EdPy lesson plans

Teaching guide and answer key

Contents

About the EdPy lesson plans

The EdPy lesson plans are a set of ten lessons designed to help you teach Python programming using the Edison robots and the online programming application, EdPy, available at www.edpyapp.com. The lessons here can be used as a first introduction to the Python programming language or as a way for students to experience a hands-on component to their growing Python knowledge.

While students of all ages thoroughly enjoy the programming process, problem solving, and collaboration involved in programming using Edison robots, a basic understanding of programming may be helpful before beginning these lesson plans. You may wish to use one of the other Edison programming languages, detailed at www.meetedison.com/robot-programming-software/ with your students before beginning EdPy.

This set of ten lessons begins with an introduction to the Edison robot and the EdPy programming environment. Students will then begin to write simple programs and build up to programs containing more complex Python programming structures. As some elements of the lessons are progressive, you may find it easiest to work through these lessons sequentially.

Each of the ten lessons contains multiple activities. On average, an activity can be completed in a 45-minute class. Depending on your students' ages, abilities and familiarity with Edison robots and Python, you may require additional time or find you need to make adjustments to the lessons. These lesson plans can also be used as a starting point to develop your own customised lessons and activities.

Using this guide

This guide provides teachers and instructors with:

- an overview of each of the ten EdPy lessons,
- a breakdown of each lesson,
- an overview of each student worksheet and activity sheet,
- the answer key to the student worksheets,
- a student worksheet tracking chart and completion certificate, and
- additional supporting information per lesson, as required.

This guide is designed to complement the student worksheets and activity sheets set, available at https://meetedison.com/robot-programming-software/edpy/ which comprise the bulk of the EdPy lessons.

Student worksheets

The student worksheets are designed to allow for independent learning, enabling students to transition through the lessons at their own pace.

Each of the ten EdPy lessons contains at least one student worksheet. Each worksheet contains:

- information about the lesson,
- instructions,
- links to any additional resources, and
- questions for students to answer to reinforce and demonstrate learning.

This guide contains an overview of each worksheet in lesson order. An answer key is also included. It is important to note that while some of the worksheet questions have set answers, many others do not. Instead, these questions allow for students to describe their own programs or experiences. In these instances, example answers are provided.

Activity sheets

Some of the lessons also include activity sheets, which provide students with a working area to quickly test their programs and run experiments.

Answer key

A complete answer key to the student worksheets, including suggestions for marking student work, is included.

Creative Commons licence attribution details

The EdPy lesson plans have been produced in conjunction with Dr Sarah Boyd from LoveSTEM.

Dr Sarah Boyd is the Director and founder of LoveSTEM, a company founded in 2016 to promote a love of STEM – Science, Technology, Engineering and Mathematics. Sarah is a registered teacher in New South Wales, Australia, and has over 20 years' experience as a software engineer. You can find Sarah on Twitter @sarahboydster and learn more about LoveSTEM at www.loveSTEM.com.au.

Lesson plan overview

This section provides a high-level overview of the scope and key learning outcomes of the ten EdPy lesson plans.

Lesson 1: Get familiar and set up

Technology skills – Students familiarise themselves with the programming environment and learn how to download a program to the robot. Students will:

- set up and become familiar with Edison using barcode programs,
- open the EdPy software application and become familiar with all the elements in the programming environment,
- review and understand all the elements in the setup code, and
- download a test program.

Lesson 1 contains four parts and has four worksheets:

- Part 1 – Meet Edison (Worksheet 1.1)
- Part 2 – Barcode programming (Worksheet 1.2)
- Part 3 – Meet the EdPy app (Worksheet 1.3)
- Part 4 – Downloading a test program (Worksheet 1.4)

Lesson 2: Robot movement – Driving

Introduction to sequential programming – Students learn how the robot responds to some basic driving commands and bring together the concepts of time, speed and distance. Lesson 2 introduces students to the concepts of:

- functions in Python (including input parameters),
- prewritten code,
- expressions in Python,
- the ‘while’ loop and indentation in Python, and
- command completion in EdPy.

Lesson 2 contains five worksheets and one activity sheet:

- Worksheet 2.1 – Drive the robot forward
- Worksheet 2.2 – Drive the robot backwards
- Worksheet 2.3 – Forwards, then backwards
- Worksheet 2.4 – Expressions in Python
- Worksheet 2.5 – Keypad activated driving

- Activity sheet 2.1 (Start/finish line)

Lesson 3: Robot movement – Turning

Sequential programming and basic geometry – Students explore additional driving commands which utilise time and geometry to enable greater variety and control in how they can drive the Edison robot. Lesson 3 introduces the concept of variables in Python and several new functions. Concepts from previous lessons which are reinforced in this lesson include:

- functions and input parameters,
- using the Documentation feature of EdPy, and
- using the 'while' loop with expressions.

Lesson 3 contains four worksheets and two activity sheets:

- Worksheet 3.1 – Turn right
- Worksheet 3.2 – Turn left 180°
- Worksheet 3.3 – Right turn, then left turn
- Worksheet 3.4 – Mini maze

- Activity sheet 3.1 (Turning)
- Activity sheet 3.2 (Mini maze)

Lesson 4: Get your robot into shape

Loops in Python – Students learn their second control structure in Python, the 'for' loop, and learn about the 'range()' function in Python. Lesson 4 has students practice writing programs using loops which allow them to drive their Edison robot in various shapes.

Lesson 4 contains four worksheets and four activity sheets:

- Worksheet 4.1 – Drive in a square
- Worksheet 4.2 – Use a loop to drive in a square
- Worksheet 4.3 – Drive in a triangle and a hexagon
- Worksheet 4.4 – Challenge! Drive in a circle

- Activity sheet 4.1 (square)
- Activity sheet 4.2 (triangle)
- Activity sheet 4.3 (hexagon)
- Activity sheet 4.4 (circle)

Lesson 5: Play sounds and dance

Sounds in Edison – Students learn about how sounds work in Edison. Lesson 5 introduces the concept of strings in Python and reinforces how to use expressions in programs.

Lesson 5 contains five worksheets:

- Worksheet 5.1 – Play tones
- Worksheet 5.2 – Make an alarm
- Worksheet 5.3 – Play a tune
- Worksheet 5.4 – Make your robot dance
- Worksheet 5.5 – Challenge! Dance to music

Lesson 6: Clap sensing

Introduction to inputs (sensors) – Students learn how to make the Edison robot respond to outside stimulus, using the sound-detecting sensor to register hand claps. Lesson 6 introduces the concept of flowcharts, which students practice reading and designing. Students also learn how to make their own function in Python.

Lesson 6 contains three worksheets:

- Worksheet 6.1 – Flash the LED in response to a clap
- Worksheet 6.2 – Drive in response to a clap
- Worksheet 6.3 – Design your own function

Lesson 7: Detect obstacle

Introduction to the concepts of obstacle detection and autonomous robotics – Students learn how to program the Edison robot using the infrared sensors, enabling the robot to make decisions autonomously in response to obstacles in the robot's environment. Students also learn about event based programming and how to use 'if' statements in Python.

Lesson 7 contains one activity sheet and five worksheets:

- Activity sheet 7.1 – Calibrate obstacle detection

- Worksheet 7.1 – Infrared obstacle detection
- Worksheet 7.2 – Detect an obstacle and stop
- Worksheet 7.3 – Obstacle avoidance
- Worksheet 7.4 – Detect an obstacle as an event
- Worksheet 7.5 – Right and left obstacle detection

Lesson 8: Line sensing and tracking

Industrial-like robotic behaviour – Students explore the Edison robot's line detecting sensor and learn about basic robot sensing and control similar to that used in advanced automated factories and warehouses. Students are also introduced to the concepts of pseudo code and algorithms.

Lesson 8 contains four worksheets and two activity sheets:

- Worksheet 8.1 – Line tracking sensor
- Worksheet 8.2 – Drive until a black line
- Worksheet 8.3 – Drive inside a border
- Worksheet 8.4 – Follow a line

- Activity sheet 8.1 (test space)
- Activity sheet 8.2 (border)

Lesson 9: Respond to light

Environmental measurement and programming mathematics – Students explore how the Edison robot's visible light sensors work, learning how the sensors can be used to measure light levels with the results being used as variables in a program. Students expand their knowledge of how to create programs which perform mathematics on input variables to control the robot's behaviour. They also practise tracing values through a program.

Lesson 9 contains three worksheets:

- Worksheet 9.1 – Light alarm
- Worksheet 9.2 – Automatic lights
- Worksheet 9.3 – Light following

Lesson 10: Design brief - A vampire robot

Creative thinking and problem solving – Students put the knowledge from all previous lessons into a practical application as they design their own Vampire robot behaviours. Students are first introduced to the concepts of a class definition and objects in Python. They then work through the stages of good program design, including using flowcharts and pseudocode, then code and test their programs. This lesson serves as a final project for the course, asking students to document and reflect on failures they experience as well as demonstrate their final product.

Lesson 10 contains one project-style worksheet:

- Worksheet 10.1 – Vampire robot

Lesson 1: Get familiar and set up

Technology skills – Students familiarise themselves with the programming environment and learn how to download a program to the robot.

There are four parts to lesson 1. During this lesson students will:

1. get to know Edison,
2. use barcodes to program Edison,
3. meet the EdPy app, and
4. check that everything is working by downloading a test program.

Before using Edison with your students, you will need to set up the computers you will be using with the EdPy app and get the Edison robots ready.

Setting up your computer to work with the EdPy app

Depending on the type of computer you are using, there are a few things you will need to do to prepare it to be able to work with the EdPy app.

If you are using computers running Windows operating systems, you will need to disable sound enhancements. Please go to https://meetedison.com/edison-robot-support/trouble-shooting/#soundenhancements to find step-by-step video guides showing you how to disable sound enhancements for standard Window's sound enhancements as well as the most common third-party software programs.

To be able to program Edison, most devices will need the volume turned up to maximum or 100%. As many devices have built-in safety settings that reduce the volume when an audio device is connected using the headphone jack, it is also important to double check that your volume is turned all the way up after plugging in the EdComm programming cable to your device.

Get Edison ready

To get Edison ready for use, you need to:

1. Open the battery compartment at the back of Edison and remove the EdComm programming cable.
2. Insert 4 ‘AAA’ batteries. Please refer to the picture to ensure that the batteries are inserted correctly. Be sure to reclose the battery case by clipping the battery cover back on.

Ensure the batteries are in the right way.

Please note: Low or flat batteries can cause a range of issues with Edison. For this reason, please be sure to use fresh, fully charged batteries.

3. To turn Edison on, flip the robot over. Slide the power switch to the ‘on’ position, as shown in the picture. This will turn Edison on and the red LED lights will start flashing.

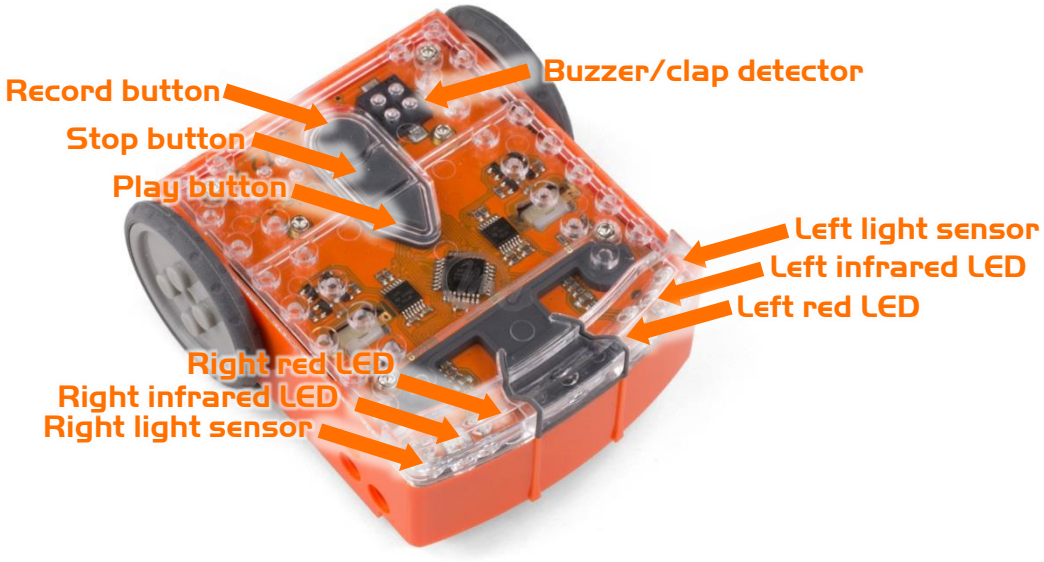
Push the switch towards the ‘on’ symbol.

