

Introduction to Python Programming

Using LEGO® Education SPIKE™ Prime Set

Introduction to Python Programming

Using LEGO® Education SPIKE™ Prime Set

OVERVIEW

In this course, students will learn the fundamentals of the Python programming language, along with programming best practices, through using LEGO® Education SPIKE™ Prime set. Through a series of scaffolded lessons, students will learn to import important libraries, how to use the hardware and software to control motors and sensors, use conditionals and loops to control the flow of your programs, and store data using Python data types and variables. They will define and document their own custom programs, write scripts, and handle errors. Most importantly, students will have multiple and ongoing opportunities to use all of this knowledge in authentic contexts to practice and develop their coding skills in Python.



By the end of the course, students will

- Design, iteratively develop and program a prototype of a robot or model
- Work collaboratively, give and receive feedback, and incorporate suggestions
- Debug and troubleshoot both hardware and software problems
- Use algorithms, data, compound conditionals, sensors, loops and Boolean logic
- Document programs, feedback, testing, and debugging
- Articulate flowcharts or pseudocode to address complex problems
- Decompose problems and subproblems into parts
- Discuss issues of bias and accessibility
- Communicate the solution to a problem, including model and programming

LEARNING PROMISE

Over the duration of this course, students will develop their Python programming knowledge by analyzing, prototyping, and communicating ideas in the areas of abstraction, algorithms, programming, and data. Students will create artifacts and build models that use motors, sensors, lights, and sounds to work effectively by creating Python programs. They will utilize various programming techniques including writing pseudocode, using conditional statements, loops, Boolean logic, as well as linear and computational thinking to accomplish a variety of tasks. Students will apply their Python knowledge to various guided and open-ended projects which culminate in presenting solutions to real-world problems.

COURSE DESIGN

The course has been developed to address numerous Python programming skills and outcomes. The lessons have been compiled into scaffolded experiences to increase in complexity to enable teachers to provide their students with

ongoing opportunities to develop proficiency in Python programming.

This course is built around the K-12 CS Framework and the CSTA standards. A matrix of the lessons, the standards covered in each lesson, and framework areas addressed is available at the end of this document.

The course is broken down into five units and a culminating project. The units have 6-8 lessons. The projects have between 10-12 lessons. During each 45-60-minute lesson, students will experience a high level of engagement to develop their Python proficiency. The course contains the following units:

- Unit 1: Hardware + Software
- Unit 2: Motors
- Unit 3: Sensor Control
- Unit 4: Loops and Variables
- Unit 5: Conditions for Games
- Project Unit: Mental Agility

Throughout the lessons, strategic questions and key objectives will guide students through the process of developing Python programming proficiency. The key objectives listed on each lesson can be used to determine whether or not each student is developing the relevant skills.

The projects within the course will include specific documentation via journaling and rubrics to detail their overall understanding and application of the Python concepts previously covered.

GETTING STARTED WITH THE SPIKE APP USING PYTHON

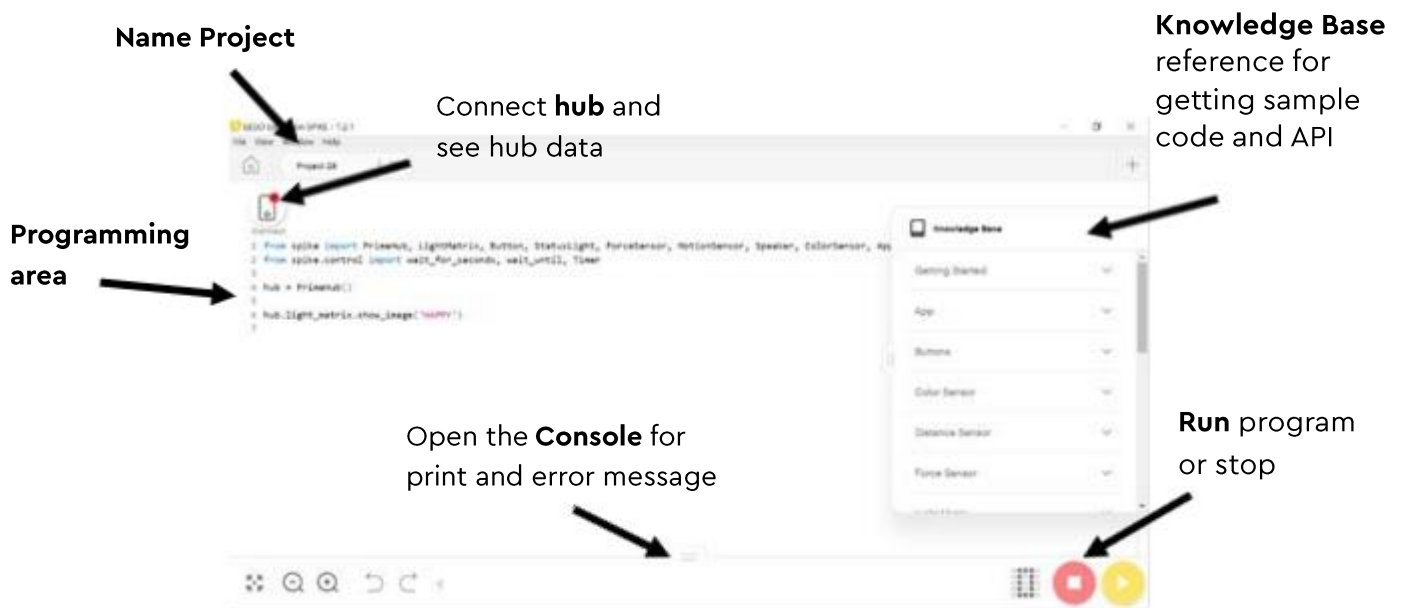
Opening a new project

Direct students to open the software and look at library options.

