

U6-1.1 Let's explore the design-build-test cycle

One of the best things about learning computer science is that it isn't just about writing programs. You can use computer science to create all sorts of things, including different inventions using your Edison robots.

By now you know how Edison robots work and different things you can get Edison to do by programming the robot in EdScratch. That means you have the skills to be an inventor using Edison and EdScratch to make and program all sorts of creations.

What could you create? How could you build it? What would the computer program you need to run your creation look like?

Inventing something with Edison and EdScratch is a big task, but you can make it a lot more manageable by **decomposing** it into smaller pieces and doing each thing one at a time.



Jargon buster

In computer science, **decomposition** is the process of breaking a complex problem or system into parts that are easier to understand, manage and solve. Breaking a large problem into smaller, more doable tasks makes it easier to determine exactly what needs to happen and in what order each task needs to be done.

Using decomposition to break any major job up into smaller parts is one of the best ways of approaching it. If you get stuck or feel overwhelmed when tackling a project, take a minute to think about how you can use decomposition to break it down into smaller tasks.

When it comes to creating your own inventions with Edison, you can also use **iterative testing** through something called the **design-build-test cycle** to help you.



Jargon buster

The **design-build-test cycle**, which is also sometimes called the design-build-test-learn cycle, is a process where you design something, build it and then test it out to see what works and what needs to be improved. You then take what you learn from the test and apply that back to the design. Because it is a cycle, you keep repeating each step, applying what you learn to make the next cycle better, until you are happy with the outcome.

Making and testing many versions of something, applying what you learn and making changes each time, is known as **iterating** or **iterative testing** and is a common practice when developing new things in computer science. Technology companies use the design-build-test cycle and iterative testing all the time when they are coming up with new products.

Name_____

It's pretty unlikely that your first design will work perfectly. That's okay! By using iterative testing and working through the design-build-test cycle, you can keep refining your idea, applying what you learn and making improvements.

Remember that a major part of computer science is using computational thinking and a major part of computational thinking is problem-solving!



Don't forget

Computational thinking means thinking about a problem or task in a similar way to how a computer thinks. It is a way of logically working through problems, decomposing them into smaller pieces, finding patterns, and then using the information to come up with a step-by-step solution.

In other words, computational thinking is a way of planning, problem-solving and analysing information the same way a computer does.

Another big part of computer science is being creative and trying new things. You never know what you might be able to create until you try!

Task 1: Brainstorm

Before you can get started designing, building and testing something, you need to come up with some ideas. Coming up with ideas is all about being imaginative, using your creativity and thinking about possibilities.

Brainstorming is one way to kick-start your thinking to generate ideas. When you brainstorm, you are trying to get thoughts to flow freely. All ideas are acceptable, no matter what they are. Don't make decisions about whether the ideas are possible or judge the ideas while you are brainstorming.

Let's try brainstorming some Edison inventions. Using activity sheet U6-1, you need to come up with six different ideas of something you could create and program using Edison robots. You will only have 45 seconds for each idea before you need to move on to your next idea.

Capture your ideas however you like. You can draw something, write down the idea in words, or do a bit of both! You are not allowed to NOT come up with ideas, however. Remember, there are no 'bad' ideas during a brainstorming session!

Set a timer for 45 seconds and get started on your first idea. As soon as the timer goes off, reset it and move on to the next idea until you have all six ideas done.

After you finish brainstorming, analyse the ideas you came up with. Looking at all your ideas together, you may decide that some of your ideas are better than others. You can then choose one of these ideas to use and move on to designing your creation.

Name_____

Task 2: Design

Once you have at least one idea you think might work, you can start planning and designing. You will need to decompose your idea into smaller parts, design each part, and plan how to tackle each one. At a minimum, you should break your idea into two parts: the physical design using Edison and the program design.

There are lots of different tools you can use to help you design and plan. You can draw out sketches or diagrams, you can create a storyboard, or you can write up an overview. When you design your program, you will probably also want to use pseudocode to help you plan how the program will function. You can always use a mix of approaches too – whatever works best for you!