Please note: While Edison will turn off automatically if not used after five minutes, we recommend you turn the robot off manually when not in use.

Lesson 1, part 1: Meet Edison (Worksheet 1.1)

To work with Edison and EdPy, students need to be familiar with their Edison robot. Worksheet 1.1 teaches students the location of all of Edison's sensors and the functions of the three buttons.

Have students review the images in Worksheet 1.1. You may want to have this worksheet available for students to review throughout the 10 lessons.



Students need to get to know Edison's sensors and buttons.

Default settings of Edison's three buttons:

- Record button – 1 press = download program; 3 presses = scan barcode
- Stop button – 1 press = stop program
- Play button – 1 press = run program

Students should also turn the Edison robot over to learn where the power switch and line tracking sensor are located on the bottom.

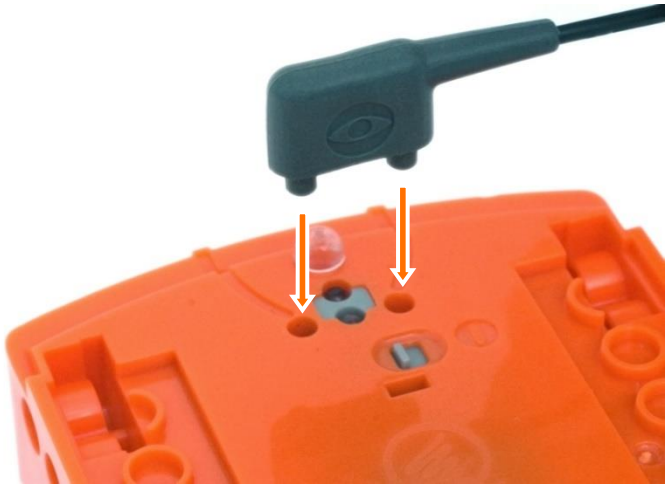
Location of Edison's power switch and line tracking sensor.

Edison's line tracking sensor is made up of two parts: a red LED light and a light sensor.

The line tracking sensor also reads special barcodes that activate pre-installed programs.

Students need to know how to connect Edison to a computer using the EdComm cable to be able to download programs. To connect Edison, plug the headphone jack into the headphone socket on your computer. The other end of the EdComm cable connects to your Edison robot as shown.

The EdComm programming cable.



Have students practice attaching the EdComm programming cable to Edison.

Lesson 1, part 2: Barcode programming (Worksheet 1.2)

Edison comes with pre-loaded programs which are activated by driving over special barcodes. Worksheet 1.2 includes four of these barcodes which Edison can scan. Using these barcodes is a good way to get students excited about robotics and what they will learn using Edison.

To scan a barcode, place Edison on the right of the barcode and press the record (round) button three times.

Have the students scan and then run each barcode one at a time. Explain what the students should do to activate a program once they have scanned the correct barcode. Describe how each program works and which sensors each program uses.

Clap controlled driving

What to do: Place Edison down on a flat surface and press the play (triangle) button. Place your hands close to Edison and clap one time. Edison will turn right. Next, try clapping your hands twice, causing Edison to drive forward about 30cm. Then try tapping Edison with your finger, first once and then twice, to see what happens.

What's happening: Edison has a sound sensor and uses it to respond to loud sounds, such as claps.

Avoid obstacles

What to do: Set up a few obstacles. The obstacles need to be opaque but not too dark (e.g. not black) and at least as tall as Edison for the robot to detect them. Press the play (triangle) button and watch Edison approach an obstacle and then turn away to avoid colliding with it.

What's happening: The avoid obstacles program uses the Edison robot's infrared (IR) light LEDs and IR sensor to detect objects directly in front of the robot. Once the pre-set program is activated, the Edison robot will drive forward, turning as needed to avoid obstacles it encounters.

Line tracking

What to do: Either use the thick, dark lines on an EdMat (free download available at https://meetedison.com/robot-activities/#edmatdownload) or make a track for Edison to follow by drawing a dark line approximately 1.5cm (0.6 inches) wide on a white background. Make sure students start by placing Edison next to the black line, not on top of it, so that the line tracking sensor is on white. Press the play (triangle) button and watch Edison follow the line.

What's happening: The line tracking program uses the Edison robot's reflected light sensor to detect differences between dark and light surfaces beneath the robot. Once the

pre-set program is activated, the Edison robot will drive until it finds a dark coloured line, then follow that line. In this program, Edison's line tracking sensor shines light on the surface and then measures the amount of light that is reflected back. White reflects a lot of light, giving a high light level reading while black reflects very little, giving a low light level reading. Edison adjusts direction according to these light level readings. When Edison is off the line, it turns right to get on the line. However, when Edison is on the line, it turns left to get off the line. This is why Edison 'waddles' back and forth at the edge of the line.

Follow torch

What to do: You will need a torch or flashlight and a flat surface located away from any other sources of bright light, such as sunlight or overhead fluorescents. Press the play button and aim the torch at Edison. Once Edison 'sees' the bright source of light, the robot will drive towards it. By moving the torch, you can control where Edison drives.

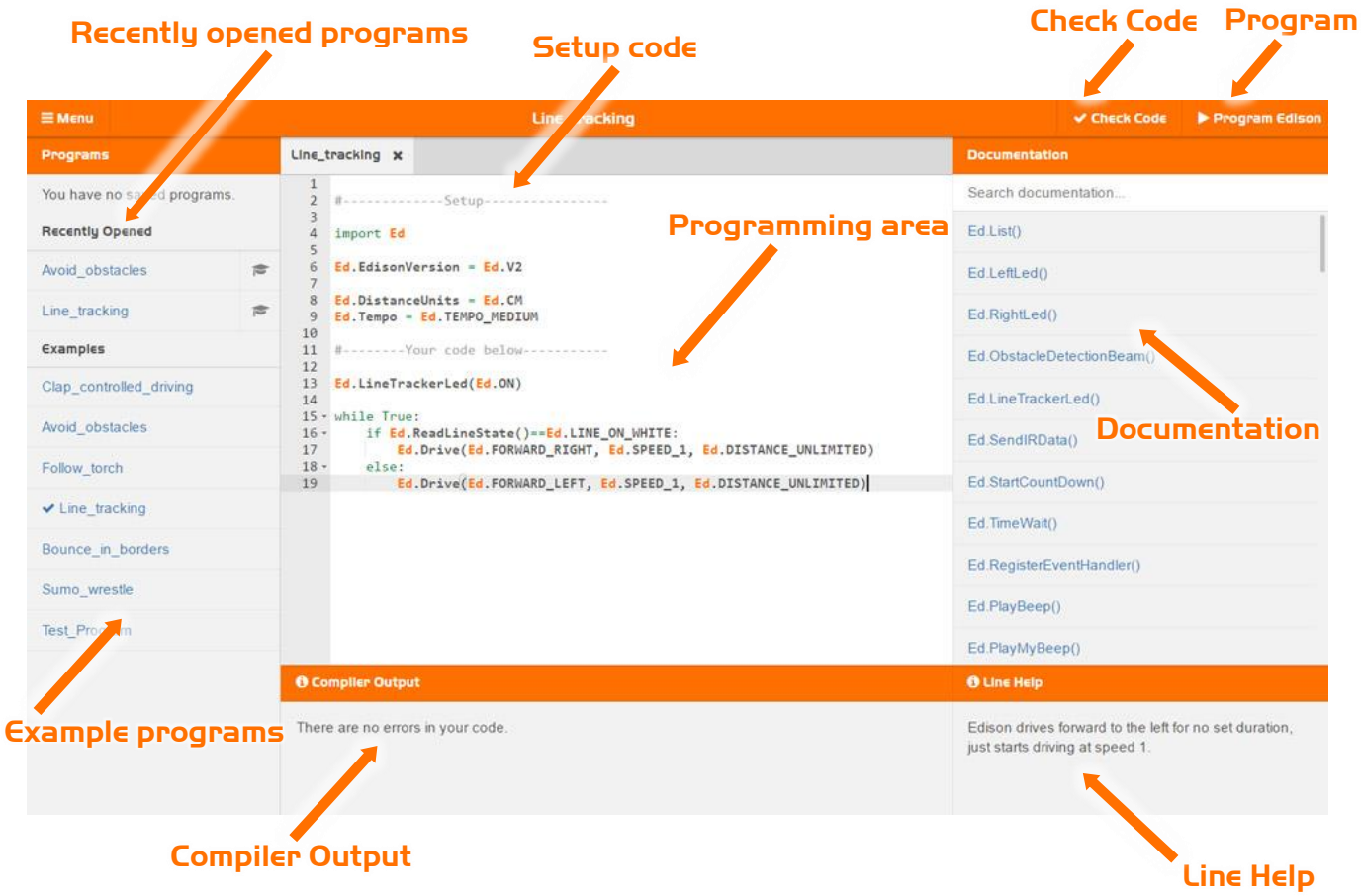
What's happening: Edison's two light sensors at the front left and right take light readings and the level of these reading are compared to each other. If the level of light on the right sensor is higher than the left sensor, then Edson's left motor is driven forward, turning Edison right, towards the light. This movement will continue until the level of light is greater on the left sensor. At that time, the left motor will stop, and the right motor will be driven forward, driving Edison, once again towards the light.

This is one of Edison's most interesting programs because it mimics the behaviour we see in some flying insects, such as moths swarming around a bright light at night. This type of behaviour is called 'phototropism' and is also found in plants that grow towards the sun.

As a lesson extension, ask your students '*Does this behaviour remind you of anything?'* and discuss phototropism and animalistic behaviours in robotics.

Lesson 1, part 3: Meet the EdPy app (Worksheet 1.3)

Having gained familiarity with Edison and seen what Edison can do by using the barcodes, students will be excited and motivated to start using the EdPy app. To access the EdPy app, open a browser and go to www.edpyapp.com. We recommend using Chrome for best results.



Worksheet 1.3 introduces students to what the EdPy app looks like and has them work through test programs to become familiar with the app. Encourage your students to play around with the EdPy app interface, opening example programs, searching through the documentation window and noticing how the line help text works.

You may also want to explain the various sections of the app to your students:

Programming area

This is where you type the Python code to control the Edison robot.

Documentation

In this window, you can search the documentation about Python functions that can be used in EdPy. Here you can find explanations of all the available commands including example code.

Recently Opened

This window contains a list of recently opened programs. Clicking on these will re-open the program in the programming area.

Examples

This window contains a list of example programs which you can select and open in the programming area. The example programs explain how to use common functions of the Edison robot, including some to the functions students encountered with the barcode programs.

Check Code

When the 'Check Code' button is clicked, the code in the programming area is checked for errors. If errors are detected, a message will appear in the 'Compiler Output' window at the bottom of the screen providing details on the errors.

Program

When the 'Program' button is clicked, the current program is downloaded to the Edison robot. Make sure the Edison robot is ready for the code by connecting the cable, turning the volume up to full and pressing Edison's round button one time.

Compiler Output

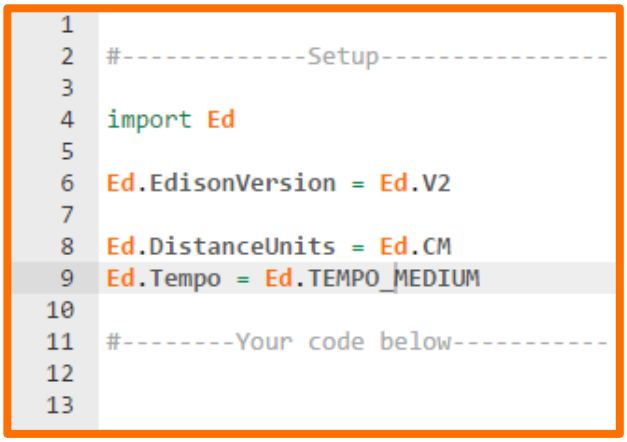
When EdPy translates the written code into commands readable by the Edison robot, this is called 'compiling the code.' If errors are detected when you click 'Check Code,' they will be displayed in the compiler output section. If there are no errors detected, this section will display a message which reads: 'There are no errors in your code.'

