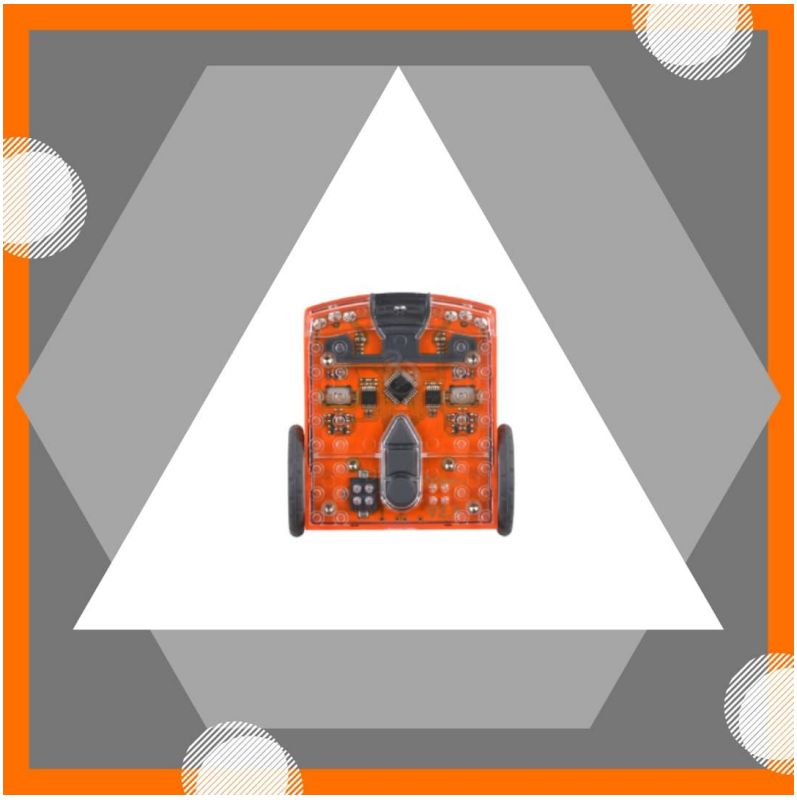
Robot shape-up

Teacher's notes



Contents

About this lesson and guide 3
Lesson overview 4
Part 1: What is a variable? 6
Part 2: What is a ‘for’ loop? 7
Part 3: How do you drive a different shape? 8

Go ahead – show off!

We love seeing how classrooms use Edison! If you and your students want to share your Hour of Code Edison EdVenture, be sure to tag us into the fun!

@meetedison twitter.com/meetedison

@meet_edison instagram.com/meet_edison

@meetedison facebook.com/meetedison

About this lesson and guide

This guide offers teachers and instructors overview information, facilitation recommendations and other supporting information for the *Robot shape-up* lesson available at https://meetedison.com/robotics-lesson-plans/robot-shape-up/

Do you need to read this whole guide to run the lesson? **Absolutely not!**

Once the robots and programming devices are set up[1], you can start learning along with your students! The student sheets for this lesson have been designed to allow students to work through the different parts of the lesson independently, learning key computer science objectives and practicing new skills as they go. This guide simply offers further information for teachers and instructors to help make using this lesson easy and fun.

Each section of the lesson is included in this guide along with any relevant supporting information for that part. Supporting information is divided into the following sections:

Overview

General information about the section and key learning objectives covered.

Delivery recommendations

Suggestions for how you can cover the lesson section if you want to run the lesson in a more facilitator-led capacity.

Tips and tricks

Helpful hints and ways to overcome common issues students may encounter.

Creative Commons licence attribution details

Developed and written by: Kat Kennewell

[1] The *Getting started with Edison and EdPy* set available at https://meetedison.com/robotics-lesson-plans/robot-shape-up/ has step-by-step help for setting up your robots and programming devices. If you are new to Edison or EdPy, it is recommended you start with that guide.
[2] https://meetedison.com/robot-programming-software/edpy/#edpy-resources
[3] https://meetedison.com/robot-programming-software/edscratch/#EdScratch-resources
[4] http://creativecommons.org/licenses/by-sa/4.0/

Lesson overview

Introduce the key computational concepts of definite loops and variables using Edison robots and EdPy[5], a text-based programming language based on Python. Students practice using definite loops (the 'for' loop) and pattern recognition by programming the Edison robot to drive in a square. They then extrapolate what they have learned in order to drive other familiar regular polygons before taking on the challenge of less common regular polygons and irregular shapes.

