Edison
robotics for everyone

EdPy worksheets

Student worksheets and activity sheets

The EdPy Lesson Plans Set by Brenton O'Brien, Kat Kennewell and Dr Sarah Boyd is licensed under a Creative Commons Attribution-ShareAlike 4.0 International License.

Name________________________________

Lesson 7: Worksheet 7.1 – Infrared obstacle detection

In this activity, you will learn more about infrared (IR) light and how Edison can use IR to detect obstacles.

What is infrared (IR) light?

There is a wide range of light, some of which is visible to the human eye and some of which is not. Infrared, also called IR, is not visible to humans.

> ***Did you know?*** Even though people cannot see it, infrared is a type of light. Therefore, it will work in the dark. That's why you can turn on a TV set using a remote control even if there are no lights on in the room!

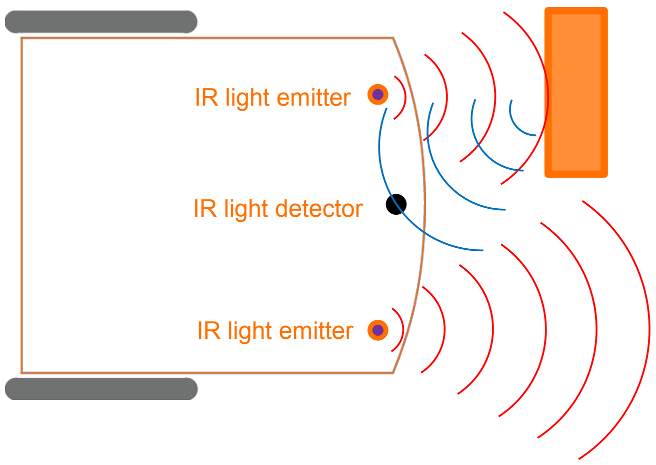
Edison and infrared

The Edison robot is equipped with an infrared system that gives the robot 'vision' of a sort. This infrared system allows Edison to detect obstacles around the robot.

Edison's infrared system is made up of two IR light emitter diodes (or LEDs) on the front. One is on the left, and one is on the right. Edison also has an IR sensor on the front, directly in the middle.

For Edison to detect obstacles, infrared light is emitted forward from the left and right IR LEDs. If the IR light encounters an obstacle, such as a wall, it is reflected back towards Edison. Edison's IR sensor then detects the reflected light.

Look at the illustration below:



Edison emits IR light (shown in red) from both the robot's left and right IR LEDs. Reflected IR light (shown in blue) bounces off obstacles and is detected by Edison's IR sensor.

In this picture, there is an obstacle in front of Edison's left side, but not the right side. That's why only IR light from the left emitter is reflected.

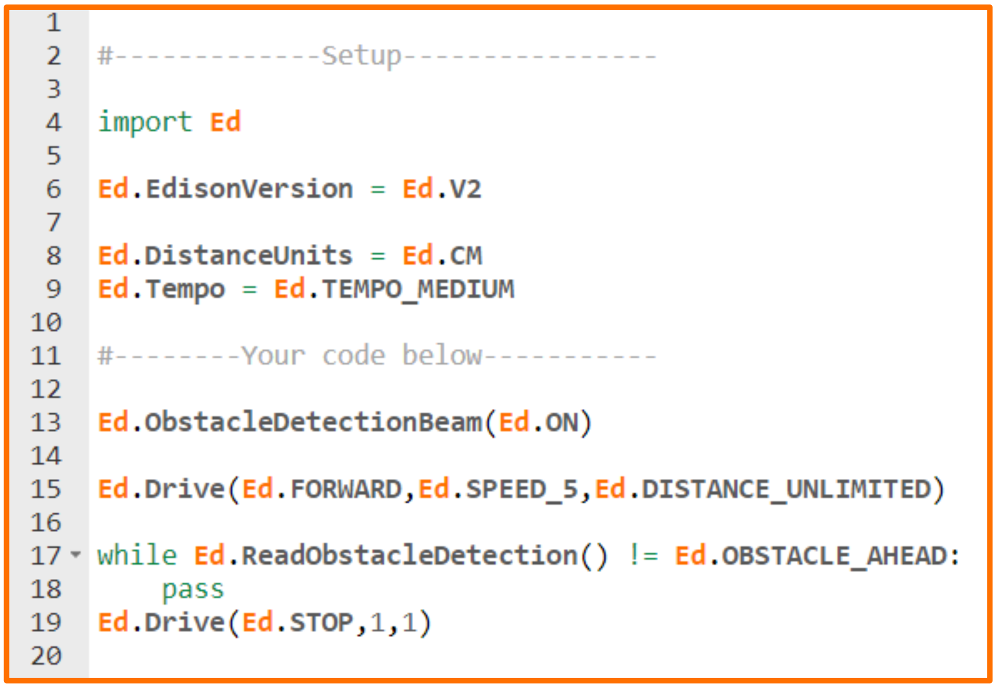
From the received signal Edison can determine that there is an obstacle to the left, but no obstacle to the right.