Line Help

The line help is a guide to use while programming which shows users what each of the Python commands do. When you select a line of code in the programming area with your mouse cursor, the 'line help' window will display a plain-English description of the command being highlighted by your cursor. This description is a 'translation' of the Python code into English.

Setup code

All Edison programs must contain the setup code that you see every time you open the EdPy app. This image is the setup code for a V2.0 Edison robot:



Review and discuss the following points about the setup code with your students:

- Line 2 of the setup code starts with a '#' (hash) character. When a line starts with this character, it is called a 'comment line.' Any characters that come after the '#' are ignored by the compiler. The text following after the # is used to document code so that other people can understand the program. Line 11 contains another comment line which will also be ignored by the EdPy compiler.
- Line 4 contains the 'import' command. This tells Python to 'import' another library of pre-written Python code. In line 4 of the setup, we are importing all the built-in Edison Python commands, in other words, all the EdPy commands. These built-in commands will all start with the prefix 'Ed.' and allow us to program the Edison robot in Python.
- Line 6 defines the built-in Edison variable that relates to the version of Edison you are using. Variables in Python are reserved memory locations for storing values.

  If you are using V2.0 version of the Edison, this line should be:

  ```
  Ed.EdisonVersion = Ed.V2
  ```

  If you are using version 1 of the Edison, you should change this line to:

  ```
  Ed.EdisonVersion = Ed.V1
  ```

  Please note: selecting the correct version of Edison when you launch EdPy will automatically set these values.

- Look at the character sets in the sample code that are written in all capitals, such as V1, V2, CM and INCH. In Python, the convention is that any variable name written in all caps is a 'constant'. Constants are variables that maintain the same value everywhere in your program and are used as a reference point throughout the program. EdPy has a range of helpful built-in constants.
- Line 8 defines the built-in Edison variable that relates to the distance units used by the robot: inches, cm or time (in seconds).

  If you want to use inches, you should change this line to:

  ```
  Ed.DistanceUnits = Ed.INCH
  ```

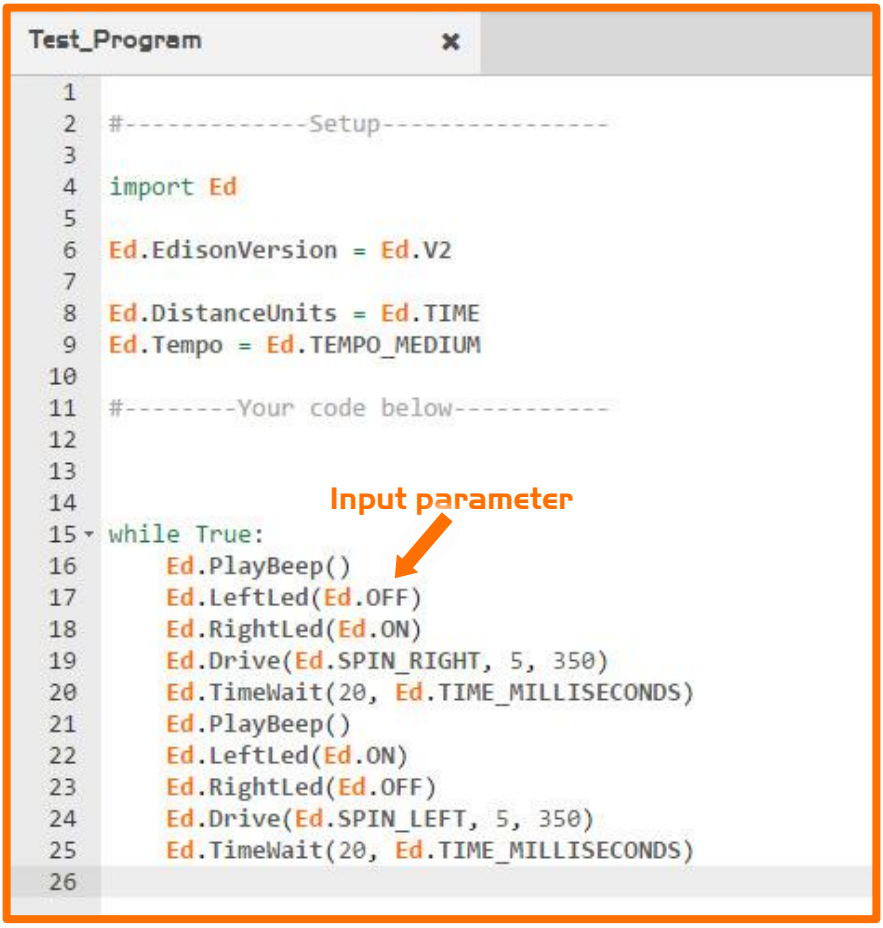
  If you are using an Edison V1, then you **must** select Ed.TIME for the distance units:

  ```
  Ed.DistanceUnits = Ed.TIME
  ```

- Line 9 defines the built-in Edison variable which relates to how fast or slow Edison plays a musical tune. See the documentation section for the different speeds that can be set.

Lesson 1, part 4: Download a test program (Worksheet 1.4)

Once the students are familiar with the EdPy app, they should experiment with a program. Have the students open the test program by selecting 'Test_Program' from the 'Examples' window on the left. The test program looks like this:




```
1
2  #-------------Setup----------------
3
4  import Ed
5
6  Ed.EdisonVersion = Ed.V2
7
8  Ed.DistanceUnits = Ed.TIME
9  Ed.Tempo = Ed.TEMPO_MEDIUM
10
11 #--------Your code below-----------
12
13
14
15 while True:
16     Ed.PlayBeep()
17     Ed.LeftLed(Ed.OFF)
18     Ed.RightLed(Ed.ON)
19     Ed.Drive(Ed.SPIN_RIGHT, 5, 350)
20     Ed.TimeWait(20, Ed.TIME_MILLISECONDS)
21     Ed.PlayBeep()
22     Ed.LeftLed(Ed.ON)
23     Ed.RightLed(Ed.OFF)
24     Ed.Drive(Ed.SPIN_LEFT, 5, 350)
25     Ed.TimeWait(20, Ed.TIME_MILLISECONDS)
26
```

Explore the 'Test_Program'

Explain to the students that this is an example of what a program in Python looks like. Use the Test program to discuss some of the basic points they will need to know to use EdPy:

- In EdPy, Edison looks at each line of code one at a time and does what the line says. Blank lines and comment lines are ignored.
    - Remind students they can learn more about what a line does by selecting it and reading the Line Help window text.
- Each line in this program is calling on the built-in Edison commands.
    - Remind students that they can explore each of the commands by searching for that command in the Documentation window.
- In Python, commands may have parameters that are passed into them to specify inputs for the command to use. Look at some of the commands in the program and examine the associated parameters.

- Example 1: the Ed.RightLed() command takes one input which tells the right LED whether it should turn on (Ed.ON) or off (Ed.OFF).
- Example 2: The Ed.PlayBeep() command has no input parameters.
- Example 3: In the Ed.TimeWait() command there are 2 input parameters – the first for the number of seconds or milliseconds to wait, and the second parameter for the unit of time (milliseconds or seconds).

Once you have looked through the Test program, have the students download the program to their robots.

Download the program

Explain to the students that the program goes from the computer through the EdComm cable to the Edison robot. The EdComm cable converts the electrical signal of sounds from the headphone jack into light, which the robot receives, converts to an electrical signal and stores as a program in its memory.

To download a program to Edison, connect the EdComm cable to the headphone socket on the computer and **turn up the volume to full**. As some devices have safety features which turn the volume down when a listening device is detected, instruct your students to always confirm the volume is at full **after** they have plugged in the EdComm cable.

Plug the other end of the EdComm cable into Edison as shown.

To download the test program, follow these steps:

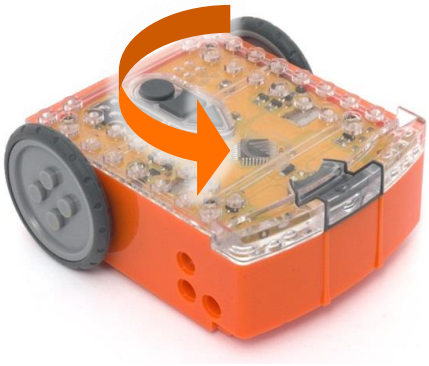
1. Turn Edison on, then press Edison's record (round) button once
2. Connect Edison to the computer using the EdComm cable and confirm the volume is turned up to full
3. Press the 'Program Edison' button in the upper right corner of the EdPy app
4. Follow the steps in the pop-up window, then press 'Program Edison'

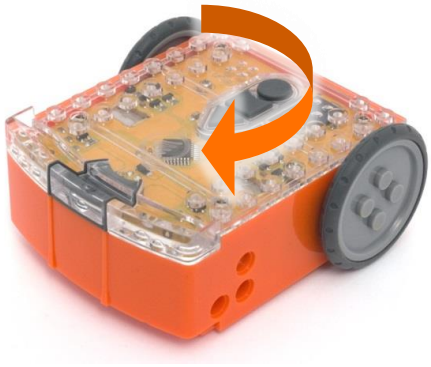
While the program is downloading to Edison, a sound like an old dial-up modem can be heard. When many users in a single location, like in a classroom, are downloading

programs at the same time, you may experience slower internet speeds. This can cause the program to take longer to generate the ‘program Edison’ pop-up box and for the program to download to Edison. With a very slow connection, you may need to try again. Press the stop (square) button on Edison, then press the record (round) button one time. Restart the download by clicking on the ‘Program Edison’ button in the top-right corner of the app.

Once the program has downloaded correctly and made the ‘success’ sound, you can unplug the EdComm cable.

Have students press the play (triangle) button on Edison one time. This will cause Edison to run the test program, making Edison spin right and left while flashing its lights and beeping.





Remind the students that the robot reads each line of the program line by line. Look at the program again, watching Edison play the program. Discuss where you can see Edison ‘play’ each line.

Lesson 2: Robot movement – driving

Introduction to sequential programming – Students learn how the Edison robot responds to Python commands in sequence order and bring together the concepts of time, speed and distance.

In this lesson, students use the built-in drive functions in the EdPy app to drive the robot forward and backwards set distances.

Students work through four main programming tasks in lesson 2, using five worksheets and one activity sheet:

- Worksheet 2.1 – Drive the robot forward
- Worksheet 2.2 – Drive the robot backwards
- Worksheet 2.3 – Forwards, then backwards
- Worksheet 2.4 – Expressions in Python
- Worksheet 2.5 – Keypad activated driving

- Activity sheet 2.1 (Start/finish line)

Lesson 2, program 1 – Drive the robot forward (Worksheet 2.1)

Students write a program that drives the robot forward a set distance from a 'start' point to a 'stop' point.

Worksheet 2.1 walks students through writing the program, exposing them to functions, input parameters and some of the help features of EdPy.

Tips and tricks:

- Instruct students to either use the start and finish lines on activity sheet 2.1 or to create start and stop markers using coloured tape on a desk or the floor.
- Student programs must have Edison stop driving before crossing over the stop/finish marker.

Lesson 2, program 2 – Drive the robot backwards (Worksheet 2.2)

Students are introduced to the idea that there are multiple ways to write a program which will give the same result. Students explore two different ways to write a program that will drive the robot backwards a set distance from a 'start' point to a 'stop' point.

Worksheet 2.2 walks students through writing a program, reinforcing the idea of using functions, input parameters and some of the help features of EdPy. Students then experiment with using a constant as an input parameter.

Tips and tricks:

- Instruct students to either use the start and finish lines on activity sheet 2.1 or to create start and stop markers using coloured tape on a desk or the floor.
- Student programs must have Edison stop driving before crossing over the stop/finish marker.

Lesson 2, program 3 – Forward, then backwards (Worksheet 2.3)

Students work more with sequential programming, creating a program with multiple steps on multiple lines of code. Students write a program that will drive the robot forward, then backwards a set distance from a start marker to a stop marker, and then back again.

Worksheet 2.3 walks students through writing the program, reinforcing the basic programming steps for using Edison and EdPy. Students practice using a constant as an input parameter.

Tips and tricks:

- Instruct students to either use the start and finish lines on activity sheet 2.1 or to create start and stop markers using coloured tape on a desk or the floor.
- Student programs must have Edison stop driving before crossing over the stop/finish marker.