- Open the SPIKE App
- Select New Project
- Select PYTHON
- Select CREATE

Using the Programming Canvas

We suggest allowing the students a few minutes in the first lesson to see the functions within the software and pointing out important features as shown in the image below.



- The programming canvas is the open white space starting on the top left side. When you open the software there will be a sample program already loaded in the programming canvas.
- You will see the hub icon in the top left corner which is where you will connect your hub.
- Along the bottom you will see the console and some functions that allow you to change the screen size and undo actions. The console is where printed code will appear as well as error messages. If the console is not showing, click the two horizontal bars in the center bottom of the canvas. The console will move up to be visible.
- Along the right side, you will see the knowledge base. The knowledge base will provide support and act as a reference for code.

Using the Knowledge Base

The Knowledge Base provides getting started support and easy transition to text-based coding for SPIKE Prime with python. The knowledge base is the API, providing a place to find the functions linked to the hardware from the SPIKE Prime set. Additionally, sample codes that can be copied and pasted into the programming canvas, error message explanations, and the type and value of each function are provided in the Knowledge Base.

Copy and Pasting from the Knowledge Base

To make programming simpler, there are several sample programs provided in the Knowledge Base. Students can copy and paste these into the programming canvas at any time. To copy and paste:

- click the small blue icon in the upper right corner of the programming box within the Knowledge Base
- right click in the box that you wish to paste the code
- choose PASTE

Connecting the Hub via Bluetooth

Guide students through connecting their hubs to the software. The hub can be connected with the USB cable or through Bluetooth. To connect via Bluetooth:

- click the hub icon
- click CONNECT VIA BLUETOOTH in the upper right corner
- press the small circular button on the upper left part of the hub
- the hub name should appear at the bottom of the screen in a few seconds
- click on the correct hub name and it will be connected
- return to the programming canvas by clicking the X to close the connection dashboard screen.

Renaming a Project

Projects will be saved in the My Projects tab. To easily locate a project, students should give each project a name relevant to the task. To rename the Project:

- click the three small vertical dots to the right of the Project name.
- two choices will open in the pop-up menu, RENAME PROJECT and MOVE TO
- choose RENAME PROJECT
- the screen changes to show the current name
- erase what is there and type in your own title
- click save
- the menu will close and return to the programming canvas

ASSESSMENT

In this course, there are three types of assessment used for lessons and an additional rubric used in the culminating project. Teachers are encouraged to use all types of assessment as a way to provide students adequate feedback to continue grow their knowledge and skills. The three types of assessment include:

- Teacher observations which encourages teachers to discuss outcomes with students through posing questions and listening to how students as a way to check for understanding.
- Peer feedback which allows students to learn how to provide and take constructive feedback that can better their solutions.
- Self-assessment which allows students to deeply reflect on their own learning to make connections, think about how to work together, and complete work in positive ways.

Introduction to Python Programming

Unit 1 Hardware and Software		
Lesson	Objectives	CSTA Standards
Importing Libraries 45 minutes	<ul style="list-style-type: none"> Learn why a Python program must have libraries imported. Import libraries. Run a program. 	2-CS-02 Design projects that combine hardware and software components to collect and exchange data. 2-AP-16 Incorporate existing code, media, and libraries into original programs, and give attribution.
Communicating with Light 45 minutes	<ul style="list-style-type: none"> Describe the function of hardware and software. Program the light matrix and learn how to debug a simple program. 	2-CS-02 Design projects that combine hardware and software components to collect and exchange data. 2-AP-13 Decompose problems and subproblems into parts to facilitate the design, implementation, and review of programs. 2-AP-16 Incorporate existing code, media, and libraries into original programs, and give attribution. 2-AP-19 Document programs in order to make them easier to follow, test, and debug.
Pair Programming 45 minutes	<ul style="list-style-type: none"> Practice pair programming. Modify programs. 	2-CS-02 Design projects that combine hardware and software components to collect and exchange data. 2-AP-16 Incorporate existing code, media, and libraries into original programs, and give attribution.
Communicating with Sounds 45 minutes	<ul style="list-style-type: none"> Describe the function of hardware and software. Program sounds and beeps and learn how to debug a simple program. Create a sound pattern. 	2-CS-02 Design projects that combine hardware and software components to collect and exchange data. 2-AP-13 Decompose problems and subproblems into parts to facilitate the design, implementation, and review of programs. 2-AP-16 Incorporate existing code, media, and libraries into original programs, and give attribution. 2-AP-19 Document programs in order to make them easier to follow, test, and debug.
Digital Sign 45-90 minutes	<ul style="list-style-type: none"> Describe the function of hardware and software. Program lights and sounds to communicate a message. 	2-CS-02 Design projects that combine hardware and software components to collect and exchange data. 2-AP-13 Decompose problems and subproblems into parts to facilitate the design, implementation, and review of programs.