Name________________________________

Lesson 7: Worksheet 7.2 – Detect an obstacle and stop

In this activity, you will need to write a program to make your Edison robot drive until it encounters an obstacle and then stop.

Look at the following program:



```
 1
 2  #------------Setup---------------
 3
 4  import Ed
 5
 6  Ed.EdisonVersion = Ed.V2
 7
 8  Ed.DistanceUnits = Ed.CM
 9  Ed.Tempo = Ed.TEMPO_MEDIUM
10
11  #--------Your code below-----------
12
13  Ed.ObstacleDetectionBeam(Ed.ON)
14
15  Ed.Drive(Ed.FORWARD,Ed.SPEED_5,Ed.DISTANCE_UNLIMITED)
16
17  while Ed.ReadObstacleDetection() != Ed.OBSTACLE_AHEAD:
18      pass
19  Ed.Drive(Ed.STOP,1,1)
20
```

This program tells Edison to drive until it encounters an obstacle.

There are a couple of important things to notice about this program.

Look at line 13 of the program. This line turns Edison's obstacle detection beam to 'on'. Whenever you want to use Edison's obstacle detection beam in an EdPy program, you always need to turn the beam to 'on' before the beam is used in the program.

Now, look at line 15 of the program. This line sets the speed to 5 in this program. When using obstacle detection, you need to use a slightly lower speed to allow the robot to detect an obstacle before colliding with it. If the speed is too fast, the robot will crash into obstacles before being able to detect them.

Your turn:

Task 1: Select your obstacles

You need to choose good obstacles to use with Edison. 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 (e.g. not black) and at least as tall as Edison.

For this program, the wall of the room would be a good obstacle.

Name_______________________________

Task 2: Get your Edison ready

When you want to run a program using obstacle detection, it is a good idea to check that your Edison robot's obstacle detection is calibrated to the distance you want. Use activity sheet 7.1 to calibrate your Edison. You may need to calibrate your robot with custom sensitivity to make sure your robot can detect and stop at your obstacle.

Task 3: Write and run the program

Write the program using the EdPy app and download it to your Edison robot. Then run the program to see how it works.

Experiment using different obstacles to see what Edison can and cannot detect. You can also try setting Edison's obstacle detection calibration to different distances to see what happens.

1. Delete the 'Ed.ObstacleDetectionBeam(Ed.ON)' line out of your program. Try to download and run the adjusted program. What happened? Why does that happen?