Lesson 2, offline activity – Expressions in Python (Worksheet 2.4)

Students are introduced to:

- the core concept of expressions,
- how expressions are written in Python,
- how expressions operate, and
- why expressions are used in programming.

Worksheet 2.4 explains how expressions work, are written and resolved in Python. The students practice reading expressions for meaning and resolving expressions to either ‘true’ or ‘false.’

Lesson 2, program 4 – Keypad activated driving (Worksheet 2.5)

Students are introduced to:

- the while loop in Python,
- the importance of indentation in Python, and
- the Ed.ReadKeyPad() function.

Worksheet 2.5 demonstrates how expressions are used inside a ‘while’ loop in EdPy. The students program their Edison robot to drive forward when the triangle or round button is pressed by using a ‘while’ loop and expressions. They will then write their own version of the program, adapting it so that Edison will drive forward when the triangle or round button is pressed, then drive backwards when the triangle or round button is pressed again.

Tips and tricks:

- Python is very particular about white space (e.g. blank characters) inside code lines. In Python syntax, using the tab key to indent a line of code and using the space key to indent it the same amount are not equivalent. Students should use the tab key.

Lesson 3: Robot movement – turning

Sequential programming and basic geometry – Students explore additional driving commands, learning how the robot responds to time and geometry.

In this lesson, students use a range of drive functions to turn the robot at different angles (90°, 180° and 270°) and then combine multiple functions to create a driving sequence.

Students work through four main programming tasks in lesson 3, using four worksheets and two activity sheets:

- Worksheet 3.1 – Turn right
- Worksheet 3.2 – Turn left 180°
- Worksheet 3.3 – Right turn, then left turn
- Worksheet 3.4 – Mini maze

- Activity sheet 3.1 (Turning)
- Activity sheet 3.2 (Mini maze)

Lesson 3, program 1 – Turn right (Worksheet 3.1)

Students are introduced to variables in Python and learn how to use variables to make Edison turn a set number of degrees.

Worksheet 3.1 explains what variables are, why they are useful and how to write them in Python. The students work through a program which uses a variable to turn the robot right 90 degrees. They then expand their program to use the variable a second time, turning the robot back. Students also practice assigning values to the variable and explore variable naming conventions in Python.

Tips and tricks:

- Instruct students to either use activity sheet 3.1 or to create start and stop angle markers using coloured tape on a desk or the floor.

Lesson 3, program 2 – Turn left 180° (Worksheet 3.2)

Students practice controlled driving by creating two different programs to turn their Edison left exactly 180 degrees.

Worksheet 3.2 has students write a program making their Edison turn left 180 degrees two different ways. The first programming task reinforces using variables in a program. The second programming task also challenges students to measure and calculate what variable they need to use.

Tips and tricks:

- Instruct students to either use activity sheet 3.1 or to create start and stop angle markers using coloured tape on a desk or the floor.

- Due to minor mechanical differences in the motors and encoders inside different Edison robots, some robots may not turn to exactly 180 degrees when given the input of 180. Encourage students to try different values around 180 (e.g. 178 or 183) to find the input that works best for their Edison.
- If students are using either CM or INCH, encourage students to calculate the variable value needed when using the Ed.DriveRightMotor() command. This can be done by measuring the width of Edison's wheelbase and using that as the radius of the half circle they need to drive.

Lesson 3, program 3 – Right turn, then left turn (Worksheet 3.3)

Students create a program using multiple elements they have learned previously; including the 'while' loop, expressions and variables; to get their robot to turn first right following one key press, then left following a second key press.

Worksheet 3.3 walks students through a program using the Ed.ReadKeyPad() command and a while loop. Students then write an expanded program containing a second while loop, so that their robot will turn right exactly 90° when the triangle button is pressed once, then turn left exactly 270° when the triangle button is pressed a second time.

Tips and tricks:

- Instruct students to either use activity sheet 3.1 or to create start and stop angle markers using coloured tape on a desk or the floor.
- Due to minor mechanical differences in the motors and encoders inside different Edison robots, some robots may not turn to exactly the degrees entered. Encourage students to try different values around the degree value to find the input that works best for their Edison.
- Remind students to put Ed.ReadKeyPad() into the line above the 'while' loop to clear any previous key presses before the loop.
- Remind students that they can do maths to expressions, for example, 3*degreesToTurn.

Lesson 3, program 4 – Mini maze (Worksheet 3.4)

Students create a program using multiple functions applying the concepts they have learned in the last two lessons to get their robot to drive through a maze.

Worksheet 3.4 challenges students to write and test their own program that will allow Edison to navigate a maze. The worksheet includes two optional challenges: 1) to race another student through the maze and 2) to design their own, more challenging maze for either themselves or a partner.

Tips and tricks:

- Instruct students to either use activity sheet 3.2 or to create a maze using coloured tape on a desk or the floor.
- Remind students that due to minor mechanical differences in the motors and encoders inside different Edison robots, some robots may not turn to exactly the degrees entered. Encourage students to experiment and try different values around the degree value to find the input that works best for their Edison.
- Challenge 1 requires students mainly to adjust their program's time inputs. Remind students of the parameters for success to avoid 'cheating'. Students should use the same maze as each other for this challenge.
- Challenge 2 asks students to design their own, more complicated maze and write a program for Edison to navigate that maze. Alternatively, have students design new mazes and exchange them with each other, writing programs to complete their partner's maze.

Mini maze program:

Key code elements:

Ed.Drive(), Ed.SPIN_RIGHT, Ed.SPIN_LEFT, Ed.FORWARD, (variable)

Example code:

```
degreesToTurn = 90

Ed.Drive(Ed.FORWARD, Ed.SPEED_5, 13)

Ed.Drive(Ed.SPIN_LEFT, Ed.SPEED_5, degreesToTurn)

Ed.Drive(Ed.FORWARD, Ed.SPEED_5,3)

Ed.Drive(Ed.SPIN_RIGHT, Ed.SPEED_5, degreesToTurn)

Ed.Drive(Ed.FORWARD, Ed.SPEED_5, 15)
```

Lesson 4: Get your robot into shape

Loops in Python – Students learn their second control structure in Python, the 'for' loop, and learn about the 'range()' function in Python.

In this lesson, students practice writing programs using loops as well as the other programming structures they have learned in previous lessons, allowing them to drive their Edison robot in various shapes (a square, a triangle, a hexagon, and a circle).

Students work through four main programming tasks in lesson 4, using four worksheets and four activity sheets:

- Worksheet 4.1 – Drive in a square
- Worksheet 4.2 – Use a loop to drive in a square
- Worksheet 4.3 – Drive in a triangle and a hexagon
- Worksheet 4.4 – Challenge! Drive in a circle

- Activity sheet 4.1 (square)
- Activity sheet 4.2 (triangle)
- Activity sheet 4.3 (hexagon)
- Activity sheet 4.4 (circle)

Lesson 4, program 1 – Drive in a square (Worksheet 4.1)

Students write a program to make their robot drive in a square, drawing on the skills from previous lessons.

Worksheet 4.1 instructs students to write a program which will make their Edison robot drive in a square, reinforcing the learning they have completed in lessons 2 and 3.

Tips and tricks:

- Instruct students to either use activity sheet 4.1 or to create a larger square using coloured tape on a desk or the floor.

Lesson 4, program 2 – Use a loop to drive in a square (Worksheet 4.2)

Students learn about the 'for' loop and 'range()' function in Python.

Worksheet 4.2 introduces students to both the 'for' loop and 'range()' function in Python. Students then use both the 'for' loop and 'range()' function to write a new program which will make their Edison robot drive in a square to see how using these commands allows them to write the program more efficiently, using less code.

Tips and tricks:

- Instruct students to either use activity sheet 4.1 or to create a larger square using coloured tape on a desk or the floor.

Lesson 4, program 3 – Drive in a triangle and a hexagon (Worksheet 4.3)

Students practice using the 'for' loop and 'range()' function in Python.

Worksheet 4.3 has students write two new programs using the 'for' loop and 'range()' function in order to drive in the shape of a triangle and then in the shape of a hexagon. Students are asked to think about the relationship between the shape they are driving and the parameter they are using in range() to determine the value set.

Tips and tricks:

- Instruct students to either use activity sheets 4.2 and 4.3 or to create larger versions of the shapes using coloured tape on a desk or the floor.
- Remind students that the sum of the interior angles of a triangle is 180° and that the sum of the interior angles of a hexagon is 720°.

Lesson 4, program 2 – Challenge! Drive in a circle (Worksheet 4.4)

Students are challenged to write a program to make their robot drive in a circle.

Worksheet 4.4 challenges students to program their Edison robot to drive in a circle. Students must draw on their knowledge of the 'for' loop and 'range()' function and their understanding of the relationship between the shape they are driving and the parameter they are using in range() to determine the value set in order to create a circle.

Tips and tricks:

- Instruct students to either use activity sheet 4.4 or to use a larger existing circle, such as the base of a rubbish bin.
- It may not be possible to drive in a perfect circle, but hint to students that a shape with thousands of very small sides can closely approximate a circle.
- Discuss with your students the logic behind the following program being able to make a circle, then test to see if it does:

```
degreesToTurn = 1
for x in range(360):
    Ed.Drive(Ed.FORWARD, Ed.SPEED_5, 1)
    Ed.Drive(Ed.SPIN_LEFT, Ed.SPEED_5, degreesToTurn)
```

Lesson 5: Play sounds and dance

Sounds in Edison – Students learn about how sounds work in Edison.

In this lesson, students are introduced to the concept of strings in Python and apply this using tune strings in EdPy. Students also practice writing programs which use expressions, resulting in Boolean datasets (true/false) as well as using the 'for' loop.

Students work through five main programming tasks in lesson 5, using five worksheets:

- Worksheet 5.1 – Play tones
- Worksheet 5.2 – Make an alarm
- Worksheet 5.3 – Play a tune
- Worksheet 5.4 – Make your robot dance
- Worksheet 5.5 – Challenge! Dance to music

Lesson 5, program 1 – Play tones (Worksheet 5.1)

Students learn about playing musical tones using Edison.

Worksheet 5.1 explains how to use EdPy and Edison to play individual musical notes. Students then write three short programs to practice playing a musical note and to learn how to get sounds to play in the background of the rest of the program, or to make the program wait until the notes have finished playing to continue.

Tips and tricks:

- Review the basic notations used in expressions in Python with students.
- Review how 'while' loops work in Python with students.

Lesson 5, program 2 – Make an alarm (Worksheet 5.2)

Students learn more about the PlayTones() function including how to customise the frequency of notes using variables.

Worksheet 5.2 first walks students through some basics of acoustics that will help them in creating their own sounds using Edison and EdPy. The first part of the worksheet explains:

- Frequency in acoustics and the base unit for frequency, hertz
- The relationship between frequency and period
- How to convert frequency to period for use in EdPy

Students then program their Edison robots to play an alarm program using a 'for' loop with a range. They will use this program to practice tracing code manually to verify their understanding of what the program is doing.

Tips and tricks:

- Students may benefit from a breakout lesson on the science of acoustics, including how frequency works, is measured and is related to pitch.
- Review indentation in Python and look at how indentation works in nested loops.

Lesson 5, program 3 – Play a tune (Worksheet 5.3)

Students learn how to use the Ed.PlayTune() and Ed.TuneString() functions to play a known tune using a string.

Worksheet 5.3 introduces strings as inputs and walks students through using a tune string in EdPy. Students program a simple song, experiment with adjusting the tempo constant in the Setup code and try modifying the base program to create a truncated version of the tune.

Tips and tricks:

- Students may benefit from a breakout lesson on reading sheet music.
- Remind students to use the autocomplete feature in the EdPy app to help them as they program. Explain that autocomplete is a feature used in some software designed to help the user complete each line of code by using built-in documentation.

Lesson 5, program 4 – Make your robot dance (Worksheet 5.4)

Students practice using a 'for' loop with variables and different drive functions they already know to make their Edison dance.

Worksheet 5.4 has students program their robot to do a version of the 'shimmy' dance routine using a 'for' loop and variables. Students are reminded about comment code and how to perform maths in their code.

Tips and tricks:

- Remind students that they can use the '#' character to write comment code in their programs.
- Review 'for' loops, proper indentation and using variables.