Using one of your ideas, work out a design for both the physical creation and the program.

Here are some of the things you should think about when designing:

- What will your creation do?
- What will your creation look like?
- What materials could you use to build your creation? How will you attach these to Edison?
- How will the program control Edison to make your creation work?

Remember, this design is just your first one. There's still a lot of learning to do! Your design might not be able to predict everything, and you might have some things you aren't sure about. If you have any questions about how parts of your creation or program can work, make a special note of these questions. These are the types of things you will want to pay close attention to when you build and test.

1. Give your project a name and write a brief description of your idea.

_____

_____

_____

_____

2. What are the different parts you have decomposed your project into to help you design it? Remember, you should have at least two parts: the physical creation using Edison and the program design.

_____

_____

_____

_____

Name_____

3. Work out the design of your physical creation. Use extra space if you need.

Name_____

4. Design your EdScratch program for your creation. Use extra space if you need.

Name_____

Optional: What have you not been able to work out in your design? What questions do you need to pay extra attention to when you build and test? Note them in this space.

Mini challenge!

Is your idea possible? Can it be built? Can it be programmed? The only way to know is to try!

Try turning your idea into a reality by using your design as a guide to help you build and program your creation. Test your invention to see what works and what doesn't. Take what you learn from your test and apply it back into your design. Keep repeating the cycle, applying what you learn to make improvements to each iteration.

Name_____

U6-1.1a Challenge up: Invent an imaginary creature

Does it live under the bed and eat dirty socks? Is it mean and scary, or shy but nice? Does it have 23 legs and a lovely singing voice? Does it dart around or move like a snail?

In this challenge, it's all up to you!

What to do

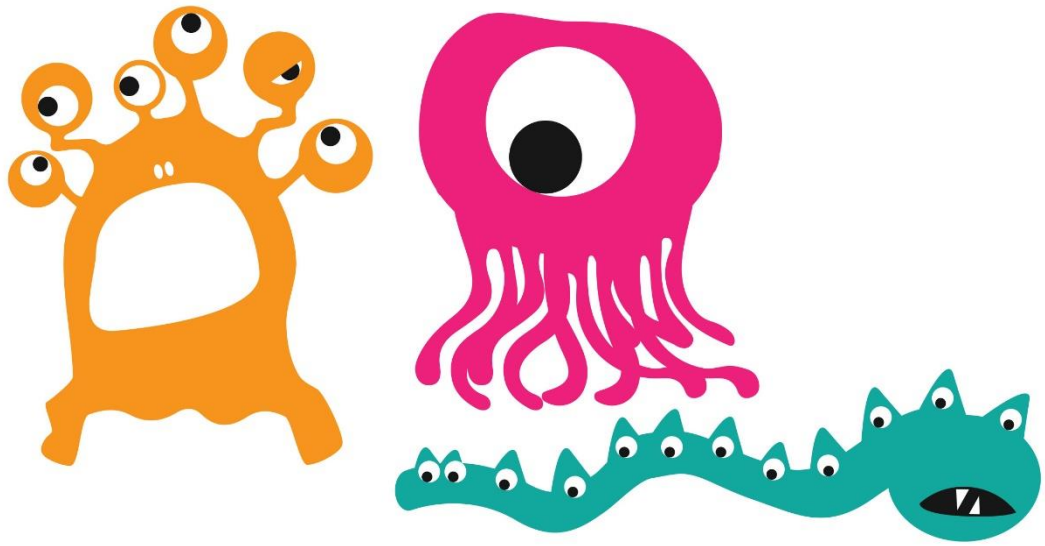
For this challenge, you need to use the design-build-test cycle to create and program an imaginary creature using your Edison robot and EdScratch. What your creature looks like and how it works is up to you, but your creature needs to be able to do the following things:

- ☐ move using Edison's motor outputs
- ☐ use at least one of Edison's sensors to trigger some sort of behaviour from your creature in response to something in its environment

Spend some time brainstorming different ideas. Once you have at least one idea you think might work, move on to designing. Decompose your idea into smaller parts, design each part and plan how to tackle each one. At a minimum, you should break your idea into two parts: the physical design using Edison and the EdScratch program design.