____

____

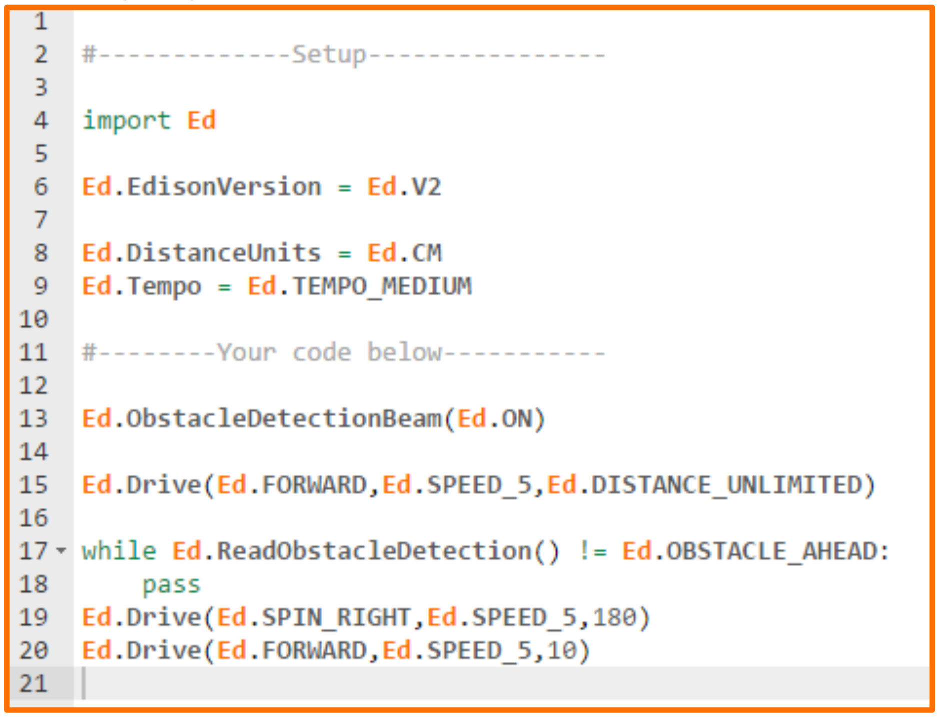
____

Name________________________________

Lesson 7: Worksheet 7.3 – Obstacle avoidance

In this activity, you will write a program to make your Edison robot drive until it encounters an obstacle, turn and drive away.

Look at the following program:



```
#------------Setup---------------

import Ed

Ed.EdisonVersion = Ed.V2

Ed.DistanceUnits = Ed.CM
Ed.Tempo = Ed.TEMPO_MEDIUM

#--------Your code below-----------

Ed.ObstacleDetectionBeam(Ed.ON)

Ed.Drive(Ed.FORWARD,Ed.SPEED_5,Ed.DISTANCE_UNLIMITED)

while Ed.ReadObstacleDetection() != Ed.OBSTACLE_AHEAD:
    pass
Ed.Drive(Ed.SPIN_RIGHT,Ed.SPEED_5,180)
Ed.Drive(Ed.FORWARD,Ed.SPEED_5,10)
```

This program tells Edison to drive forward until an obstacle is detected. Once Edison detects an obstacle, the program tells Edison to turn 180° and drive away for 10 cm.

Your turn:

Task 1: Write and run the program

Write the program using the EdPy app and download it to your Edison robot. Then run the program to see how it works.

After you have run the program, look at it again and think about how you could change the program. Try to modify the code so that the robot behaves differently when it detects an obstacle ahead. (*Hint*: try using sound and lights.)

1. Think about how you could improve the original program. What could you change so that the program does more than just turn and drive away after encountering one obstacle?

____

____

____

Name_______________________________

Task 2: Syntax errors and logical errors

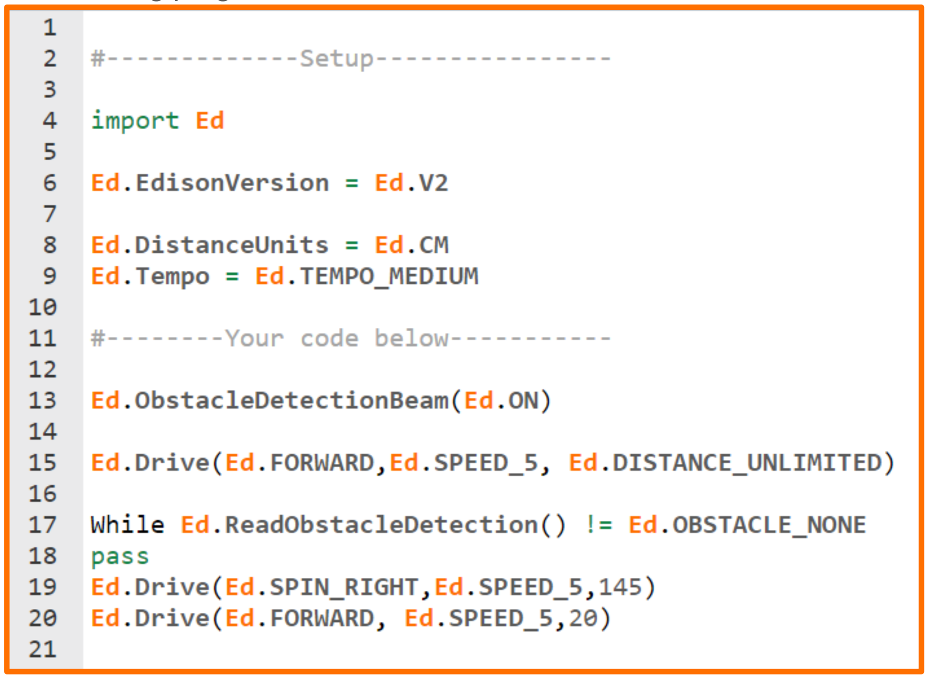
Programmers often make ‘typo’ like errors, called syntax errors. It is important to be good at spotting your own syntax errors so you can correct your code.