Introduction to Python Programming

Unit 1 Hardware and Software		
Lesson	Objectives	CSTA Standards
		2-AP-16 Incorporate existing code, media, and libraries into original programs, and give attribution. 2-AP-19 Document programs in order to make them easier to follow, test, and debug.
Ideas to Support Your Design 30-45 minutes	<ul style="list-style-type: none"> • Give specific feedback on a peer's project. • Explore how to use feedback to improve a project. 	1B-IC-20 Seek diverse perspectives for the purpose of improving computational artifacts
Career Connections – Lesson Extension 60-90 minutes	<ul style="list-style-type: none"> • Articulate their personal interests and goals. • Relate their personal interests and goals into possible career pathways. • Explore various careers in career pathways. 	Career Ready Practice 10- Plan education and career path aligned to personal goals. (CCTC)

Introduction to Python Programming

Unit 2 Motors		
Lesson	Objectives	CSTA Standards
Making Moves with Motors 45 minutes	<ul style="list-style-type: none"> • Program motors to turn individually using parameters of time and speed • Create a robot dance party 	2-CS-02 Design projects that combine hardware and software components to collect and exchange data. 2-AP-10 Use flowcharts and/or pseudocode to address complex problems as algorithms 2-AP-16 Incorporate existing code, media, and libraries into original programs, and give attribution. 2-AP-17 Systematically test and refine programs using a range of test cases. 2-AP-19 Document programs in order to make them easier to follow, test, and debug.
New Moves with Motors 45 minutes	<ul style="list-style-type: none"> • Program a motor to move to position using the shortest path. • Program a motor to move to a specific position. • Program a motor to move a defined number of degrees. 	2-CS-02 Design projects that combine hardware and software components to collect and exchange data. 2-AP-10 Use flowcharts and/or pseudocode to address complex problems as algorithms 2-AP-16 Incorporate existing code, media, and libraries into original programs, and give attribution. 2-AP-17 Systematically test and refine programs using a range of test cases. 2-AP-19 Document programs in order to make them easier to follow, test, and debug.
Automating Action 45 minutes	<ul style="list-style-type: none"> • Build and program a model that automates a task. 	2-CS-02 Design projects that combine hardware and software components to collect and exchange data. 2-AP-10 Use flowcharts and/or pseudocode to address complex problems as algorithms 2-AP-16 Incorporate existing code, media, and libraries into original programs, and give attribution. 2-AP-17 Systematically test and refine programs using a range of test cases. 2-AP-19 Document programs in order to make them easier to follow, test, and debug.
Hopper Run 45 minutes	<ul style="list-style-type: none"> • Program two motors to move simultaneously. • Build and program a robot without wheels to move forward. 	2-CS-02 Design projects that combine hardware and software components to collect and exchange data. 2-AP-10 Use flowcharts and/or pseudocode to address complex problems as algorithms 2-AP-13 Decompose problems and subproblems into parts to facilitate the design, implementation, and review of programs. 2-AP-16 Incorporate existing code, media, and libraries into original programs, and give attribution.

Introduction to Python Programming

Unit 2 Motors		
Lesson	Objectives	CSTA Standards
		2-AP-17 Systematically test and refine programs using a range of test cases. 2-AP-19 Document programs in order to make them easier to follow, test, and debug.
Race Day 45 minutes	<ul style="list-style-type: none"> • Create a program to move through a series of steps and turns. • Utilize motor pair in multiple ways. 	2-CS-02 Design projects that combine hardware and software components to collect and exchange data. 2-AP-10 Use flowcharts and/or pseudocode to address complex problems as algorithms 2-AP-13 Decompose problems and subproblems into parts to facilitate the design, implementation, and review of programs. 2-AP-16 Incorporate existing code, media, and libraries into original programs, and give attribution. 2-AP-17 Systematically test and refine programs using a range of test cases. 2-AP-19 Document programs in order to make them easier to follow, test, and debug.
Ideas to Help with Race Day 30-45 minutes	<ul style="list-style-type: none"> • Give specific feedback on a peer's project. • Explore how to use feedback to improve a project. 	1B-IC-20 Seek diverse perspectives for the purpose of improving computational artifacts.

Introduction to Python Programming

Unit 3 Sensor Control		
Lesson	Objectives	CSTA Standards
Start Sensing 45 minutes	<ul style="list-style-type: none"> • Program the force sensor. • Create conditional statements. 	2-CS-02 Design projects that combine hardware and software components to collect and exchange data. 2-AP-10 Use flowcharts and/or pseudocode to address complex problems as algorithms 2-AP-16 Incorporate existing code, media, and libraries into original programs, and give attribution. 2-AP-17 Systematically test and refine programs using a range of test cases. 2-AP-19 Document programs in order to make them easier to follow, test, and debug.
Charging Rhino 45 minutes	<ul style="list-style-type: none"> • Explore the force sensor • Understand effects of power on movement 	2-AP-10 Use flowcharts and/or pseudocode to address complex problems as algorithms 2-AP-13 Decompose problems and subproblems into parts to facilitate the design, implementation, and review of programs. 2-AP-16 Incorporate existing code, media, and libraries into original programs, and give attribution. 2-AP-17 Systematically test and refine programs using a range of test cases. 2-AP-19 Document programs in order to make them easier to follow, test, and debug.
Cart Control 45 minutes	<ul style="list-style-type: none"> • Program the distance sensor. • Explore movements with distance. • Understand ultrasonic. 	2-CS-02 Design projects that combine hardware and software components to collect and exchange data. 2-AP-10 Use flowcharts and/or pseudocode to address complex problems as algorithms. 2-AP-13 Decompose problems and subproblems into parts to facilitate the design, implementation, and review of programs. 2-AP-16 Incorporate existing code, media, and libraries into original programs, and give attribution. 2-AP-17 Systematically test and refine programs using a range of test cases. 2-AP-19 Document programs in order to make them easier to follow, test, and debug.
Safe Delivery 45 minutes	<ul style="list-style-type: none"> • Program model to move safely using sensors. • Investigate effects of motor power when using sensors. 	2-CS-02 Design projects that combine hardware and software components to collect and exchange data. 2-AP-10 Use flowcharts and/or pseudocode to address complex problems as algorithms