Work out a design for both the physical creation and the program. Build your creature and write your creature's EdScratch program. Test your invention to see what works and what needs improvement.

Apply what you learned from your test back to your design, create the next iteration of your build, and test again. Keep repeating the design-build-test cycle until you get your creature working just like you want.



Name_____

U6-1.1b Challenge up: Invent a cotton ball launcher

Grab a bag of cotton balls, then get ready, aim, and FIRE! It's time to build a cotton ball launcher. Will your cotton balls shoot straight up like rockets? Will you be able to hit a bullseye? Or will your cotton balls fly down the hallway?

In this challenge, it's all up to you!

What to do

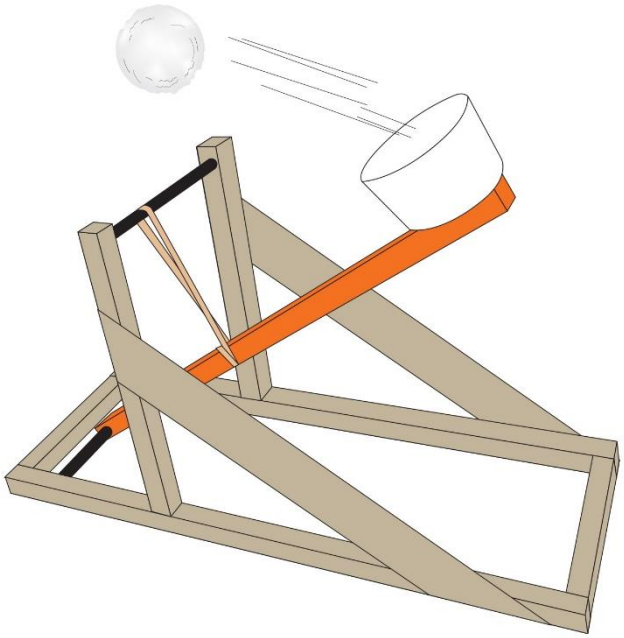
For this challenge, you need to use the design-build-test cycle to create and program a cotton ball launcher using your Edison robot and EdScratch. What your launcher looks like and how it works is up to you, but your launcher needs to be able to do at least one of the following things:

- ☐ throw a cotton ball as high as possible, or
- ☐ throw a cotton ball as far as possible, or
- ☐ throw a cotton ball as accurately as possible

Spend some time brainstorming different ideas. Once you have at least one idea you think might work, move on to designing. Decompose your idea into smaller parts, design each part and plan how to tackle each one. At a minimum, you should break your idea into two parts: the physical design using Edison and the EdScratch program design.

Work out a design for both the physical creation and the program. Build your launcher and write your EdScratch program. Test your invention to see what works and what needs improvement.

Apply what you learned from your test back to your design, create the next iteration of your build, and test again. Keep repeating the design-build-test cycle until you get your launcher working just like you want.



Name_____

U6-1.1c Challenge up: Invent a burglar alarm

You have been entrusted with a valuable treasure to protect. But there are sneaky thieves out to get it! How will you keep the treasure safe and sound?

In this challenge, it's up to you!