Grade levels	Difficulty	Duration
Year 6+	Intermediate	55+ minutes

Computer science and computational thinking topics	☐ Variables ☐ Definite loops (the 'for' loop) ☐ Pattern recognition and extrapolation
Tie-ins to other subjects	☐ Mathematics: geometry

Prerequisite knowledge

To be successful with this activity, it is recommended that students:

- Have used Edison and EdPy[6]
- Understand sequence, sequential programming, inputs and input parameters[7]
- Are familiar with polygon shapes and using degrees as a measurement for angles

Supplies you need

- Full set of Edison robots[8] and EdComm programming cables
- Full set of prepared programming devices (computers)
- 4x AAA batteries per robot
- Print or digital copies of the student sheets and print-outs of the activity sheets
- Space and supplies for test-spaces [such as large paper and markers or coloured electrical tape]
- **Optional**: supplies for the 'artist' pen challenge [such as EdCreate kits, LEGO bricks, rubber-bands and other similar materials, plus pens or markers]

---

[5] https://meetedison.com/robot-programming-software/edpy/

[6] The *Getting started with Edison and EdPy* set available at https://meetedison.com/robotics-lesson-plans/robot-shape-up/ has a step-by-step activity for introducing Edison and EdPy. If this is your first time using Edison or EdPy, start with that guide and activity.

[7] The *Edison and EdPy: Get Edison moving* lesson set available at https://meetedison.com/robotics-lesson-plans/get-edison-moving-in-edpy/ covers these concepts and introduces the syntax requirements of these concepts in the EdPy language. Consider running this beginner lesson prior to starting the *Robot shape-up* activity.

[8] This activity assumes Version 2.0 Edison robots. If you have Version 1 Edison robots, you will need to ensure students select Version 1 robots when loading EdPy and only use time as the constant for distance units. See more information in The *Getting started with Edison and EdPy* set available at https://meetedison.com/robotics-lesson-plans/get-edison-moving-in-edpy/

Aren't variables hard? A special note about this lesson

A key focus of this lesson is variables and data, two of the most fundamentally important parts of general-purpose programming languages. It's not uncommon for students (and instructors!) to feel like these are 'hard' areas of programming, especially if this is your first time working with variables. Learning about variables and maths in programming is an important step in meaningful computer science education. Just like anything that is brand new, becoming comfortable with maths and variables in programs can take some time. Encourage students to be patient with themselves as they work through the activity. They will soon see the creative potential these new skills unlock inside programming!

Some great advice from the Hour of Code team

It's okay not to know! Respond to student questions and struggles with phrases like:

- "I don't know. Let's figure this out together."
- "Technology doesn't always work out the way we want."
- "Learning to program is like learning a new language; you won't be fluent right away."

And don't forget to have fun! **(^_^)**

Part 1: What is a variable?

Overview

The concept of variables, one of the fundamental components of code, is introduced in this section. The student sheets explain how variables work, then asks students to practice using a variable with sequential programming to get their Edison robot to drive a square.

Delivery recommendations

- Recommended time: 15 minutes
- Depending on your students' reading levels and comfort with coding, you may choose to run this section as an instructor-led explanation and demonstration.
- If you are using Version 1 Edison robots with EdPy, be sure that students always have the correct setup code[9]. Pre-made programs, including some of the example programs shown in this activity, will need to have the following changes made to the setup code[10]:
    - Ensure that the Ed.EdisonVersion = Ed.V1
    - Ensure that the Ed.DistanceUnits = Ed.TIME

A note about turning degrees in the Ed.Drive() function:

- Version 2.0 Edison robots can use Ed.CM or Ed.INCH as the constant for Ed.DistanceUnits. If you are using version 2.0 robots it is strongly recommended you choose one of these options for this activity.
- When either Ed.CM or Ed.INCH is used as the constant for Ed.DistanceUnits, and a 'spin' or 'turn' is used for the direction input in the Ed.Drive() function, EdPy will convert the distance unit input number into degrees. In other words:

    Ed.Drive(Ed.SPIN_LEFT, Ed.SPEED_10, 90) → Edison will spin left 90 degrees going speed ten.

    Ed.Drive(Ed.FORWARD, Ed.SPEED_10, 90) → Edison will drive forward 90 (cm/inches) going speed ten.

- EdPy will NOT convert Ed.TIME into degrees in the Ed.Drive() function. Students will need to work out how long it takes the robot to turn the desired distance.

[9] Edison V1 robots can only use TIME, not CM or INCH, as the input parameter for distance units. That's why the setup code must always be Ed.DistanceUnits = Ed.TIME. The time distance unit is milliseconds, which means that to drive for 2 seconds, you need to set the distance input parameter to 2000 (since 2000 milliseconds = 2 seconds).

[10] If a program was made using Ed.CM as the distance units constant and you change it to Ed.TIME, you may also need to change the input parameter's values inside the program to adjust for this change. Why? If the original program said to drive for '5' where '5' was CM and it is now milliseconds, the result will be very different!