You can also get another type of error, called a logical error, when you program. In coding, any error that is not a syntax error is a logical error.

When there is a logical error in a program, the code doesn’t behave the way that the programmer expects it to. In EdPy, if you have a logical error in your program, the program will usually still download to Edison. When you run the program, however, it won’t behave the way you think it should.

A logical error can be as simple as using the wrong function or leaving out a function. An example of a logical error would be writing a program that uses obstacle detection but not turning the obstacle detection beam on.

Look at the following program:



```

#-------------Setup----------------

import Ed

Ed.EdisonVersion = Ed.V2

Ed.DistanceUnits = Ed.CM
Ed.Tempo = Ed.TEMPO_MEDIUM

#--------Your code below-----------

Ed.ObstacleDetectionBeam(Ed.ON)

Ed.Drive(Ed.FORWARD,Ed.SPEED_5, Ed.DISTANCE_UNLIMITED)

While Ed.ReadObstacleDetection() != Ed.OBSTACLE_NONE
pass
Ed.Drive(Ed.SPIN_RIGHT,Ed.SPEED_5,145)
Ed.Drive(Ed.FORWARD, Ed.SPEED_5,20)

```

The program is meant to make the Edison robot drive until it encounters an obstacle, then turn 135° and drive away from the obstacle for 20 cm.

There are five errors in the program. Can you identify all five errors in the program and determine if they are syntax errors or logical errors? Fill in your answers in the table on the next page.

Name_______________________________

Hint: You can write this program in EdPy and use the ‘Check Code’ button to help you find the syntax errors.

Error #	**Line #**	**Error type (syntax or logical)**	**Error description**
1			
2			
3			
4			
5			

Name________________________________

Lesson 7: Worksheet 7.5 – Right and left obstacle detection

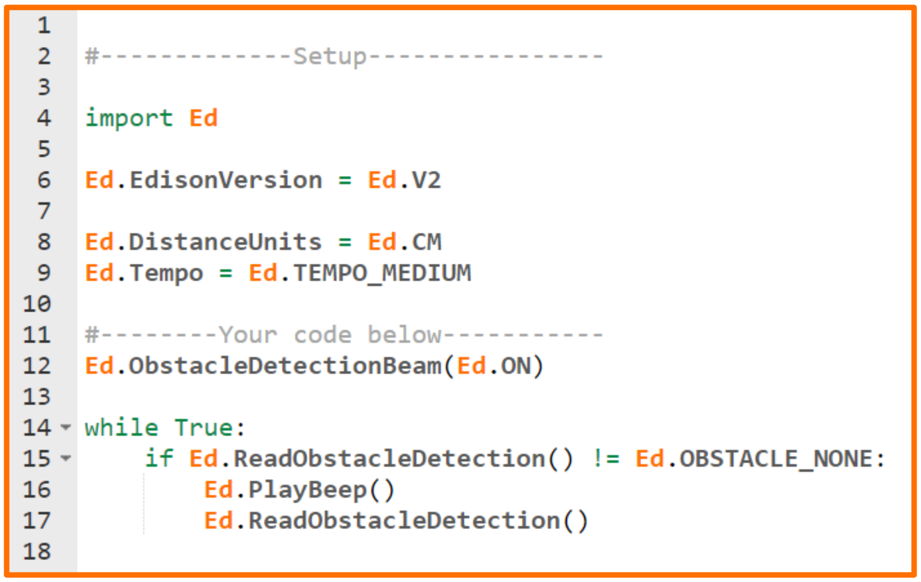
In this activity, you will write a program for Edison to react to obstacles on the left or right of the robot. To do this, we will use ‘if statements’.