What to do

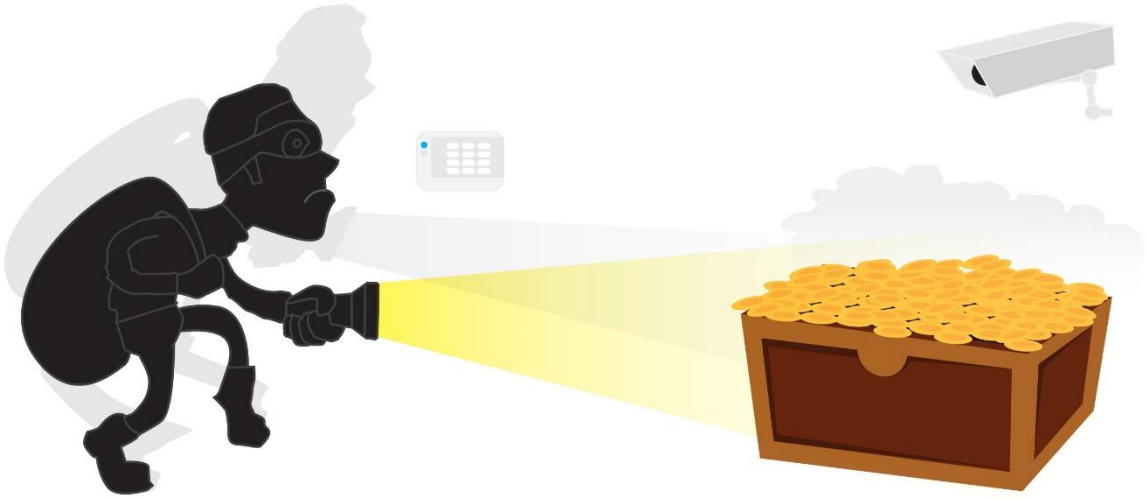
For this challenge, you need to use the design-build-test cycle to create and program a burglar alarm using your Edison robot and EdScratch. What your invention looks like and how it works is up to you, but your burglar alarm needs to be able to do the following things:

- ☐ use at least one of Edison's sensors to trigger the alarm
- ☐ do something to scare the intruder away

Spend some time brainstorming different ideas. Once you have at least one idea you think might work, move on to designing. Decompose your idea into smaller parts, design each part and plan how to tackle each one. At a minimum, you should break your idea into two parts: the physical design using Edison and the EdScratch program design.

Work out a design for both the physical creation and the program. Build your burglar alarm and write your EdScratch program. Test your invention to see what works and what needs improvement.

Apply what you learned from your test back to your design, create the next iteration of your build, and test again. Keep repeating the design-build-test cycle until you get your burglar alarm working just like you want.



Name_____

U6-1.1d Challenge up: Invent a mousetrap

Break out the cheese! Or do mice prefer peanut butter? Whatever you want to use as bait, it's time to get prepared to catch a mouse! What will set off the trap? How will you know if you have caught a mouse?

In this challenge, it's up to you!

What to do

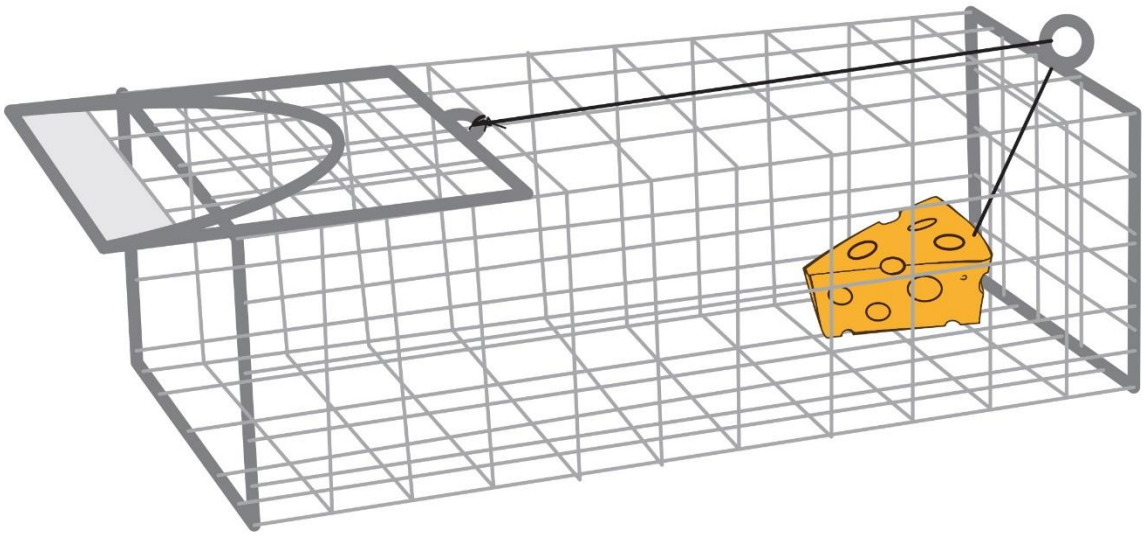
For this challenge, you need to use the design-build-test cycle to create and program a mousetrap using your Edison robot and EdScratch. What your invention looks like and how it works is up to you, but your mousetrap needs to be able to do the following things:

- ☐ use at least one of Edison's sensors to trigger the trap
- ☐ do something to alert you that the trap has been sprung

Spend some time brainstorming different ideas. Once you have at least one idea you think might work, move on to designing. Decompose your idea into smaller parts, design each part and plan how to tackle each one. At a minimum, you should break your idea into two parts: the physical design using Edison and the EdScratch program design.

Work out a design for both the physical creation and the program. Build your mousetrap and write your EdScratch program. Test your invention to see what works and what needs improvement.

Apply what you learned from your test back to your design, create the next iteration of your build, and test again. Keep repeating the design-build-test cycle until you get your trap working just like you want.



Name_____

U6-1.1e Challenge up: Invent a combination safe

Where do you keep your most valuable possessions? Why not put your belongings in a combination safe with a robotic lock! What will your safe look like? How will you enter the code to get into the safe? What will the code be?

In this challenge, it's all up to you!

What to do

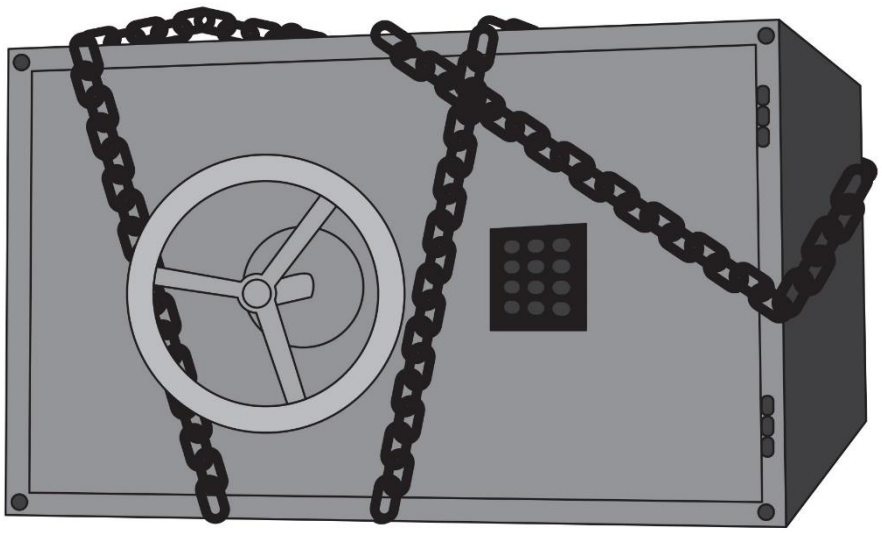
For this challenge, you need to use the design-build-test cycle to create and program a combination safe that uses your Edison robot as a lock. What your invention looks like and how it works is up to you, but your lock needs to open your safe only when the right combination is entered. Your lock should do one of the following things:

- only open for the right sequence of round button and triangle button presses, or
- only open for the right sequence of TV or DVD remote control button presses, or
- only open for the right sequence of IR messages from another Edison robot, or
- only open for the right combination of a mix of button presses, remote-control signals or IR messages

Spend some time brainstorming different ideas. Once you have at least one idea you think might work, move on to designing. Decompose your idea into smaller parts, design each part and plan how to tackle each one. At a minimum, you should break your idea into two parts: the physical design using Edison and the EdScratch program design.

Work out a design for both the physical creation and the program. Build your combination safe and write your EdScratch program. Test your invention to see what works and what needs improvement.

Apply what you learned from your test back to your design, create the next iteration of your build, and test again. Keep repeating the design-build-test cycle until you get your combination safe working just like you want.