Introduction to Python Programming

Unit 3 Sensor Control		
Lesson	Objectives	CSTA Standards
		<p>2-AP-13 Decompose problems and subproblems into parts to facilitate the design, implementation, and review of programs.</p> <p>2-AP-16 Incorporate existing code, media, and libraries into original programs, and give attribution.</p> <p>2-AP-17 Systematically test and refine programs using a range of test cases.</p> <p>2-AP-19 Document programs in order to make them easier to follow, test, and debug.</p>
<p>Grasshopper Troubles 45 minutes</p>	<ul style="list-style-type: none"> • Make appropriate hardware decisions • Re-design a model to add a sensor 	<p>2-CS-02 Design projects that combine hardware and software components to collect and exchange data.</p> <p>2-AP-13 Decompose problems and subproblems into parts to facilitate the design, implementation, and review of programs.</p> <p>2-AP-17 Systematically test and refine programs using a range of test cases.</p> <p>2-AP-19 Document programs in order to make them easier to follow, test, and debug.</p>
<p>Ideas to Help Your Grasshopper 30-45 minutes</p>	<ul style="list-style-type: none"> • Give specific feedback on a peer's project. • Explore how to use feedback to improve a project. 	<p>1B-IC-20 Seek diverse perspectives for the purpose of improving computational artifacts.</p>

Introduction to Python Programming

Unit 4 Loops and Variables		
Lesson	Objectives	CSTA Standards
Warm Up Loop with Leo 45 minutes	<ul style="list-style-type: none"> • Program with loops. • Build and program a sit-up machine. 	2-CS-02 Design projects that combine hardware and software components to collect and exchange data. 2-AP-10 Use flowcharts and/or pseudocode to address complex problems as algorithms 2-AP-13 Decompose problems and subproblems into parts to facilitate the design, implementation, and review of programs. 2-AP-16 Incorporate existing code, media, and libraries into original programs, and give attribution. 2-AP-17 Systematically test and refine programs using a range of test cases. 2-AP-19 Document programs in order to make them easier to follow, test, and debug.
Counting Reps with Leo 45 minutes	<ul style="list-style-type: none"> • Program a sit-up machine to count the reps and to complete a count down. 	2-CS-02 Design projects that combine hardware and software components to collect and exchange data. 2-AP-10 Use flowcharts and/or pseudocode to address complex problems as algorithms 2-AP-13 Decompose problems and subproblems into parts to facilitate the design, implementation, and review of programs. 2-AP-16 Incorporate existing code, media, and libraries into original programs, and give attribution. 2-AP-17 Systematically test and refine programs using a range of test cases. 2-AP-19 Document programs in order to make them easier to follow, test, and debug.

Introduction to Python Programming

Unit 4 Loops and Variables		
Lesson	Objectives	CSTA Standards
Dance Loop with Coach 45 minutes	<ul style="list-style-type: none"> • Program a model using for loops. • Debug four programs to learn tips and tricks. 	2-CS-02 Design projects that combine hardware and software components to collect and exchange data. 2-AP-10 Use flowcharts and/or pseudocode to address complex problems as algorithms 2-AP-13 Decompose problems and subproblems into parts to facilitate the design, implementation, and review of programs. 2-AP-16 Incorporate existing code, media, and libraries into original programs, and give attribution. 2-AP-17 Systematically test and refine programs using a range of test cases. 2-AP-19 Document programs in order to make them easier to follow, test, and debug.
Setting Conditions for Yoga 45-90 minutes	<ul style="list-style-type: none"> • Investigate while statements. • Program a model using while loops. 	2-CS-02 Design projects that combine hardware and software components to collect and exchange data. 2-AP-10 Use flowcharts and/or pseudocode to address complex problems as algorithms 2-AP-13 Decompose problems and subproblems into parts to facilitate the design, implementation, and review of programs. 2-AP-16 Incorporate existing code, media, and libraries into original programs, and give attribution. 2-AP-17 Systematically test and refine programs using a range of test cases. 2-AP-19 Document programs in order to make them easier to follow, test, and debug.

Introduction to Python Programming

Unit 4 Loops and Variables		
Lesson	Objectives	CSTA Standards
Infinite Moves 45-90 minutes	<ul style="list-style-type: none"> • Program infinite loops. • Create a model that includes a force sensor that will provide a condition for the robot to move. 	2-CS-02 Design projects that combine hardware and software components to collect and exchange data. 2-AP-10 Use flowcharts and/or pseudocode to address complex problems as algorithms 2-AP-13 Decompose problems and subproblems into parts to facilitate the design, implementation, and review of programs. 2-AP-16 Incorporate existing code, media, and libraries into original programs, and give attribution. 2-AP-17 Systematically test and refine programs using a range of test cases. 2-AP-19 Document programs in order to make them easier to follow, test, and debug.
Leading the Team with Loops 90-120 minutes	<ul style="list-style-type: none"> • Design a model for repetition. • Program a model to move using loops. 	2-CS-02 Design projects that combine hardware and software components to collect and exchange data. 2-AP-13 Decompose problems and subproblems into parts to facilitate the design, implementation, and review of programs. 2-AP-16 Incorporate existing code, media, and libraries into original programs, and give attribution.
Ideas to Help with Leading the Team with Loops 30-45 minutes	<ul style="list-style-type: none"> • Give specific feedback on a peer's project. • Explore how to use feedback to improve a project. 	1B-IC-20 Seek diverse perspectives for the purpose of improving computational artifacts.