Lesson 5, program 5– Challenge! Dance to music (Worksheet 5.5)

Students design their own dance routine for their Edison robot, synchronising it with some tones or tunes of their own design.

Worksheet 5.5 has students program their robot so that Edison dances and plays tones at the same time. Students are then challenged to create their own dance and music combination, getting Edison to move with the music if they can.

Tips and tricks:

- You may wish to hand out basic sheet music to help students with the creation of their songs.

Lesson 6: Clap sensing

Introduction to inputs (sensors) – Students learn how to make the Edison robot respond to outside stimulus.

In this lesson, students learn about sensors in robots and how a robot can detect outside stimulus. Using Edison's sound-detecting sensor to register loud noises, such as hand claps, students create programs to determine the robot's response to the outside stimulus. Lesson 6 also introduces the concept of flowcharts, which students practice reading and designing, and teaches students how to make their own function in Python.

Students work through three main programming tasks in lesson 6, using three worksheets:

- Worksheet 6.1 – Flash the LED in response to a clap
- Worksheet 6.2 – Drive in response to a clap
- Worksheet 6.3 – Design your own function

Lesson 6, program 1 – Flash the LED in response to a clap (Worksheet 6.1)

Students are introduced to the concept of using data gathered from Edison's sensors in their program as a way to influence the robot's behaviour.

Worksheet 6.1 introduces the concept of flowcharts to plan and organise code. Students see how they can use flowcharts to describe their programs diagrammatically. The concept of an infinite loop is introduced. Students learn how to write an infinite loop using the hard-coded condition True. Students then experiment with the program, testing the sensors' capabilities and testing the functionality of different elements of the code in their program.

Tips and tricks:

- Review the basic notations used in expressions in Python with students.
- Review how 'while' loops work in Python with students.
- Remind students that they can tap their finger on their Edison instead of clapping. This can be helpful if many students are running the program near one another.

Lesson 6, program 2 – Drive in response to a clap (Worksheet 6.2)

Students incorporate learning from previous lessons to drive their Edison robot in response to an event (clap) and design their own flowchart to describe their code.

Worksheet 6.2 has students use the 'Ed.ReadClapSensor()' function to write two programs: first a program getting Edison to drive forward when a clap is detected and then a second program where Edison drives forward, waits for a clap, then drives backwards. Students also practice creating a flowchart.

Tips and tricks:

- Remind students the importance of clearing stored data when using a ‘read’ function in a loop.
- Review the basic flowchart shapes.

Lesson 6, program 3 – Design your own function (Worksheet 6.3)

Students learn more about functions in Python, including how to make their own function.

Worksheet 6.3 explores what functions are, how they work and how they are written in Python. Students learn how to structure a function and practice writing and exercising functions in their program. Students learn about organising their code to keep things neat and easily readable. They then design their own function, working through designing, coding and testing the function in their own program.

Tips and tricks:

- Remind students the importance of clearing stored data when using a ‘read’ function in a loop.
- Review the basic flowchart shapes.

Lesson 7: Detect obstacles

Introduction to the concepts of obstacle detection and autonomous robotics – Students use Edison's infrared sensors to program the robot to make decisions in response to obstacles in the robot's environment.

In this lesson, students learn more about Edison's infrared sensors and how these sensors can be used to enable the robot to make decisions in response to obstacles in Edison's environment. This serves as an introduction to the concept of autonomous robotics. Lesson 7 also exposes students to event based programming and how to use 'if' and 'if/else' statements in Python.

Students work through four main programming tasks in lesson 7 using one activity sheet and five worksheets:

- Activity sheet 7.1 – Calibrate obstacle detection

- Worksheet 7.1 – Infrared obstacle detection
- Worksheet 7.2 – Detect an obstacle and stop
- Worksheet 7.3 – Obstacle avoidance
- Worksheet 7.4 – Detect an obstacle as an event
- Worksheet 7.5 – Right and left obstacle detection

Lesson 7, offline activity – Infrared obstacle detection (Worksheet 7.1)

Students become familiar with the Edison robot's infrared obstacle detection technology and learn how the robot can detect obstacles in its path.

Worksheet 7.1 introduces students to IR and Edison's IR obstacle detection system. Students complete a short offline activity to reinforce their understanding of how Edison emits IR light and detects reflected IR light.

Tips and tricks:

- It may be beneficial to have a breakout session on the electromagnetic spectrum and where IR and visible light fall within the spectrum.
- Using different colours to mark their answers of IR light and reflected IR light on the worksheet will make it easier for students to demonstrate their understanding of the difference.

Lesson 7, program 1 – Detect an obstacle and stop (Worksheet 7.2)

Students write a program that drives Edison forward until an obstacle is detected in the robot's path. The robot then stops, avoiding a collision.

Worksheet 7.2 walks students through the fundamentals of programming Edison to use the robot's in-built obstacle detection technology. Students are provided with some basic

information about what is required of their EdPy program (such as always turning the obstacle detection beam on) as well as what they need to prepare offline regarding obstacles and Edison. Students then run a basic obstacle detection program.

Tips and tricks:

- To ensure the best results using obstacle detection, have students calibrate their Edison using activity sheet 7.1 before beginning this activity. This is especially important if the obstacle detection is too sensitive or not sensitive enough.
- Remind students that for Edison to be able to detect an obstacle, the obstacles need to be opaque but not too dark (e.g. not black) and at least as tall as Edison.

Lesson 7, program 2 – Obstacle avoidance (Worksheet 7.3)

Students write a program that drives Edison forward until an obstacle is detected in the robot's path. The robot then turns away, avoiding a collision.

Worksheet 7.3 has students program Edison with an obstacle detection program, building on what they have previously done. Students are then challenged to think about how they can further change and improve the program, bringing in knowledge from previous lessons. Students are also introduced to the concept of logical errors in programming and reminded about syntax errors. They then complete a challenge to identify both syntax and logical errors in an example program.

Tips and tricks:

- To ensure the best results using obstacle detection, have students calibrate their Edison using activity sheet 7.1 before beginning this activity. This is especially important if the obstacle detection is too sensitive or not sensitive enough.
- Remind students that for Edison to be able to detect an obstacle, the obstacles need to be opaque but not too dark (e.g. not black) and at least as tall as Edison.
- Remind students they must include 'Ed.ObstacleDetectionBeam(Ed.ON)' in any program running obstacle detection.

Lesson 7, program 3 – Detect an obstacle as an event (Worksheet 7.4)

Students use event handling to write an event driven program which makes Edison drive forward continuously, including an interrupt causing the robot to turn around whenever an obstacle is detected.

Worksheet 7.4 introduces students to new programming structures including 'events' and 'interrupts'. Students learn how event driven programming works and how to write event driven programs using event handlers. They then apply this knowledge by writing and modifying an event driven program for Edison using obstacle detection.

Tips and tricks:

- To ensure the best results using obstacle detection, have students calibrate their Edison using activity sheet 7.1 before beginning this activity. This is especially important if the obstacle detection is too sensitive or not sensitive enough.
- Remind students that for Edison to be able to detect an obstacle, the obstacles need to be opaque but not too dark (e.g. not black) and at least as tall as Edison.
- Remind students they must include 'Ed.ObstacleDetectionBeam(Ed.ON)' in any program running obstacle detection.

Lesson 7, program 4 – Right and left obstacle detection (Worksheet 7.5)

Students learn about 'if statements' including elif (Python syntax for 'else if') and else structures which they use to write three programs, including a program that can respond to whether there is an obstacle on the left or right of Edison.

Worksheet 7.5 has students work through three programs to progressively introduce the concept of 'if statements' and explain the importance of decision-making in coding. Students first work with a simple program using a basic if statement which has the Edison robot beep if an obstacle is detected. The concept of autonomous robotics is presented as the robot is effectively making decisions without human guidance and is, therefore, acting as an autonomous robot.

Students then progress to if/else statements working with another short program before progressing to the if/elif/else Python syntax structure. The third program uses this structure to create a program which reacts differently depending on where Edison detects an obstacle.

Tips and tricks:

- To ensure the best results using obstacle detection, have students calibrate their Edison using activity sheet 7.1 before beginning this activity. This is especially important if the obstacle detection is too sensitive or not sensitive enough.
- Remind students that for Edison to be able to detect an obstacle, the obstacles need to be opaque but not too dark (e.g. not black) and at least as tall as Edison.
- Remind students they must include 'Ed.ObstacleDetectionBeam(Ed.ON)' in any program running obstacle detection.
- This lesson can be used to stimulate discussion about autonomous robotics, artificial intelligence and real-world applications such as robotic cars that have no driver but use sensors to avoid collisions with people, buildings and other cars.

Lesson 8: Line sensing and tracking

Industrial-like robotic behaviour – Students explore the Edison robot's line detecting sensor and learn about basic robot sensing and control similar to that used in advanced automated factories and warehouses.

In this lesson, students learn more about how the Edison robot's line detecting sensor works. They apply knowledge from previous lessons, such as if statements, to programs, exploring how they can use the line tracking sensor to perform a range of tasks. They also learn about pseudo code and algorithms.

Students work through three main programming tasks in lesson 8 using four worksheets and two activity sheets:

- Worksheet 8.1 – Line tracking sensor
- Worksheet 8.2 – Drive until a black line
- Worksheet 8.3 – Drive inside a border
- Worksheet 8.4 – Follow a line

- Activity sheet 8.1 (test space)
- Activity sheet 8.2 (border)

Lesson 8, offline activity – Line tracking sensor (Worksheet 8.1)

Students become familiar with the Edison robot's line tracking sensor technology and learn how the robot can detect whether its driving surface is reflective or non-reflective.

Worksheet 8.1 breaks down how the Edison robot's line tracking sensor works. Students then use their Edison to determine if white or black surfaces are more reflective. Building on this, students think through and then test which other colours will be seen as reflective or non-reflective by the robot. The worksheet also includes a link to the 15-minute-long video *Humans Need Not Apply* along with questions designed to get the students to think about how robots may affect their own lives today and in the future.

Tips and tricks:

- It may be beneficial to have a breakout session on how light reflection and absorption works, including coloured light.
- Have students use activity sheet 8.1. Alternatively, students can use coloured paper.
- You may want to use the *Humans Need Not Apply* video as a partner, group or class project to stimulate discussion or coordinate a debate. Alternatively, this video can be used as a homework assignment, requiring students to write up their reactions and reflections about the information presented.

Lesson 8, program 1 – Drive until a black line (Worksheet 8.2)

Students write a program that drives the robot forward on a white (reflective) surface until a black (non-reflective) line is crossed.

Worksheet 8.2 walks students through a simple line tracking program. The idea of debugging code is expanded on, including examples of how to add lines of code into a program specifically for debugging purposes. Students then write and test the line tracking program using black, red, green and blue lines.

Tips and tricks:

- Remind students they must include 'Ed.LineTrackerLed(Ed.ON)' in any program running line tracking.
- Remind students that whenever they use Edison's line tracking sensor in a program, they must always start the robot on the white (reflective) surface – never the black (non-reflective) surface.
- Have students use activity sheet 8.1 for this activity.

Lesson 8, program 2 – Drive inside a border (Worksheet 8.3)

Students write a program that drives the robot forward, using the line tracking sensor to detect a non-reflective border and turn the robot around, keeping Edison inside the border.

Worksheet 8.3 introduces students to pseudocode and why it is useful in program design. Students work through an example of pseudocode, comparing it to a finished EdPy program to see how the two correspond. They then write and modify a program that will keep Edison inside a black border using the robot's line tracking sensor.

Tips and tricks:

- Remind students they must include 'Ed.LineTrackerLed(Ed.ON)' in any program running line tracking.
- Remind students that whenever they use Edison's line tracking sensor in a program, they must always start the robot on the white (reflective) surface – never the black (non-reflective) surface.
- Have students use activity sheet 8.2 for this activity or create their own border using a large piece of paper and a black marker or black electrical tape on a white desk or floor. The lines need to be a very dark colour, such as black, and approximately 1.5cm (0.6 inches) wide. Make sure the background is white or another highly reflective colour.
- For a fun class activity, place all the robots inside a large border and run the program with all the robots at the same time.

Lesson 8, program 3 – Follow a line (Worksheet 8.4)

Students translate a pseudocode algorithm for following a black line into their own Python program.

Worksheet 8.4 introduces the concept of algorithms in programming. Students then translate an algorithm written in pseudocode into a program enabling Edison to follow any black line using the robot's line tracking sensor.