If statements

An important part of coding is making decisions. The most common way to do this is to use an ‘if statement’.

An ‘if statement’ asks whether a condition is true or false. If the result is true, then the program executes the block of statements following the if statement. If the result is false, the program ignores the statements inside the if statement and moves to the next line of code outside of the if statement.

Look at the following program:



```

#------------Setup---------------

import Ed

Ed.EdisonVersion = Ed.V2

Ed.DistanceUnits = Ed.CM
Ed.Tempo = Ed.TEMPO_MEDIUM

#--------Your code below----------
Ed.ObstacleDetectionBeam(Ed.ON)

while True:
    if Ed.ReadObstacleDetection() != Ed.OBSTACLE_NONE:
        Ed.PlayBeep()
        Ed.ReadObstacleDetection()

```

This program uses an if statement to give the robot the ability to make decisions without human guidance. When a robot can make decisions on its own in this way, it is called an autonomous robot.

Your turn:

Task 1: Beep if there’s an obstacle

Write the above program using the EdPy app, download it to your Edison robot and run the program. Then try moving an obstacle, like your hand, in and out of Edison’s obstacle detection beam to see what happens.

Name________________________________

Task 2: Beep if there's an obstacle, else turn on the light

In addition to telling a program what to do when an if statement condition is true, you can also tell a program what to do if that condition is false.

If, else

Using if statements with 'else' allows us to write programs making more complicated decisions. If/else statements are basically a way of making a decision between two things.

Watch Bill Gates, founder of Microsoft, explain if/else statements and decision making in programming: https://youtu.be/fVUL-vzrIcM

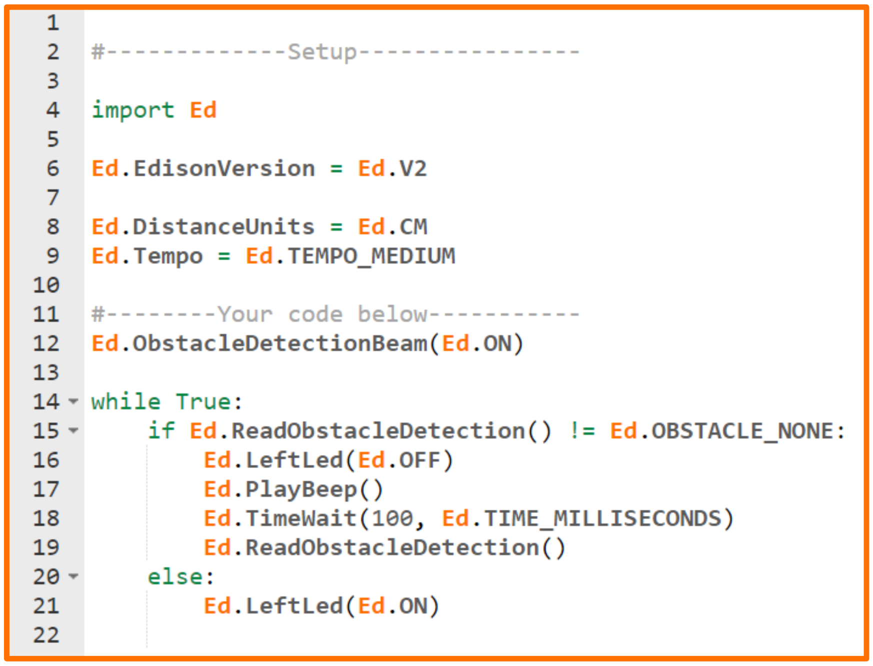
In Python, the syntax for if/else is:

```
if expression:
    statement(s)
else:
    statement(s)
```

The program moves sequentially from the top down, starting with the if condition. If the if statement is true, the program runs the indented code for the if expression and skips the else. If the if statement is false, however, the program skips that section of indented code and runs the 'else' indented code instead.