Introduction to Python Programming

Unit 5 Conditions for Games		
Lesson	Objectives	CSTA Standards
Controlling Motion with Tilt 45 minutes	<ul style="list-style-type: none"> • Program the motion sensor. • Create conditional statements. 	2-CS-02 Design projects that combine hardware and software components to collect and exchange data. 2-AP-10 Use flowcharts and/or pseudocode to address complex problems as algorithms 2-AP-16 Incorporate existing code, media, and libraries into original programs, and give attribution. 2-AP-17 Systematically test and refine programs using a range of test cases. 2-AP-19 Document programs in order to make them easier to follow, test, and debug.
Claw Machine 45 minutes	<ul style="list-style-type: none"> • Create a basic loop. • Program a grabber model based on set conditions. 	2-CS-02 Design projects that combine hardware and software components to collect and exchange data. 2-AP-10 Use flowcharts and/or pseudocode to address complex problems as algorithms 2-AP-13 Decompose problems and subproblems into parts to facilitate the design, implementation, and review of programs. 2-AP-16 Incorporate existing code, media, and libraries into original programs, and give attribution. 2-AP-17 Systematically test and refine programs using a range of test cases. 2-AP-19 Document programs in order to make them easier to follow, test, and debug.
Charting Game Decisions 45-90 minutes	<ul style="list-style-type: none"> • Understand how to use flowcharts in planning. • Create flowcharts and write programs that follow them. 	2-CS-02 Design projects that combine hardware and software components to collect and exchange data. 2-AP-10 Use flowcharts and/or pseudocode to address complex problems as algorithms 2-AP-13 Decompose problems and subproblems into parts to facilitate the design, implementation, and review of programs. 2-AP-16 Incorporate existing code, media, and libraries into original programs, and give attribution. 2-AP-17 Systematically test and refine programs using a range of test cases. 2-AP-19 Document programs in order to make them easier to follow, test, and debug.
Guess Which Color 45 minutes	<ul style="list-style-type: none"> • Program the color sensor using conditional code. 	2-CS-02 Design projects that combine hardware and software components to collect and exchange data.

Introduction to Python Programming

Unit 5 Conditions for Games		
Lesson	Objectives	CSTA Standards
	<ul style="list-style-type: none"> • Create a game. 	2-AP-10 Use flowcharts and/or pseudocode to address complex problems as algorithms 2-AP-13 Decompose problems and subproblems into parts to facilitate the design, implementation, and review of programs. 2-AP-16 Incorporate existing code, media, and libraries into original programs, and give attribution. 2-AP-17 Systematically test and refine programs using a range of test cases. 2-AP-19 Document programs in order to make them easier to follow, test, and debug.
Guessing Game 45 minutes	<ul style="list-style-type: none"> • Write code that uses multiple condition statements using if/elif/else programming. • Add a loop to code. • Debug coding that has incorrect/missing syntax, missing code, or incorrect indentation. 	2-CS-02 Design projects that combine hardware and software components to collect and exchange data. 2-AP-10 Use flowcharts and/or pseudocode to address complex problems as algorithms 2-AP-13 Decompose problems and subproblems into parts to facilitate the design, implementation, and review of programs. 2-AP-16 Incorporate existing code, media, and libraries into original programs, and give attribution. 2-AP-17 Systematically test and refine programs using a range of test cases. 2-AP-19 Document programs in order to make
Score! 45 minutes	<ul style="list-style-type: none"> • Program movement and light matrix. • Apply knowledge of conditional statements. 	2-CS-02 Design projects that combine hardware and software components to collect and exchange data. 2-AP-10 Use flowcharts and/or pseudocode to address complex problems as algorithms 2-AP-13 Decompose problems and subproblems into parts to facilitate the design, implementation, and review of programs. 2-AP-16 Incorporate existing code, media, and libraries into original programs, and give attribution. 2-AP-17 Systematically test and refine programs using a range of test cases. 2-AP-19 Document programs in order to make them easier to follow, test, and debug.

Introduction to Python Programming

Unit 5 Conditions for Games		
Lesson	Objectives	CSTA Standards
Game Time 90 minutes	<ul style="list-style-type: none"> • Write code that includes conditions that must be met in a game format • Create a game that requires a series of events requiring a robot to respond 	2-CS-02 Design projects that combine hardware and software components to collect and exchange data. 2-AP-10 Use flowcharts and/or pseudocode to address complex problems as algorithms 2-AP-13 Decompose problems and subproblems into parts to facilitate the design, implementation, and review of programs. 2-AP-16 Incorporate existing code, media, and libraries into original programs, and give attribution. 2-AP-17 Systematically test and refine programs using a range of test cases. 2-AP-19 Document programs in order to make them easier to follow, test, and debug. 2-IC-22 Collaborate with many contributors through strategies such as crowdsourcing or surveys when creating a computational artifact.
Ideas to Help with Game Time 30-45 minutes	<ul style="list-style-type: none"> • Give specific feedback on a peer's project. • Explore how to use feedback to improve a project. 	1B-IC-20 Seek diverse perspectives for the purpose of improving computational artifacts.