Name_____

U6-1.2 Let's explore a haunted house

All of Edison's different sensors and outputs mean that there are lots of things you can do with Edison robots. Using EdScratch, you can program Edison to drive, play musical sounds, detect obstacles, follow black lines and much, much more.

You can also turn Edison into other things, building inventions with the robot and using Edison's sensors and outputs in different ways. You have seen, for example, how you can use the blocks in the Drive category in EdScratch not to drive Edison around, but instead to power creations using Edison's motors. Likewise, you can use Edison's various sensors to do things in unexpected ways.

In this project, you are going to work together, using everything you have learned about Edison and EdScratch to turn your robots into monster hunters. You will use your monster-hunting robots to help you detect and trap different ghouls in a haunted house.

About the haunted house

There are different types of monsters living in the haunted house. Luckily, each type of monster only hangs out in one room. Your job is to design, create, and program inventions using Edison to detect and trap all the different monsters, one room at a time.

One room in the haunted house has ghosts in it. These white ghosts hang out on a black floor. You will need to program an Edison robot to drive around the room with the ghosts and locate each one so that you can come in and remove each ghost.

There are two other rooms in the haunted house as well. What monsters are in each of these rooms? How will Edison help you detect, and even trap, these other creatures? That part is up to you!

Task 1: Detect the ghosts

Your haunted house has a room with ghosts in it. Your job is to clear this room of all the ghosts using an Edison robot to help you.

There are two major parts to this first task. One, you need to create a test space to represent the room and ghosts. Two, you need to program Edison to locate all the ghosts, alerting you each time a ghost is found, so you can come in and remove it.

The test space

You need to make a test space to represent the room and the ghosts. Your room needs walls and a black floor. You will need something to represent the ghosts as well, such as white masking tape. You need to have at least three ghosts in your room.

The program

Usually, we use Edison's line tracking sensor to detect black (non-reflective) on a white (reflective) background. This time you need to do just the opposite: find white 'ghosts' on a black surface.

You need to write a program so that Edison will drive around the test space. If Edison detects a ghost, the robot should stop moving, 'trapping' the ghost below it. Edison then needs to do something, like play a tune, to let you know that the robot has detected a ghost. This will be your

cue to come in and remove that ghost from the room. Edison should wait until you have gotten the ghost it found out of the room, then go back to driving. Once Edison has detected all of the ghosts, the robot should signal to you that all the ghosts are gone, and the program should end.



Hint!

Edison's line tracking sensor works in a special way. Remember that the line tracker measures the amount of reflected light it detects from underneath the robot and stores that measurement as a value. The more light that is detected, the higher the value. Readings with high values are seen as 'reflective' to the robot.

When the line tracker first comes on, the sensor takes a reading of the reflected light coming from the surface below the robot. The robot uses this initial value to determine what is 'reflective'. In other words, the robot sets the first value the line tracker generates as the value of 'reflective' and uses this to determine if a new value is 'reflective' or 'non-reflective'. This is why you always start a program using the line detector on a white surface.

In your ghost hunter program, you also need to start your program on a white surface. Put Edison on a piece of white paper before you run your program, so that when you first start your program the robot can set the 'reflective' value. Your program should then have Edison wait until you do something, like press one of Edison's buttons, to move on to the rest of the program and start looking for ghosts. You will be able to use this time to move Edison from the white paper to the black surface of your test area.

Design your ghost hunting program, write it in EdScratch, then test it out using your Edison robot in your 'ghost room' test space. Note what works in your program and what needs to be improved. Apply your learnings to your program design, write your changes in EdScratch and test again. Keep iterating until you get Edison to run the program, successfully detecting all the ghosts.

Task 2: Create your other haunted house rooms

Your haunted house needs two other rooms with a different type of monster in each one. For each other room, you need to decide what type of monster is in that room, then create a test space to represent the room and the monsters. You also need to use the design-build-test cycle to create and program a way to detect and trap all the monsters in each room using at least one Edison robot and EdScratch.