Tips and tricks:

- Remind students they must include 'Ed.LineTrackerLed(Ed.ON)' in any program running line tracking.
- Remind students that whenever they use Edison's line tracking sensor in a program, they must always start the robot on the white (reflective) surface – never the black (non-reflective) surface.
- Have students use activity sheet 8.2 for this activity or create their own line using a large piece of paper and a black marker or black electrical tape on a white desk or floor. The lines need to be a very dark colour, such as black, and approximately 1.5cm (0.6 inches) wide. Make sure the background is white or another highly reflective colour.

Lesson 9: Respond to light

Environmental measurement and programming mathematics – Students utilise the Edison robot's visible light sensors to measure light level for use as variables in programs which, by performing mathematics on these variables, can be used to control the robot's behaviour.

In this lesson, students learn more about how the Edison robot's visible light sensors work. They apply knowledge from previous lessons, such as variables and mathematics operations, to programs allowing them to use the robot's light sensors to perform a range of tasks. They also practise tracing values through a program.

Students work through three main programming tasks in lesson 9 using three worksheets:

- Worksheet 9.1 – Light alarm
- Worksheet 9.2 – Automatic lights
- Worksheet 9.3 – Light following

Lesson 9, program 1 – Light alarm (Worksheet 9.1)

Students write a program that sounds an alarm when the lights in the room are turned on.

Worksheet 9.1 examines how the Edison robot's light sensors work and how the data that the sensors produce can be fed into a program. Students then work with a program which uses the light sensor's light level reading as a variable to determine the behaviour of the robot. The worksheet reminds students that mathematics can be used inside a program, in this case with the basic mathematics of using a 'less than' (<) comparison.

Tips and tricks:

- It may be helpful to review variables and expressions, including mathematical notations in expressions in Python.
- If the alarm is sounding immediately after the play button is pushed, try blocking off the left light sensor with dark paper taped on to the robot (alternatively, a few coloured sticky notes stacked together works very well) before the program is run. After the play button is pushed, have students wait a bit, then when they are ready for the alarm to sound, remove the paper. This will trigger the alarm.

Lesson 9, program 2 – Automatic lights (Worksheet 9.2)

Students write a program that drives the robot forward while monitoring light levels. If the robot drives into a dark area, the front lights are automatically turned on.

Worksheet 9.2 provides the template program and a brief explanation of what is happening in the program. Students then write and run the program as well as modifications of the program to see light level monitoring as a fluctuating input variable in action.

Tips and tricks:

- Reviewing how to define functions in Python may be helpful.
- Creating tunnels for the robots can be turned into a problem-solving design challenge to see what sorts of materials (overturned books, shoe boxes with cut-outs, chairs, etc.) work best.

Lesson 9, program 3 – Light following (Worksheet 9.3)

Students write a program that drives the robot towards a bright light, such as that from a torch/flashlight.

Worksheet 9.3 walks students through the logic of the light following program. As this program uses more computational mathematics, including subtraction and then a 'less than' (<) comparison, the practice of tracing a program is reintroduced. After students trace the program to check the flow of the maths and logic, they then write and run the program.

Tips and tricks:

- It may be helpful to review how tracing works in programming.
- This worksheet includes a final set of questions designed to challenge students to think about the intelligence of both insects and robots as well as the definition of life. This can make for a good partner, group or class discussion. It can also be extended into independent research or essay projects.

Lesson 10: A Vampire robot

Creative thinking and problem-solving – Students are first introduced to the concepts of a class definition and objects in Python, then put the knowledge from all previous lessons into a practical application as they design their own Vampire robot behaviours program.

In this lesson, which serves as a final project for the course, students learn about object-oriented programming and the concepts of a class and objects in Python. They then work through a programming project, beginning with program design, coding and testing. Students are asked to capture and reflect on failures they experience during the project and to demonstrate their understanding of both their own project and the coding structures they use.

Students work through one main program in lesson 10 using one project-style worksheet:

- Worksheet 10.1 – Vampire robot

Lesson 10, program 1 – Vampire robot (Worksheet 10.1)

Students design a program using a class and objects in Python containing a range of their own 'Vampire robot' behaviours.

Worksheet 10.1 first introduces students to the idea of object-oriented programming and the concepts of a class definition and objects in Python. Students then use the EdPy app to write a simple class for a Vampire robot with a __init__ function and two other functions: one for daytime behaviour for the robot and one for night time behaviour. Once they have completed the introduction, students embark on a programming project which acts as a final project for the course challenging students to program their robot to act like a vampire (i.e. have different behaviours in the day and the night).

The project contains 5 tasks:

1. Project design, including development of flowcharts and pseudocode
2. Coding and testing
3. Problem-solving and rework
4. Demonstration of program knowledge
5. Presentation of the final project

Tips and tricks:

- This lesson provides a basic overview of object-oriented programming and using a class in Python. If you want to have students dig deeper into how class definitions and objects can be used in Python, you can have them read more at https://en.wikibooks.org/wiki/A_Beginner%27s_Python_Tutorial/Classes as a starting point.
- It may be helpful to review key content from previous lessons before students begin designing their programs.

- This lesson may take multiple lesson sessions. You may wish to assign elements of this lesson as a homework exercise.
- Encourage students to use multiple robot sensors in their program and different programming structures, such as event handlers.
- Encourage students to set up different Vampire objects with different input parameters.
- If students are struggling with the idea of a single Edison robot being multiple Vampires (e.g. multiple Vampire objects), change the explanation slightly. Rename input parameter ‘age’ to something like ‘temperamentValue’. Then, rather than giving the Vampire objects names such as ‘Dracula’, think of the various objects as emotions or statuses. For example, ‘shy’, ‘sleepy’ ‘happy’ or ‘noisy’. This way, a single Edison robot is only one Vampire, but with a variety of Vampire tendency sets.
- If you want to more strictly define what the students’ final programs must do, consider assigning specific requirements or tasks that the programs must adhere to/achieve. These can also serve as suggestions to help students struggling to come up with ideas.
- The final task, demonstrating their program and talking through their process, can make for a good partner, group or class presentation. Alternatively, turn the project into a multimedia assignment where students write up their project and create a video of their Edison performing their program.

Answer key

Using the answer key

The EdPy lesson plans answer key contains the answers and marking guidance to all questions in the student worksheets.

ID number

The ID number identifies to which lesson, worksheet and question the answer relates. Example: Lesson 1, worksheet 1.2, questions 3 is written as L1-W1.2_Q3

Answer type

The answer type identifies what style of response the question has asked students to provide. There are three possible answer types:

- Exact input (EI): a question which has an exact solution which the student must enter.
- Result code (RC): a question which asks students to capture the code they have as a result of their programming, where a variety of code solutions are possible.
- Result input (RI): a question which asks students to capture their experiences or outcomes as a result of their experimentation and programming, where a variety of responses are possible.

Answer

The answer, or elements which should be contained in the answer, is given along with sample answers where applicable. Student work can be marked against this guide, as follows:

- Exact input (EI): The correct solution to the question is given.
- Result code (RC): Key functions or code elements which should be in the students' code are noted. A sample code solution is also provided.
- Result input (RI): Keywords or key ideas which should be included in the students' answers are noted, where applicable. A sample answer, or notes for marking, is also provided.

Answer key: lesson 1

ID number	Answer Type	Answer
L1-W1.2_Q1 (Clap controlled driving)	RI	**Keywords:** clap, two claps, drive forwards, turn **Sample answer:** Edison turns after one clap is detected and drives forward after it detects two claps.
L1-W1.2_Q2 (Avoid obstacles)	RI	**Keywords:** drive, obstacle, sense/see/detect, turn **Sample answer:** Edison drives forward until an obstacle is detected, then Edison turns away from the obstacle before driving forwards again.
L1-W1.2_Q3 (Line tracking)	RI	**Keywords:** line, reflective/white, non-reflective/black, drive/follow **Sample answer:** Edison turns away from the line while on a non-reflective surface and towards it while on a reflective surface. This causes Edison to wiggle while driving along the line.
L1-W1.2_Q4 (Follow torch)	RI	**Keywords:** turn, left, right, light/light level, drive/follow **Sample answer:** Edison follows the bright torchlight, always turning towards the side that is sensing the highest light level.
L1-W1.3_Q1	EI	Ed.EdisonVersion = Ed.V1
L1-W1.3_Q2.1	EI	0
L1-W1.3_Q2.2	EI	2
L1-W1.3_Q2.3	EI	1
L1-W1.3_Q2.4	EI	3
L1-W1.3_Q3	EI	Ed.DistanceUnits = Ed.INCH
L1-W1.3_Q4	EI	Compiler Output window
L1-W1.4_Q1	RI	**Keywords:** Turn/turned, left, right, beep/beeping, lights/LEDs **Sample answer:** Edison turned left and right while beeping and flashing the LEDs on and off.
L1-W1.4_Q2	RI	**Keywords:** One line at a time/one-by-one, commands/lines

		Sample answer: Edison followed the Python commands one at a time, line by line, working from the top of the code to the bottom, skipping blank lines and comment lines. That is why Edison turned to the right first because it is first in the code.
L1-W1.4_Q3	RI	**Keywords:** Sound/.wav file, EdComm cable, download/downloaded **Sample answer:** The program is converted into a sound file and downloaded to Edison through the EdComm cable.

Answer key: lesson 2

ID number	Answer Type	Answer
L2-W2.1_Q1	RI	Ed.CM or Ed.INCH or Ed.TIME
L2-W2.1_Q2	RI	*Answer assumes activity sheet 2.1 was used* ~16 cm OR ~6 Inches OR ~1100 milliseconds if Speed = 1 ~900 milliseconds if Speed = 2 ~800 milliseconds if Speed = 3 ~700 milliseconds if Speed = 4 ~600 milliseconds if Speed = 5 ~550 milliseconds if Speed = 6 ~500 milliseconds if Speed = 7 ~450 milliseconds if Speed = 8 ~440 milliseconds if Speed = 9 ~400 milliseconds if Speed = 10
L2-W2.1_Q3	RI	**Keywords:** Fast, accuracy, drop/lower/reduce, time to stop **Sample answer:** At speed 10 Edison is driving fast, this causes Edison to take longer to stop, and lowers Edison's accuracy.
L2-W2.2_Q1	RI	*Answer assumes activity sheet 2.1 was used* ~550-600 milliseconds
L2-W2.2_Q2.1 (fastest)	RI	*Answer assumes activity sheet 2.1 was used* ~400-450
L2-W2.2_Q2.2 (slowest)	RI	*Answer assumes activity sheet 2.1 was used* ~1000-1100
L2-W2.3_Q1	RI	*Answer assumes activity sheet 2.1 was used* *Both 'forward' and 'backwards' should have the same number.* ~16 cm OR

		~6 Inches OR ~1100 milliseconds if Speed = 1 ~900 milliseconds if Speed = 2 ~800 milliseconds if Speed = 3 ~700 milliseconds if Speed = 4 ~600 milliseconds if Speed = 5 ~550 milliseconds if Speed = 6 ~500 milliseconds if Speed = 7 ~450 milliseconds if Speed = 8 ~440 milliseconds if Speed = 9 ~400 milliseconds if Speed = 10
L2-W2.3_Q2	RC	**Key functions:** Ed.Drive(),Ed.TimeWait() **Sample code:** *NOTE: inputs to time wait function may vary slightly* Ed.Drive(Ed.FORWARD, Ed.SPEED_6, Ed.DISTANCE_UNLIMITED) Ed.TimeWait(560, Ed.TIME_MILLISECONDS) Ed.Drive(Ed.STOP, Ed.SPEED_6, 2) Ed.Drive(Ed.BACKWARD, Ed.SPEED_6, Ed.DISTANCE_UNLIMITED) Ed.TimeWait(560, Ed.TIME_MILLISECONDS) Ed.Drive(Ed.STOP, Ed.SPEED_6, 2)
L2-W2.4_Q1	EI	Is 2*2 the same as 4? True
L2-W2.4_Q2	EI	Is 2 greater than or equal to 4? False
L2-W2.4_Q3	EI	Is 2 + 2 not equal to 4? False
L2-W2.4_Q4	EI	Is 2-1 less than 4-1? False
L2-W2.5_Q1	RI	**Keywords:** Drive/drives/drove, forwards, **Sample answer:** Edison drove forward after either button was pressed.
L2-W2.5_Q2	RI	**Keywords:** Stops, standby mode/lights flashing **Sample answer:** The program just stops without Edison driving and Edison goes back to standby mode. This is because the square

		button always stops whatever program is running and puts Edison back into standby mode.