Introduction to Python Programming

Unit 6 Troubleshooting and Debugging		
Lesson	Objectives	CSTA Standards
Testing Prototypes 45 minutes	<ul style="list-style-type: none"> Brainstorm ideas and develop solutions to a problem. Program a model. 	2-AP-15 Seek and incorporate feedback from team members and users to refine a solution that meets user needs. NGSS MS-ETS1-2. Evaluate competing design solutions using a systematic process to determine how well they meet the criteria and constraints of the problem. MS-ETS1-4 Develop a model to generate data for iterative testing and modification of a proposed object, tool or process such that an optimal design can be achieved.
Break Dancer Break Down 45 minutes	<ul style="list-style-type: none"> Identify a problem and debug the program. 	2-AP-10 Use flowcharts and/or pseudocode to address complex problems as algorithms. 2-AP-13 Decompose problems and subproblems into parts to facilitate the design, implementation, and review of programs. 2-AP-17 Systematically test and refine programs using a range of test cases. 2-AP-19 Document programs in order to make them easier to follow, test, and debug.
Dance to the Beat 45 minutes	<ul style="list-style-type: none"> Identify a problem and debug the program. 	2-AP-10 Use flowcharts and/or pseudocode to address complex problems as algorithms. 2-AP-13 Decompose problems and subproblems into parts to facilitate the design, implementation, and review of programs. 2-AP-17 Systematically test and refine programs using a range of test cases. 2-AP-19 Document programs in order to make them easier to follow, test, and debug.
Testing for Trouble 90 minutes	<ul style="list-style-type: none"> Identify and repair a hardware problem in a design. 	2-AP-10 Use flowcharts and/or pseudocode to address complex problems as algorithms. 2-AP-13 Decompose problems and subproblems into parts to facilitate the design, implementation, and review of programs. 2-AP-17 Systematically test and refine programs using a range of test cases. 2-AP-19 Document programs in order to make them easier to follow, test, and debug.
Debug-inator 45 minutes	<ul style="list-style-type: none"> Debug a software problem. 	2-AP-10 Use flowcharts and/or pseudocode to address complex problems as algorithms. 2-AP-13 Decompose problems and subproblems into parts to facilitate the design, implementation, and review of programs.

Introduction to Python Programming

Unit 6 Troubleshooting and Debugging		
Lesson	Objectives	CSTA Standards
		2-AP-17 Systematically test and refine programs using a range of test cases. 2-AP-19 Document programs in order to make them easier to follow, test, and debug.
Ideas to Help with the Debug-inator 30-45 minutes	<ul style="list-style-type: none">• Give specific feedback on a peer's project.• Explore how to use feedback to improve a project.	1B-IC-20 Seek diverse perspectives for the purpose of improving computational artifacts.

Introduction to Python Programming

Unit 7 Functions		
Lesson	Objectives	CSTA Standards
Turtle Trouble 45 minutes	<ul style="list-style-type: none"> • Write a program that will make the turtle's flippers move • Modify a program to allow different reactions or lines of code to run based on the situation 	2-CS-02 Design projects that combine hardware and software components to collect and exchange data. 2-AP-13 Decompose problems and subproblems into parts to facilitate the design, implementation, and review of programs. 2-AP-16 Incorporate existing code, media, and libraries into original programs, and give attribution. 2-AP-17 Systematically test and refine programs using a range of test cases. 2-AP-19 Document programs in order to make them easier to follow, test, and debug.
Clean Up with Multiple Functions 45 minutes	<ul style="list-style-type: none"> • Build and program a grabber to pick up items • Modify the program to include multiple functions 	2-CS-02 Design projects that combine hardware and software components to collect and exchange data. 2-AP-13 Decompose problems and subproblems into parts to facilitate the design, implementation, and review of programs. 2-AP-17 Systematically test and refine programs using a range of test cases.
Clean Indicator 45 minutes	<ul style="list-style-type: none"> • Create functions that use parameters • Investigate debugging functions and functions that use parameters 	2-CS-02 Design projects that combine hardware and software components to collect and exchange data. 2-AP-10 Use flowcharts and/or pseudocode to address complex problems as algorithms 2-AP-13 Decompose problems and subproblems into parts to facilitate the design, implementation, and review of programs. 2-AP-16 Incorporate existing code, media, and libraries into original programs, and give attribution. 2-AP-17 Systematically test and refine programs using a range of test cases. 2-AP-19 Document programs in order to make them easier to follow, test, and debug.
Automate the Clean Up 90 minutes	<ul style="list-style-type: none"> • Program a sorting robot to identify if a material is recyclable or non-recyclable • Incorporate a second function into a program to make the program more efficient 	2-CS-02 Design projects that combine hardware and software components to collect and exchange data. 2-AP-13 Decompose problems and subproblems into parts to facilitate the design, implementation, and review of programs. 2-AP-14 Create procedures with parameters to organize code and make it easier to reuse.

Introduction to Python Programming

Unit 7 Functions		
Lesson	Objectives	CSTA Standards
		2-AP-17 Systematically test and refine programs using a range of test cases. 2-AP-19 Document programs in order to make them easier to follow, test, and debug.
Taking Care of My Environment 90 minutes	<ul style="list-style-type: none"> Design, build, and program an environmental helper to take care of a local area 	2-CS-02 Design projects that combine hardware and software components to collect and exchange data. 2-AP-10 Use flowcharts and/or pseudocode to address complex problems as algorithms 2-AP-13 Decompose problems and subproblems into parts to facilitate the design, implementation, and review of programs. 2-AP-14 Create procedures with parameters to organize code and make it easier to reuse 2-AP-16 Incorporate existing code, media, and libraries into original programs, and give attribution. 2-AP-17 Systematically test and refine programs using a range of test cases. 2-AP-19 Document programs in order to make them easier to follow, test, and debug.
Ideas to Help with Taking Care of My Environment 30-45 minutes	<ul style="list-style-type: none"> Give specific feedback on a peer's project. Explore how to use feedback to improve a project. 	1B-IC-20 Seek diverse perspectives for the purpose of improving computational artifacts.