Name________________________________

Write the following program:



```

#------------Setup---------------

import Ed

Ed.EdisonVersion = Ed.V2

Ed.DistanceUnits = Ed.CM
Ed.Tempo = Ed.TEMPO_MEDIUM

#--------Your code below----------
Ed.ObstacleDetectionBeam(Ed.ON)

while True:
    if Ed.ReadObstacleDetection() != Ed.OBSTACLE_NONE:
        Ed.LeftLed(Ed.OFF)
        Ed.PlayBeep()
        Ed.TimeWait(100, Ed.TIME_MILLISECONDS)
        Ed.ReadObstacleDetection()
    else:
        Ed.LeftLed(Ed.ON)

```

Download and run the program. Try moving an obstacle in and out of Edison's obstacle detection beam to see what happens.

This program has two pathways: one for if an obstacle is detected and one for if no obstacle is detected.

If, elif, else

You can also make a program which makes a decision using more than two conditions. To do this, you use another Python syntax structure:

```
if expression:
    statement(s)
elif expression:
    statement(s)
else:
    statement(s)
```

'Elif' is how you say 'else if' in Python. You can use elif to write a program with multiple if conditions.

A program using if/elif/else still moves sequentially from the top down. Once the program runs any indented code inside any part of the if statement structure, it will skip the rest of the structure and move on to the next line of code outside the structure.

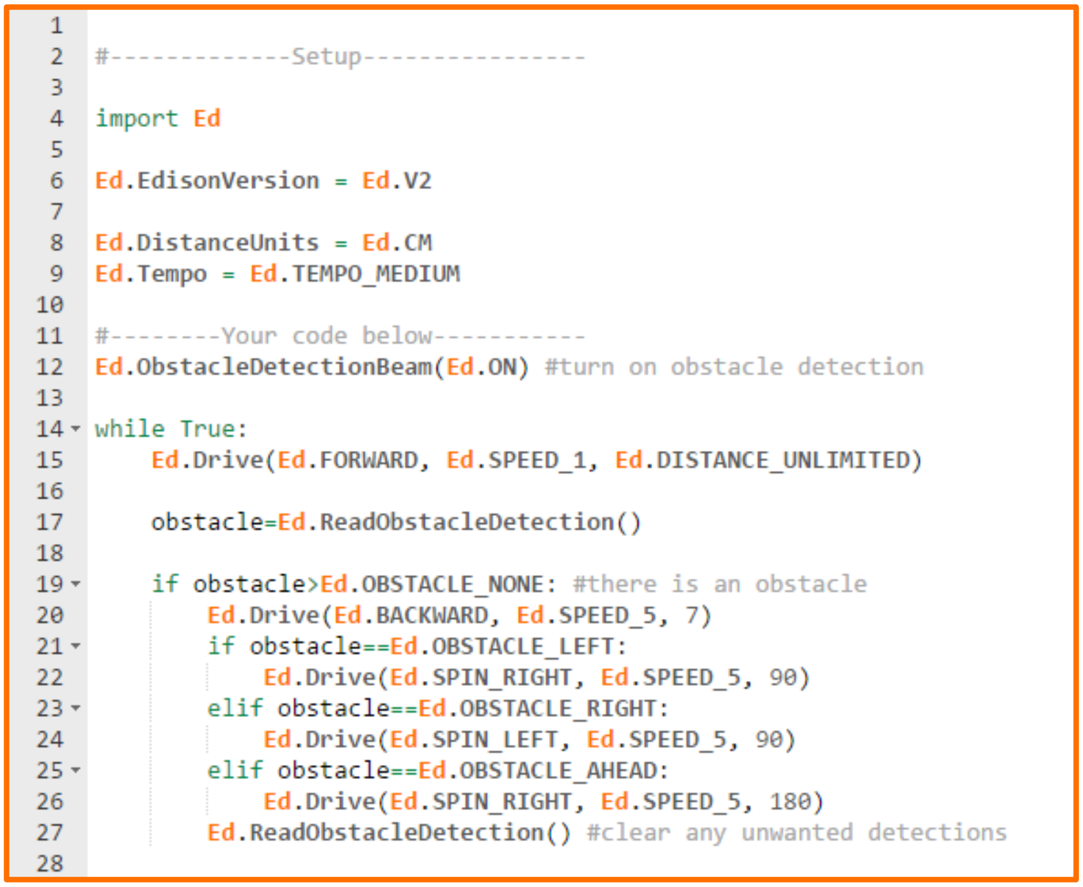
This means that if the if statement at the top is true, the program runs the indented code for the if expression and skips any elif sections as well as the else section if there is one. If the if statement is false, however, the program skips that section of indented code and moves to the first elif section.