L2-W2.5_Q3	RC	**Key code elements:**
		Ed.Drive(),while, pass,
		Ed.ReadKeypad()==Ed.KEYPAD_NONE, Ed.BACKWARD
		
		Sample code:
		Ed.ReadKeypad()
		while Ed.ReadKeypad()==Ed.KEYPAD_NONE:
		  pass
		Ed.Drive(Ed.FORWARD, Ed.SPEED_6, 8)
		while Ed.ReadKeypad()==Ed.KEYPAD_NONE:
		  pass
		Ed.Drive(Ed.BACKWARD, Ed.SPEED_6, 8)

Answer key: lesson 3

ID number	Answer Type	Answer
L3-W3.1_Q1	RI	**Keywords:** Turn/turns, 90 degrees/right angle, degreesToTurn, variable, assign(ed) value **Sample answer:** Edison turns right 90 degrees. It does this because the distance input of the drive function is set to the variable degreesToTurn with the assigned value = 90.
L3-W3.1_Q2	RI	**Keywords:** Turn/turns, 90 degrees/right angle, back/left, degreesToTurn, variable **Sample answer:** Edison turns right 90 degrees to the right, then turns back left 90 degrees. This is because the distance input of the drive function in both lines is set to the variable degreesToTurn with the assigned value = 90.
L3-W3.1_Q3	RI/RC	**Sample answer:** Line 12 degreesToTurn = 180
L3-W3.1_Q4	RI	**Key idea:** Any variable name containing special characters anywhere in the name or containing numbers at the start of the name. **Examples:** degre#esToTurn degree$ToTurn degreesToTurn! 2degreesToTurn
L3-W3.2_Q1	RI	**Sample answer:** Ed.SPIN_LEFT, Ed.SPEED_4, degreesToTurn = 180
L3-W3.2_Q2	RI	**Sample answer:** Using CM: Ed.FORWARDS, Ed.SPEED_4, ~24 Using INCH: Ed.FORWARDS, Ed.SPEED_4, ~9 Using TIME: Ed.FORWARDS, Ed.SPEED_4, ~980 milliseconds
L3-W3.3_Q1	RC	**Key code elements:** While, pass, Ed.ReadKeypad(),!=, Ed.KEYPAD_TRIANGLE, Ed.Drive()

		Example code:
		degreesToTurn = 90
		Ed.ReadKeypad()
		while Ed.ReadKeypad() != Ed.KEYPAD_TRIANGLE:
		  pass
		Ed.Drive(Ed.SPIN_RIGHT, Ed.SPEED_6, degreesToTurn)
		Ed.ReadKeypad()
		while Ed.ReadKeypad() != Ed.KEYPAD_TRIANGLE:
		  pass
		Ed.Drive(Ed.SPIN_LEFT, Ed.SPEED_6, 3*degreesToTurn)

L3-W3.4_Q1	RI	**Keywords:** Forwards, left turn, right turn, 90 degrees **Sample answer:** Edison drives forwards, then turns left, drives a short distance forwards then turns right. Finally, Edison drives forwards to complete the maze.
L3-W3.4_Q2	RI	*Note:* Coding is a trial and error based experience. The exact difficulties noted by the students in this question are less significant than the acknowledgement and overcoming of those difficulties.
L3-W3.4_Q3 Challenge 1	RI	*Opposing student's name* *Winner's name*
L3-W3.4_Q4 Challenge 1	RI	*Winner's time through maze*
L3-W3.4_Q5 Challenge 2	RI	*Drawing of maze*

Answer key: lesson 4

ID number	Answer Type	Answer
L4-W4.1_Q1	RI	**Key code elements:** Ed.Drive(), Ed.SPIN_LEFT, Ed.FORWARD, (variable) **Example code:** degreesToTurn = 90 Ed.Drive(Ed.FORWARD, Ed.SPEED_5, 15) Ed.Drive(Ed.SPIN_LEFT, Ed.SPEED_5, degreesToTurn) Ed.Drive(Ed.FORWARD, Ed.SPEED_5, 15) Ed.Drive(Ed.SPIN_LEFT, Ed.SPEED_5, degreesToTurn) Ed.Drive(Ed.FORWARD, Ed.SPEED_5, 15) Ed.Drive(Ed.SPIN_LEFT, Ed.SPEED_5, degreesToTurn) Ed.Drive(Ed.FORWARD, Ed.SPEED_5, 15) Ed.Drive(Ed.SPIN_LEFT, Ed.SPEED_5, degreesToTurn)
L4-W4.1_Q2	EI	8
L4-W4.1_Q3	RI	**Sample answer:** Yes. Ed.Drive(Ed.FORWARD, Ed.SPEED_5, 15) was used 4 times. Ed.Drive(Ed.SPIN_LEFT, Ed.SPEED_5, degreesToTurn) was used 4 times.

L4-W4.2_Q1	RC	**Key code elements:**
		for, range(), Ed.Drive(), Ed.SPIN_LEFT, Ed.FORWARD, (variable)
		
		Example code:
		degreesToTurn = 90
		for x in range(4):
		  Ed.Drive(Ed.FORWARD, Ed.SPEED_5, 15)
		  Ed.Drive(Ed.SPIN_LEFT, Ed.SPEED_5, degreesToTurn)

L4-W4.3_Q1	EI	3
L4-W4.3_Q2	EI	6
L4-W4.3_Q3	RI	**Key idea:** The number of times the loop executes is equal to the number of sides of the shape. **Sample answer:** The loop executes the same amount of times as the shape has sides.
L4-W4.3_Q4	EI	12
L4-W4.4_Q1	RI	*Will depend on the distance the robot travels and degrees it turns* *Approximate minimum:* 20-40

L4-W4.4_Q2	RI	*Will depend on the number of loop executions and degrees it turns* *Approximate maximum:* 1-2 cm
L4-W4.4_Q3	RI	**Keywords:** No, turn(ing), small (forward) movements **Sample answer:** No, Edison is actually turning and making small movements forward each loop, rather than driving in a continuous arch.

Answer key: lesson 5

<table>
<tr><th>ID number</th><th>Answer Type</th><th>Answer</th></tr>
<tr><td>L5-W5.1_Q1</td><td>RI</td><td>Keywords:
Waits/finishes(d), music/sound/note, drive(s) forward

Sample answer:
Edison played a note for 2 seconds, then drove forward after the note finished.</td></tr>
<tr><td>L5-W5.1_Q2</td><td>RI</td><td>Key idea:
The loop is comparing the two sides of the expression to see if the music is finished. While it is not, the program does not continue onto the next line.

Sample answer:
The loop is comparing the left side of the expression to the right to see if the music is finished. Whenever the expression evaluates to ‘true’ (i.e. the music is not finished) the program continues to execute the loop. Once it evaluates to ‘false’, the program continues to the next line.</td></tr>
<tr><td>L5-W5.1_Q3</td><td>RI</td><td>Keywords:
Simultaneously/at the same time/while, music/sound/note, drive(s) forward

Sample answer:
Edison began to play the note, then started to drive forward while still playing the note. Edison finished driving before the note stopped playing.</td></tr>
<tr><td>L5-W5.1_Q4</td><td>RI</td><td>Key idea:
Music plays in the background unless a while loop with Ed.ReadMusicEnd() is used.

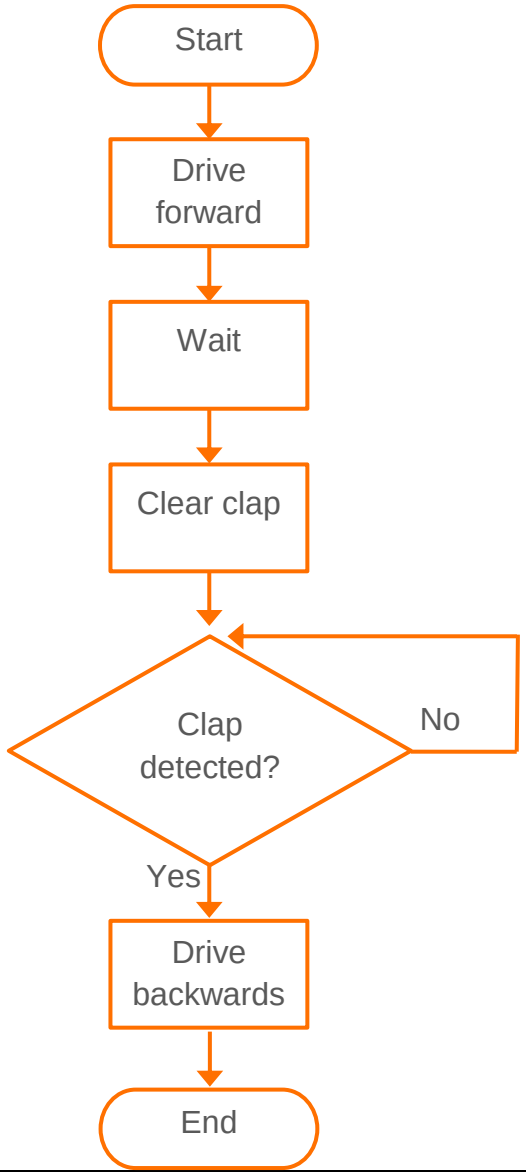
Sample answer:
The second program does not have an Ed.ReadMusicEnd() function in a ‘while’ loop, so the music plays in the background, and the program continues with the next lines of the program.</td></tr>
<tr><td colspan="3"></td></tr>
<tr><td>L5-W5.2_Q1</td><td>RI</td><td>Keywords:
Sounds/tones, frequency/pitch decreasing, period increasing

Sample answer:
Edison makes sounds which decrease in frequency each loop. This is because the period is increasing each loop.</td></tr>
<tr><td>L5-W5.2_Q2</td><td>EI</td><td><table><tr><td>Value of i</td><td>1st input parameter to PlayTone()</td></tr><tr><td>0</td><td>100</td></tr><tr><td>1</td><td>200</td></tr><tr><td>2</td><td>300</td></tr></table></td></tr>
</table>

L5-W5.2_Q3	EI	32
L5-W5.2_Q4	EI	3300
L5-W5.2_Q5	EI	33
L5-W5.3_Q1	EI	Ed.TEMPO_VERY_FAST Ed.TEMPO_FAST Ed.TEMPO_MEDIUM Ed.TEMPO_SLOW Ed.TEMPO_VERY_SLOW
L5-W5.3_Q2	EI	Ed.TEMPO_VERY_FAST
L5-W5.3_Q3	RI	**Key idea:** Pairs of characters (a note and duration per pair) must be removed together for a shorter version of the tune to continue to play. **Sample answer:** I removed 14 characters from the middle of the tune string (seven pairs of note/duration), so the tune would skip a verse.
L5-W5.4_Q1	EI	4
L5-W5.4_Q2	EI	4
L5-W5.4_Q3	RI	**Key idea:** By having Edison turn a little past the start point before the loop, Edison swings past the start point in the loop. **Sample answer:** Line 19 of the program makes Edison move off the start line so that during the loop, Edison will shimmy back and forth past the start point. That is why the first movement must be shorter than the rest and why we want it in the program.
L5-W5.5_Q1	RI	*Note:* Any movement can be accepted here, but bonus points for movements actually synced to the tune playing!
L5-W5.5_Q2	RI	*Note:* Any tune can be accepted here.

Answer key: lesson 6

ID number	Answer Type	Answer
L6-W6.1_Q1	RI	~1.5m
L6-W6.1_Q2	RI	**Keywords:** Duration, on and off, **Sample answer:** The TimeWait() functions add a longer duration to the turn LED on/off action. Without the functions, the LED would turn on and off too fast to see.
L6-W6.1_Q3	RI	**Keywords:** Pass, clap detected, loop back/return/flow to **Sample answer:** When the decision of 'clap detected?' is 'no,' the program will pass and loop back to the start of the 'while' loop, and ask the question again.
L6-W6.2_Q1	RI	**Key idea:** The additional read function call clears any previous data from before the while loop, making sure no previous clap detections are being stored. **Sample answer:** A read function in a loop may register that event from before the read function is called in the code. To make sure the read function in the while loop will not consider any claps that happen before the while loop, the extra 'read' function call is made to clear any previous claps from before the loop.