Introduction to Python Programming

Unit 8 Compound Conditionals and Logic Operators		
Lesson	Objectives	CSTA Standards
Password Protection 45 minutes	<ul style="list-style-type: none"> Investigate cyber security through setting passwords Explore physical security measures 	2-CS-02 Design projects that combine hardware and software components to collect and exchange data. 2-NI-05 Explain how physical and digital security measures protect electronic information. 2-AP-13 Decompose problems and subproblems into parts to facilitate the design, implementation, and review of programs. 2-AP-16 Incorporate existing code, media, and libraries into original programs, and give attribution. 2-AP-19 Document programs in order to make them easier to follow, test, and debug.
Make it Physically Safe 45-90 minutes	<ul style="list-style-type: none"> Investigate nested conditional statements Explore physical security measures 	2-NI-05 Explain how physical and digital security measures protect electronic information. 2-CS-02 Design projects that combine hardware and software components to collect and exchange data. 2-AP-10 Use flowcharts and/or pseudocode to address complex problems as algorithms 2-AP-13 Decompose problems and subproblems into parts to facilitate the design, implementation, and review of programs. 2-AP-16 Incorporate existing code, media, and libraries into original programs, and give attribution. 2-AP-17 Systematically test and refine programs using a range of test cases. 2-AP-19 Document programs in order to make them easier to follow, test, and debug.

Introduction to Python Programming

Unit 8 Compound Conditionals and Logic Operators		
Lesson	Objectives	CSTA Standards
Make a Safer Safe 45-90 minutes	<ul style="list-style-type: none"> Investigate using logic operators to combine conditions Explore physical security measures 	2-NI-05 Explain how physical and digital security measures protect electronic information. 2-CS-02 Design projects that combine hardware and software components to collect and exchange data. 2-AP-10 Use flowcharts and/or pseudocode to address complex problems as algorithms 2-AP-13 Decompose problems and subproblems into parts to facilitate the design, implementation, and review of programs. 2-AP-16 Incorporate existing code, media, and libraries into original programs, and give attribution. 2-AP-17 Systematically test and refine programs using a range of test cases. 2-AP-19 Document programs in order to make them easier to follow, test, and debug.
Security Operating with Logic 45-90 minutes	<ul style="list-style-type: none"> Investigate using sensors for security Create two-step security programs 	2-CS-02 Design projects that combine hardware and software components to collect and exchange data. 2-AP-10 Use flowcharts and/or pseudocode to address complex problems as algorithms 2-AP-13 Decompose problems and subproblems into parts to facilitate the design, implementation, and review of programs. 2-AP-16 Incorporate existing code, media, and libraries into original programs, and give attribution. 2-AP-17 Systematically test and refine programs using a range of test cases. 2-AP-19 Document programs in order to make them easier to follow, test, and debug.

Introduction to Python Programming

Unit 8 Compound Conditionals and Logic Operators		
Lesson	Objectives	CSTA Standards
Escape Room 90 minutes	<ul style="list-style-type: none"> • Create a security device to simulate a break out game • Design a device that meets given constraints 	2-CS-02 Design projects that combine hardware and software components to collect and exchange data. 2-AP-10 Use flowcharts and/or pseudocode to address complex problems as algorithms 2-AP-13 Decompose problems and subproblems into parts to facilitate the design, implementation, and review of programs. 2-AP-16 Incorporate existing code, media, and libraries into original programs, and give attribution. 2-AP-17 Systematically test and refine programs using a range of test cases. 2-AP-19 Document programs in order to make them easier to follow, test, and debug. 2-IC-22 Collaborate with many contributors through strategies such as crowdsourcing or surveys when creating a computational artifact.
Ideas to Help with Escape Room 30-45 minutes	<ul style="list-style-type: none"> • Give specific feedback on a peer's project. • Explore how to use feedback to improve a project. 	1B-IC-20 Seek diverse perspectives for the purpose of improving computational artifacts.

Introduction to Python Programming

Unit 9 Data and Math Functions		
Lesson	Objectives	CSTA Standards
Get Moving to Get Data 45 minutes	<ul style="list-style-type: none"> Investigate ways to take in data from sensors Create a new program that will provide data using the force sensor 	2-CS-02 Design projects that combine hardware and software components to collect and exchange data. 2-AP-10 Use flowcharts and/or pseudocode to address complex problems as algorithms 2-AP-13 Decompose problems and subproblems into parts to facilitate the design, implementation, and review of programs. 2-AP-16 Incorporate existing code, media, and libraries into original programs, and give attribution. 2-AP-17 Systematically test and refine programs using a range of test cases. 2-AP-19 Document programs in order to make them easier to follow, test, and debug.
Bike Riding for Data 45 minutes	<ul style="list-style-type: none"> Program a bike model to move forward at a constant speed Create a program increase and decrease the speed of the bike model using math functions 	2-CS-02 Design projects that combine hardware and software components to collect and exchange data. 2-AP-10 Use flowcharts and/or pseudocode to address complex problems as algorithms 2-AP-13 Decompose problems and subproblems into parts to facilitate the design, implementation, and review of programs. 2-AP-16 Incorporate existing code, media, and libraries into original programs, and give attribution. 2-AP-17 Systematically test and refine programs using a range of test cases. 2-AP-19 Document programs in order to make them easier to follow, test, and debug.