Name________________________________

Again, if the first elif condition is true, the program runs its indented code and skips everything below it in the if statement structure (any other elifs and the else condition if there is one). If this elif condition is false, the program moves to the next part of the if statement structure and so on.

Task 3: Detect an obstacle on the left or the right

Look at the following program:



```

#-------------Setup---------------

import Ed

Ed.EdisonVersion = Ed.V2

Ed.DistanceUnits = Ed.CM
Ed.Tempo = Ed.TEMPO_MEDIUM

#--------Your code below-----------
Ed.ObstacleDetectionBeam(Ed.ON) #turn on obstacle detection

while True:
    Ed.Drive(Ed.FORWARD, Ed.SPEED_1, Ed.DISTANCE_UNLIMITED)

    obstacle=Ed.ReadObstacleDetection()

    if obstacle>Ed.OBSTACLE_NONE: #there is an obstacle
        Ed.Drive(Ed.BACKWARD, Ed.SPEED_5, 7)
        if obstacle==Ed.OBSTACLE_LEFT:
            Ed.Drive(Ed.SPIN_RIGHT, Ed.SPEED_5, 90)
        elif obstacle==Ed.OBSTACLE_RIGHT:
            Ed.Drive(Ed.SPIN_LEFT, Ed.SPEED_5, 90)
        elif obstacle==Ed.OBSTACLE_AHEAD:
            Ed.Drive(Ed.SPIN_RIGHT, Ed.SPEED_5, 180)
        Ed.ReadObstacleDetection() #clear any unwanted detections

```

This program has three different paths that it can take when an obstacle is detected based on where the detected obstacle is in relation to Edison.

Write the program using the EdPy app and download it to your Edison robot. Then run the program to see how it works.

2. When running this program, explain in your own words what the robot does when:

Obstacle detected ahead: ____

Obstacle detected on right: ____

Obstacle detected on left: ____

Optional Calibration

Lesson 7: Activity sheet 7.1 – Calibrate obstacle detection

You can regulate the sensitivity of Edison's obstacle detection system. By making the obstacle detections system more sensitive, Edison can detect obstacles further away. By making the system less sensitive, Edison will only detect very close obstacles. Use this activity sheet to adjust your Edison's obstacle detection system.

Step 1: read the barcode

1. Place Edison on the right side, facing the barcode
2. Press the record (round) button three times
3. Edison will drive forward and scan the barcode

Barcode – Calibrate obstacle detection

Step 2: set maximum sensitivity

After scanning the barcode, set Edison down on a table or desk and remove any obstacles in front of Edison. Then press the play (triangle) button. Edison is now in calibration mode.

The left sensitivity is calibrated first.

1. Repeatedly press the play (triangle) button, which increases sensitivity, until the red LED on the left is flickering.
2. Repeatedly press the record (round) button, which decreases the sensitivity, until the LED completely stops flickering.
3. Press the stop (square) button to switch over to calibrate the right side.
4. Repeatedly press the play (triangle) button until the right red LED is flickering. Now repeatedly press the record (round) button until the LED completely stops flickering.
5. Press the stop button to complete the calibration.

Special note: custom sensitivity

It is possible to set the distance that obstacles are detected. To do this, scan the 'calibrate obstacle detection' barcode, place an obstacle in front of Edison at the distance you want Edison to detect obstacles, press the play button and then repeat steps 1 through 5.