Spend some time brainstorming different ideas about what monsters you want to use and how you will detect and trap them. Once you have at least one idea you think might work, move on to designing. Decompose your idea into smaller sections, design each part, and plan how to tackle each one. At a minimum, you should break your idea into three parts for each room: the test space, the physical creation using Edison robots, and the EdScratch program design.

Name_____

Room 1

Answer the questions and use this space to design the room, the monsters, and your trap. Once you have your design for each part, build your room, the monsters, and your monster hunting invention. Program your invention in EdScratch then test your creation in your room to see what works and what needs improvement.

Apply what you learned from your test back to your design, create the next iteration of your build and test again. Keep repeating the design-build-test cycle until you are able to detect and trap all the monsters.

1. What monsters are in this room? How many are there? Write a brief description of what is in the room and how you plan on detecting and trapping the monsters.

_____

_____

_____

_____

_____

_____

2. What are the different parts you have decomposed your project down into to help you design it? Remember, you should have at least three parts: the test space, the physical monster hunter (to detect and trap the monsters) using Edison, and the program design.

_____

_____

_____

_____

_____

Name_____

3. Design your room (test space). Use extra space if you need.

Name_____

4. Design your monster hunter (Edison creation). Use extra space if you need.

Name_____

5. Design your EdScratch program for your creation. Use extra space if you need.

Name_____

Room 2

Answer the questions and use this space to design the room, the monsters, and your trap. Once you have your design for each part, build your room, the monsters, and your monster hunting invention. Program your invention in EdScratch, then test your creation in your room to see what works and what needs improvement.

Apply what you learned from your test back to your design, create the next iteration of your build, and test again. Keep repeating the design-build-test cycle until you are able to detect and trap all the monsters.

6. What monsters are in this room? How many are there? Write a brief description of what is in the room and how you plan on detecting and trapping the monsters.

_____

_____

_____

_____

_____

_____

7. What are the different parts you have decomposed your project down into to help you design it? Remember, you should have at least three parts: the test space, the physical monster hunter (to detect and trap the monsters) using Edison, and the program design.

