EdPy worksheets

Student worksheets and activity sheets

Name_____

Lesson 8: Worksheet 8.2 – Drive until a black line

In this activity, you will write a program so that your Edison robot will drive forward on a white (reflective) surface until a black (non-reflective) line is crossed.

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.LineTrackerLed(Ed.ON)
14
15 Ed.Drive(Ed.FORWARD,Ed.SPEED_6,Ed.DISTANCE_UNLIMITED)
16
17 while True:
18     if Ed.ReadLineState() == Ed.LINE_ON_BLACK:
19         Ed.PlayBeep()
20         Ed.Drive(Ed.STOP,Ed.SPEED_6,0)
```

Look at line 13. This line calls the function Ed.LineTrackerLed() and turns the state to ‘on’.

Just like with Edison’s obstacle detection beam, to use the line tracking sensor in a program, you must first turn the sensor on. Turning the line tracking sensor on will also activate the line tracker’s red LED.

Now, look at line 19. This line calls the Ed.PlayBeep() function. This line doesn’t affect the way the line tracking program works. Instead, this line’s purpose is for debugging.

Debugging

Debugging is the process of finding ‘bugs’ or errors in your program. Often, programmers will put lines like line 19 in this program into their code to keep track of the program’s flow.

Let’s say you run your program, but the robot does not stop at the black line. There are two possible reasons: (1) the robot may not be detecting the black line or (2) there could be a mistake in the final Ed.Drive() command.

If we hear the beep played, we would know that the black line was detected. Therefore, we know that the error was in the next command. This extra debugging code helps us to determine the error more easily.

Other functions could also be used for debugging, such as the Ed.LeftLed() command. For example, you could use this command to turn the left LED on to indicate that a certain point in the program has been reached.

Name_____

Lesson 8: Worksheet 8.3 – Drive inside a border

In this activity, you need to write a program that will keep Edison inside a black border using the robot's line tracking sensor.

First, we need to plan out the program using pseudocode.

Pseudocode

Planning out your program before you begin coding is an important and useful skill in programming.

One way to do this is by using a flowchart to summarise your program flow, which you learned about in lesson 6. Pseudocode is another way to represent your program before you begin coding.

Pseudocode is a way of writing out a program in a simplified, easy to read way. Pseudocode resembles a simplified programming language, but it isn't based on any specific programming language, so it is syntax-free. Instead, pseudocode uses English language to describe what the program will do. That's why pseudocode is also called 'structured English'.

When you plan out your program using pseudocode, you should indent in the same way you would in the programming language so that the pseudocode is easy to read and understand.

Here is an example of some pseudocode describing a program which makes the Edison robot stay within a non-reflective border using the line tracking sensor:

```
Turn the line tracker on
Loop forever
    Drive Forward
    If the robot detects a black line then
        Stop
        Play a beep
        Spin right 135°
```

Here is the corresponding code in EdPy:

```
11  #--------Your code below-----------
12
13  Ed.LineTrackerLed(Ed.ON)
14
15  while True:
16      Ed.Drive(Ed.FORWARD, Ed.SPEED_3, Ed.DISTANCE_UNLIMITED)
17      if Ed.ReadLineState() == Ed.LINE_ON_BLACK:
18          Ed.Drive(Ed.STOP, Ed.SPEED_3, Ed.DISTANCE_UNLIMITED)
19          Ed.PlayBeep()
20          Ed.Drive(Ed.SPIN_RIGHT, Ed.SPEED_5, 135)
21
```

Compare the pseudocode to the program. Do you see how the pseudocode translated into the Python program?

Name_____

Your turn:

Write 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.LineTrackerLed(Ed.ON)
14
15 while True:
16     Ed.Drive(Ed.FORWARD, Ed.SPEED_3, Ed.DISTANCE_UNLIMITED)
17     if Ed.ReadLineState() == Ed.LINE_ON_BLACK:
18         Ed.Drive(Ed.STOP, Ed.SPEED_3, Ed.DISTANCE_UNLIMITED)
19         Ed.PlayBeep()
20         Ed.Drive(Ed.SPIN_RIGHT, Ed.SPEED_5, 135)
21
```

Download the program to your Edison robot and run it to see how it works.

Use activity sheet 8.2 as a border to test the program. You can also create your own border using a large piece of paper and a thick black marker or use black electrical tape on a white desk or the floor to create a large border.

Modify the program:
Try removing the stop by removing line 18 of the program. Then experiment with changing the program to use different speeds. Test your modified programs to see what happens.

1. How fast can the robot go before there are problems?

   _____

2. What happens when the robot drives too fast?

_____

_____

_____

Name_____

Lesson 8: Worksheet 8.4 – Follow a line

In this activity, you will use the Edison robot's line tracker to write a program in EdPy which makes Edison follow any black line.

To do this, we first need to create an algorithm for following a black line.

What is an algorithm?

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 your friends have 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.

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

Algorithms in programming

In programming, we often have sets of problems we want to solve.

In this activity, for example, we want to be able to have Edison follow any black line. Our set of problems, therefore, is 'follow any black line'. Any specific line you make for Edison to follow is a new problem inside this set.

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.

The algorithm produces 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.

To solve our set of problems, we need the algorithm to produce a set of instructions that will work for any black line.

You can plan out an algorithm using pseudocode or a flowchart, just like you do when you plan out a program.

Name_____

Here is an algorithm in pseudocode that will allow Edison to follow any black line:

```
Turn the line tracker on
Loop forever
    If the robot detects a black line then
        Drive Forward Right
    Else
        Drive Forward Left
```

This algorithm says that if the line tracking sensor is on a non-reflective surface (black), then the Edison robot should drive forwards to the right, moving the sensor towards the white surface. If the line tracking sensor is not on a black surface, then the robot should drive forwards to the left, moving the sensor towards the black surface. In this way, the robot continuously moves forward and tracks the edge of the line.

Notice that no speeds or distances are mentioned in the pseudocode. Those sorts of details are usually left to the coding stage.

Your turn:

Translate the pseudocode algorithm into a Python program to make your Edison robot follow any black line. Experiment with different speeds and distances to achieve the smoothest line tracking. Download your code into Edison and test it using the track on activity sheet 8.2.

1. What was the best combination of speed and distance to achieve smooth line tracking you found?

_____

2. What did your Python program look like? Write your code here.

_____

_____

_____

_____

_____

_____

_____

_____

Try it!