Tips and tricks

- Due to minor mechanical differences in the motors and encoders inside different Edison robots, some robots may not turn to exactly 90 degrees when given the input of 90. Encourage students to try different values around 90 (e.g. 87 or 93) to find the input that works best for their Edison.

Answer key[11]

1. Edison turns right 90 degrees.
2. Yes: Ed.Drive(Ed.FORWARD, Ed.SPEED_6, 15) was used 4 times. Ed.Drive(Ed.SPIN_RIGHT, Ed.SPEED_6, degreesToTurn) was used 4 times.

Part 2: What is a 'for' loop?

Overview

Definite loops, including the 'for' loop, are foundational concepts in coding. Students need to understand both the basic concept of what a loop does (makes other code repeat) and how to construct a loop (how to write a loop in text-based coding) in order to use loops in their own programs. The concept of variables is also built upon in this section. Students see how a variable can be used to track a piece of information even as the value of that information changes.

As students begin to work with the 'for' loop and range() function, they will start to recognise patterns related to iteration in their code and the regular polygon shapes they are working with.

Delivery recommendations

- Recommended time: 10 minutes
- This is an information-dense section as it sets up students for the active application of the concepts in the next section of the activity. Depending on your students' reading levels and comfort with coding, you may choose to run this section as an instructor-led explanation and demonstration.
- For many students, learning syntax isn't the most exciting thing to do when learning to code. Explain to students that the point isn't about memorising syntax rules: it's about understanding what is actually happening in the code. Being able to follow what is happening in a program empowers students to create their own code with far less frustration and greater creative options.

[11] Student answers can vary from the answers provided. If their code or wording differs from the sample answer that's fine! The important thing is that they understand the core concepts.

Tips and tricks

- Remind students that it's okay not to understand everything when they are first learning new concepts. Trying out the activities themselves will often help concepts make more sense.
- Remind students they can read through the documentation about the 'Ed.Drive()' function for more information about using that function. Reading documentation and being able to extract from it what you need to write code is an important skill in general-purpose programming.
- When turning, if you have Ed.DistanceUnits set to be either Ed.CM or Ed.INCH, the robot will convert the value you enter as the distance unit parameter to degrees.
    - If you have Ed.DistanceUnits = Ed.TIME the distance unit parameter will be time in milliseconds.
- 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 value of degrees input. For example, the robot may not turn exactly 90 degrees when given the input of 90. Encourage students to try different values around the value they want (e.g. 88 or 93 if they want the robot to turn 90 degrees) to find the input that works best for their Edison.

Answer key[12]

3. 4
4. A square has four sides and four equal angles. Edison needs to repeat the same sequential actions (drive and turn) four times each to drive a complete square.

Part 3: How do you drive a different shape?

Overview

It's time to experiment! Students apply what they've learned to a set of progressively more 'open' challenges. Extrapolating on the patterns they recognised in previous sections, students program Edison to drive in different shapes. Activity sheets of a triangle and hexagon offer stepping-stone practice before students design and test their own shapes.

Delivery recommendations

- Recommended time: 30 minutes
- Depending on your students' comfort with geometry you may choose to provide them with the degree inputs they need for the triangle and hexagon:

[12] Student answers can vary from the answers provided. If their code or wording differs from the sample answer that's fine! The important thing is that they understand the core concepts.

    - Triangle (outside angle): 120 degrees.
    - Hexagon (outside angle): 60 degrees.
- Regular polygons will be easier for students to program than irregular shapes. The patterns students have seen with the square, triangle and hexagon can be extrapolated out with any regular polygon. Irregular shapes will require students to use loops and sequence more creatively.
- The 'artist' pen challenge brings in engineering and art to physical computing, but can take a while. If you have the opportunity, consider making this activity its own lesson to allow students the chance to explore the design-build-test cycle as they design their pen attachment. You can see ideas for how to run this bonus challenge at https://meetedison.com/teach-engineering-design-with-edcreate/

Tips and tricks

- A helpful hint for completing the triangle and hexagon activity sheets is that the robot is actually turning on the outside of the polygon, not the inside.
- Remind students that when turning, if you have Ed.DistanceUnits set to be either Ed.CM or Ed.INCH, the robot will convert the value you enter as the distance unit parameter to degrees.
    - If you have Ed.DistanceUnits = Ed.TIME the distance unit parameter will be time in milliseconds.
- 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 value of degrees input. For example, the robot may not turn exactly 90 degrees when given the input of 90. Encourage students to try different values around the value they want (e.g. 88 or 93 if they want the robot to turn 90 degrees) to find the input that works best for their Edison.
- First attempts in programming are almost always 'wrong' – and that is perfectly okay! Persevering and changing programs iteratively, testing as they go, will get students to the solutions they want. Resilience is key!

What should I do if a student finishes the lesson early?

- Get students to complete all of the bonus challenges.
- Or, ask students who finish early to help classmates who are having trouble with the activity.