_____

_____

_____

_____

_____

Name_____

8. Design your room (test space). Use extra space if you need.

Name_____

9. Design your monster hunter (Edison creation). Use extra space if you need.

Name_____

10. Design your EdScratch program for your creation. Use extra space if you need.

Activity sheet U6-1: Six ideas

1

2

3

4

5

6



Index

Algorithms, 153, 154, 155, 156, 160
Barcodes, 11, 12, 13, 14, 15, 16, 17, 18, 169, 171
Bugs, 27, 30, 31, 64, 65, 66, 67, 130, 131, *See* debugging
Buttons, 7, 11, 12, 13, 14, 15, 17, 18, 26, 27, 48, 106, 108, 110, 111, 131, 132, 139, 141, 144, 149, 168, 169, 170, 171, 174, 190, 194, 195, 205, *See* inputs, sensors
Buzzer, 48, 109, *See* sound sensor
Comments, 112, 113, 114, 115, 116, 117
Computational thinking, 36, 37, 64, 220
Condition, 126, 127, 128, 129, 131, 134, 135, 137, 138, 172, 190, *See* conditionals
Conditionals, 126, 127, 128, 129, 130, 134, 135, 137, 140, 146, 148, *See* condition
Data, 142, 186, 201, 202, 204, 207, 212, 216, *See* values
Debugging, 64, 67, 113, 115, 188
Decomposition, 219, 221, 225, 226, 227, 228, 229, 231
Definite loop, 86, 88, 89, 90, 91, 93, 97, *See* loops
Detect obstacles, 13, 15, 158, 159, 160, 161, 162, 163, 164, 165, 178, 230
Drive, 7, 12, 15, 17, 20, 21, 22, 43, 44, 45, 51, 52, 54, 57, 73, 74, 75, 85, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97, 102, 103, 104, 141, 142, 150, 152, 154, 159, 161, 164, 172, 173, 174, 175, 198, 199, 200, 201, 208, 210, 230, *See* motors, wheels
Event, 107, 108, 128, 129, 131, 161, 203, *See* interrupts
Expressions, 11, 186, 187, 188
How to program in EdScratch, 23
If statements, 134, 137, 140, 141, 146, *See* if-else
If-else, 135, 138, 140, 142, 143, *See* if statements
Indefinite loop, 97, 127, *See* loops
Infrared, 13, 158, 159, 160, 161, 163, 165, 166, 168, 170, 189, 204, 207, 212
Input parameters, 44, 50, 52, 53, 54, 55, 58, 59, 68, 69, 70, 87, 88, 89, 90, 91, 93, 98, 112, 116, 127, 137, 161, 166, 167
Input-process-output cycle, 47
Inputs, 44, 47, 48, 50, 52, 53, 54, 55, 58, 59, 68, 69, 70, 71, 87, 88, 89, 90, 91, 93, 98, 112, 116, 127, 137, 150, 159, 161, 166, 167, 186, 196, 197, 199
Interrupts, 106, 108, 109, 110, 111
IR messages, 30, 65, 165, 166, 167, 178, 204, 205, 206, 207, 209, 210, 211, 212, 214, 215, 216
LED, 7, 47, 48, 49, 51, 102, 105, 108, 109, 110, 111, 149, 150, 151, 158, 161, 163, 165, 166, 189, 191, 204, 207, 212
Light sensor, 13, 149, 158, 165, 189, 190, 191, 192, 204, 207, 212
Line tracking sensor, 8, 14, 149, 150, 151, 152, 153, 154, 155, 156, 157, 178, 189, 190, 201, 203, 230
Logical error, 65, 67, 69, 130, 132, 133, *See* debugging
Loops, 84, 85, 86, 88, 89, 90, 91, 93, 95, 97, 98, 99, 100, 101, 102, 103, 104, 105, 106, 127, 140, 142, 143, 146, 148, 196, 197
Maze, 44, 45, 51, 162, 164, 179, 207, 208
Motors, 7, 20, 21, 22, 43, 47, 68, 69, 70, 71, 73, 74, 85, 95, 144, 168, 204, 230, *See* wheels
Music, 58, 59, 60, 61, 62, 63, 67, 105, 109, 212, 213, 230, *See* song, tune
Outputs, 47, 48, 49, 51, 58, 68, 85, 88, 89, 95, 105, 144, 161, 163, 167, 168, 171, 204, 206, 230
Patterns, 54, 56, 85, 90, 91, 94, 100, 102, 103, 104, 105, 198, 214
Phototropism, 192, 193
Problem-solving, 64, 220, *See* computational thinking
Pseudocode, 145, 146, 147, 148, 152, 154, 156, 157, 160, 193, 221
Remote control, 17, 18, 19, 20, 21, 22, 165, 168, 169, 170, 171, 174

Sensors, 7, 149, 150, 158, 159, 162, 178, 189, 190, 192, 201, 202, 230, *See* inputs, outputs
Sequence, 37, 39, 40, 41, 42, 44, 48, 57, 59, 95, 96, 98, 101, 106, 128, 137, 140, 211
Song, 58, 60, 61, 62, 63, 99, 105, 212, 213, 214, 215, *See* music
Sound sensor, 12, 48, 109, *See* buzzer
Subroutine, 107, 108, 109, 110, 111
Syntax, 65, 67, 106, 112
Syntax error, 65, 67, *See* debugging
Tune, 17, 58, 99, 212, 230, *See* music
Values, 57, 70, 71, 87, 88, 89, 90, 149, 167, 187, 188, 189, 195, 196, 197, 199, 200, 201, 204, 205, 206, 207, 209, 210, 211, 212, 214, 215, 216, *See* data
Variables, 185, 194, 195, 196, 197, 199, 200, 201, 202, 203, 204, 206, 207, 210, 211, 212, 215, 216
Warning messages, 27, 30, 65, *See* bugs, debugging
Wheels, 8, 10, 20, 21, 22, 68, 73, 74, 144, *See* motors