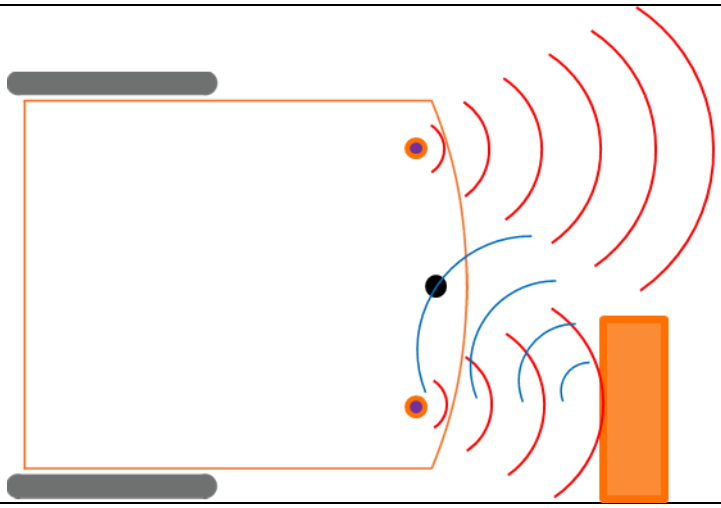
L6-W6.2_Q2	RI	**Example flowchart:** 

L6-W6.3_Q1	RI	**Sample answer:**
		def driveInaSquare():
		for i in range(4):
		Ed.Drive(Ed.FORWARD, Ed.SPEED_6, 2)
		Ed.Drive(Ed.SPIN_LEFT, Ed.SPEED_9, 90)

L6-W6.3_Q2	EI	driveInaSquare()
L6-W6.3_Q3	RI	**Sample answer:** No, Edison is turning a little more than a right angle, so all squares after the first are slightly different.
L6-W6.3_Q4	RI	*Note:* Any function description using a clap can be accepted here.
L6-W6.3_Q5	RI	*Note:* Coding is a trial and error based experience. The exact difficulties noted by the students in this question are less

		significant than the acknowledgement and overcoming of those difficulties.

Answer key: lesson 7

ID number	Answer Type	Answer
L7-W7.1_Q1	EI	
L7-W7.2_Q1	RI	**Key idea:** While the program will download, Edison will not detect obstacles and will continue to drive forward even when detectable obstacles are in front of the robot because the obstacle detection beam is not on. **Sample answer:** I checked the code for errors, but it did not show as having any and I could download the program to Edison. When I hit play, Edison drove forward straight into a wall. Even when the robot was up against a wall, it kept trying to drive forward. This happened because, even though the program was using obstacle detection, the obstacle detection beam was never turned on.
L7-W7.2_Q2	RI	*Note:* Any real-world example is acceptable. **Sample answer:** I have seen this in the reversing sensors on a car.
L7-W7.2_Q3	RI	*Note:* Any example where obstacle detection would be helpful is acceptable.
L7-W7.3_Q1	RI	**Key idea:** Change the program with an improvement of some kind, such as: • A loop to repeat functionality • Lights or sounds to better signal an obstacle has been detected • Different conditions to change where an obstacle is detected from

<table>
<tr><td></td><td></td><td>Sample answer:
I could add a loop so that the program would be able to keep detecting and avoiding obstacles.</td></tr>
<tr><td>L7-W7.3_Q2</td><td>EI</td><td>
<table>
<tr><th>Error #</th><th>Line #</th><th>Error type</th><th>Error description</th></tr>
<tr><td>1</td><td>17</td><td>syntax</td><td>The word ‘while’ is spelt with a capital W.</td></tr>
<tr><td>2</td><td>17</td><td>logical</td><td>“Ed.ReadObstacleDetection() != Ed.OBSTACLE_NONE” should be Ed.ReadObstacleDetection() == Ed.OBSTACLE_NONE</td></tr>
<tr><td>3</td><td>17</td><td>syntax</td><td>The line needs to end with a colon (:).</td></tr>
<tr><td>4</td><td>18</td><td>syntax</td><td>This line should be indented.</td></tr>
<tr><td>5</td><td>19</td><td>logical</td><td>The last parameter of Ed.Drive() should be 135, not 145.</td></tr>
</table>
</td></tr>
<tr><td colspan="3"></td></tr>
<tr><td>L7-W7.4_Q1</td><td>RI</td><td>Keywords:
Drive, forward(s), detect obstacle, spin/turn, away

Sample answer:
Edison starts driving forwards. Then, when an obstacle is detected, Edison spins away from it before continuing to drive forward.</td></tr>
<tr><td>L7-W7.4_Q2</td><td>EI</td><td>Key idea:
Lines 19 and 20. This is the ‘avoidObstacle’ function called when the event occurs.

Sample answer:
Lines 19 and 20. This is because when an obstacle is detected ahead, the ‘avoidObstacle’ function is called and the ‘avoidObstacle’ function is defined by line 18 as containing lines 19 and 20.</td></tr>
<tr><td>L7-W7.4_Q3</td><td>EI</td><td>Key idea:
This read call clears any previously stored data.

Sample answer:
Line 20 is a read call which clears any ‘obstacle detected’ data already stored, resetting the program to look for a new obstacle.</td></tr>
<tr><td colspan="3"></td></tr>
<tr><td>L7-W7.5_Q1</td><td>RI</td><td>Key idea:
While the robot is acting autonomously, it is not learning and therefore does not have intelligence (artificial or otherwise).

Sample answer:</td></tr>
</table>

		No, Edison is not remembering or learning anything about the inputs and would crash into the obstacle if any one of the lines of code was changed.
L7-W7.5_Q2	EI	**Sample answer:** Obstacle detected ahead: Spin right 180 degrees Obstacle detected on right: Spin left 90 degrees Obstacle detected on left: Spin right 90 degrees

Answer key: lesson 8

ID number	Answer Type	Answer
L8-W8.1_Q1	EI	White
L8-W8.1_Q2	EI	Red surface: reflective Green surface: non-reflective Blue surface: non-reflective
L8-W8.2_Q1	EI	Red
L8-W8.2_Q2	RI	**Keywords:** Red, reflect/reflective, detect **Sample answer:** Edison uses a red LED in the line tracking sensor to detect a surface. As red surfaces reflect a lot of red light, Edison sees a red surface as a reflective surface.
L8-W8.2_Q3	RI	**Key idea:** Each flag can be assigned a unique indicator so the program can alert the user when any given flag has been detected. **Sample answer:** I would assign each flag a different function, so one beep, one left LED and one right LED. When Edison detects one of the flags, it uses the output assigned to that type of flag to signal it has been seen. That would let me know what flags Edison has seen.
L8-W8.3_Q1	RI	*Note: as the speed increases, problems will occur. These begin at approximately speed 5 or speed 6.* **Sample answer:** SPEED_6
L8-W8.3_Q2	RI	**Keywords:** Overshoot/run over, turn, too fast, outside/crosses (the border) **Sample answer:** If Edison is going too fast, the robot will overshoot the line before beginning to turn. This results in Edison turning outside the border.
L8-W8.4_Q1	RI	**Key idea:** A low speed and distance_unlimited produce the best result. **Sample answer:** The best combination I found was SPEED_2 for speed and DISTANCE_UNLIMITED for distance.

L8-W8.4_Q2	RC	**Key code elements:**
		Ed.LineTrackerLed(Ed.ON), while True, if, else, Ed.ReadLineState(), Ed.Drive(), Ed.FORWARD_RIGHT, Ed.FORWARD_LEFT
		
		Example code:
		Ed.LineTrackerLed(Ed.ON)
		while True:
		  if Ed.ReadLineState() == Ed.LINE_ON_BLACK:
		  Ed.Drive(Ed.FORWARD_RIGHT, Ed.SPEED_3, Ed.DISTANCE_UNLIMITED)
		  else:
		  Ed.Drive(Ed.FORWARD_LEFT, Ed.SPEED_3, Ed.DISTANCE_UNLIMITED)

Answer key: lesson 9

<table>
<tr><th>ID number</th><th>Answer Type</th><th>Answer</th></tr>
<tr><td>L9-W9.1_Q1</td><td>RI</td><td>Note: any appropriate real-world scenario is acceptable.

Sample answer:
This could be used in a safe. The way it would work is that when the safe is dark (so when it is closed), the alarm does nothing. However, when the safe is opened, light is allowed in, and the alarm would sound.</td></tr>
<tr><td>L9-W9.1_Q2</td><td>RI</td><td>Note: Multiple solutions are possible. The simplest solution is included below.

Sample answer:
Change line 13 to: while Ed.ReadLeftLightLevel()>100:</td></tr>
<tr><td colspan="3"></td></tr>
<tr><td>L9-W9.2_Q1</td><td>RI</td><td>Key idea:
The higher the value, the lower a reduction of light (e.g. amount of shade or degree of darkness) is needed to turn the lights on.

Sample answer:
Edison turns the lights on with only a small amount of shade.</td></tr>
<tr><td>L9-W9.2_Q2</td><td>RI</td><td>Key idea:
The lower the value, the higher a reduction of light (e.g. amount of shade or degree of darkness) is needed to turn the lights on.

Sample answer:
Edison will only turn the lights on when it gets really dark.</td></tr>
<tr><td colspan="3"></td></tr>
<tr><td>L9-W9.3_Q1</td><td>EI</td><td>
<table>
<tr><th></th><th>Right_Light</th><th>Left_Light</th><th>Expected behaviour</th></tr>
<tr><td>Torch on right</td><td>200</td><td>100</td><td>Drive forward right</td></tr>
<tr><td>Torch on left</td><td>100</td><td>200</td><td>Drive forward left</td></tr>
<tr><td>No torch</td><td>100</td><td>100</td><td>Drive forward right</td></tr>
</table>
</td></tr>
<tr><td>L9-W9.3_Q2</td><td>RI</td><td>Key idea:
Edison avoids light from whichever side, driving away from it instead of towards it.

Sample answer:
Edison turns away from the light, regardless of the side the light is on. Edison avoids light.</td></tr>
</table>

Answer key: lesson 10

ID number	Answer Type	Answer
L10	RI	*Notes:* As this a project-based design challenge, all answers will vary from student to student. However, all work in this lesson needs to flow from one stage to the next. For example, the flowcharts must be consistent with the pseudocode, which must, in turn, be consistent with the actual code. Coding is a trial and error based experience. The exact problems identified by the students and the number of attempts at resolving these issues are less significant factors than the acknowledgement, creative problem solving and overcoming of such difficulties. When complete, students should be able to demonstrate an accurate understanding of the individual structures they use in their program as well as the overall logic of the final program.

Student programming achievement chart

Program	Stamp
2.1 Drive the robot forward	
2.2 Drive the robot backward	
2.3 Forward, then backwards	
2.5 Keypad activated driving	
3.1 Turn right	
3.2 Turn left 180°	
3.3 Right turn, then left turn	
3.4 Mini maze	
4.1 Drive in a square	
4.2 Use a loop to drive in a square	
4.3 Drive in a triangle and a hexagon	
4.4 Challenge! Drive in a circle	
5.1 Play tones	
5.2 Make an alarm	
5.3 Play a tune	
5.4 Make your robot dance	

Program	Stamp
5.5 Challenge! Dance to music	
6.1 Flash the LED in response to a clap	
6.2 Drive in response to a clap	
6.3 Design your own function	
7.2 Detect an obstacle and stop	
7.3 Obstacle avoidance	
7.4 Detect an obstacle as an event	
7.5 Right and left obstacle detection	
8.2 Drive until a black line	
8.3 Drive inside a border	
8.4 Follow a line	
9.1 Light alarm	
9.2 Automatic lights	
9.3 Light following	
10.1 Vampire robot	

Congratulations!

This certifies that

Student's name

has completed the

EdPy robotics and programming course

and in so doing has:

- Used the Edison robot and EdPy programming language
- Designed, tested and evaluated robot programs
- Developed an understanding of robot programming principals
- Demonstrated an understanding of robot movement
- Utilised robot sensors in an advanced programming language
- Applied acquired knowledge to design solutions to challenges

Well done on completing another step of your robotics and coding adventure!

Edison
robotics for everyone

Teacher's Signature

/ /
Date