Introduction to Python Programming

Unit 9 Data and Math Functions		
Lesson	Objectives	CSTA Standards
Counting Your Steps 45 minutes	<ul style="list-style-type: none"> Integrate mathematical calculations into their programs using variables Create a program that will measure footsteps 	2-CS-02 Design projects that combine hardware and software components to collect and exchange data. 2-AP-10 Use flowcharts and/or pseudocode to address complex problems as algorithms 2-AP-13 Decompose problems and subproblems into parts to facilitate the design, implementation, and review of programs. 2-AP-16 Incorporate existing code, media, and libraries into original programs, and give attribution. 2-AP-17 Systematically test and refine programs using a range of test cases. 2-AP-19 Document programs in order to make them easier to follow, test, and debug.
Make It Move 45 minutes	<ul style="list-style-type: none"> Program a driving base to move forward and change directions Create a program with mathematical functions 	2-CS-02 Design projects that combine hardware and software components to collect and exchange data. 2-AP-10 Use flowcharts and/or pseudocode to address complex problems as algorithms 2-AP-13 Decompose problems and subproblems into parts to facilitate the design, implementation, and review of programs. 2-AP-16 Incorporate existing code, media, and libraries into original programs, and give attribution. 2-AP-17 Systematically test and refine programs using a range of test cases. 2-AP-19 Document programs in order to make them easier to follow, test, and debug.
Parking Lot 90 minutes	<ul style="list-style-type: none"> Use conditional statements in a program using sensors and motors. Apply sensors to real-life problems. 	2-AP-18 Distribute tasks and maintain a project timeline when collaboratively developing computational artifacts. 2-AP-12 Design and iteratively develop programs that combine control structures, including nested loops and compound conditionals. 2-CS-02 Design projects that combine hardware and software components to collect and exchange data. 2-AP-10 Use flowcharts and/or pseudocode to address complex problems as algorithms. 2-AP-13 Decompose problems and subproblems into parts to facilitate the design, implementation, and review of programs.

Introduction to Python Programming

Unit 9 Data and Math Functions		
Lesson	Objectives	CSTA Standards
My Transportation 90 minutes	<ul style="list-style-type: none"> Design, build, and program a transportation vehicle to bring them to school 	2-CS-02 Design projects that combine hardware and software components to collect and exchange data. 2-AP-10 Use flowcharts and/or pseudocode to address complex problems as algorithms 2-AP-13 Decompose problems and subproblems into parts to facilitate the design, implementation, and review of programs. 2-AP-16 Incorporate existing code, media, and libraries into original programs, and give attribution. 2-AP-17 Systematically test and refine programs using a range of test cases. 2-AP-19 Document programs in order to make them easier to follow, test, and debug. 2-IC-22 Collaborate with many contributors through strategies such as crowdsourcing or surveys when creating a computational artifact.
Ideas to Help with My Transportation 30-45 minutes	<ul style="list-style-type: none"> Give specific feedback on a peer's project. Explore how to use feedback to improve a project. 	1B-IC-20 Seek diverse perspectives for the purpose of improving computational artifacts.

Introduction to Python Programming

Unit 10 Lists		
Lesson	Objectives	CSTA Standards
Listing Letters 45 minutes	<ul style="list-style-type: none"> • Create and utilize lists. • Code with compound conditionals using lists. 	2-CS-01 Recommend improvements to the design of computing devices, based on an analysis of how users interact with the devices. 2-DA-09 Refine computational models based on the data they have generated. 2-AP-10 Use flowcharts and/or pseudocode to address complex problems as algorithms. 2-AP-12 Design and iteratively develop programs that combine control structures, including nested loops and compound conditionals. 2-AP-18 Distribute tasks and maintain a project timeline when collaboratively developing computational artifacts. 2-AP-16 Incorporate existing code, media, and libraries into original programs, and give attribution. 2-CS-02 Design projects that combine hardware and software components to collect and exchange data. 2-AP-13 Decompose problems and subproblems into parts to facilitate the design, implementation, and review of programs.
Stretch Your Muscles and Lists 90 minutes	<ul style="list-style-type: none"> • Create a program using values for the motion sensor as the variables in their list • Use a list to create a yoga routine 	2-CS-02 Design projects that combine hardware and software components to collect and exchange data. 2-AP-10 Use flowcharts and/or pseudocode to address complex problems as algorithms 2-AP-13 Decompose problems and subproblems into parts to facilitate the design, implementation, and review of programs. 2-AP-17 Systematically test and refine programs using a range of test cases. 2-AP-19 Document programs in order to make them easier to follow, test, and debug.
Mind Games 90 minutes	<ul style="list-style-type: none"> • Create two lists in one program • Compare two lists within the program 	2-CS-02 Design projects that combine hardware and software components to collect and exchange data. 2-AP-10 Use flowcharts and/or pseudocode to address complex problems as algorithms 2-AP-13 Decompose problems and subproblems into parts to facilitate the design, implementation, and review of programs. 2-AP-16 Incorporate existing code, media, and libraries into original programs, and give attribution.

Introduction to Python Programming

Unit 10 Lists		
Lesson	Objectives	CSTA Standards
		2-AP-17 Systematically test and refine programs using a range of test cases. 2-AP-19 Document programs in order to make them easier to follow, test, and debug.
Jumping for Lists 90 minutes	<ul style="list-style-type: none"> • Create data from the force and distance sensors to use in a list • Program a list based on the data gathered from the jumping trials (i.e., height of jumps) 	2-CS-02 Design projects that combine hardware and software components to collect and exchange data. 2-AP-10 Use flowcharts and/or pseudocode to address complex problems as algorithms 2-AP-11 Create clearly named variables that represent different data types and perform operations on their values. 2-AP-13 Decompose problems and subproblems into parts to facilitate the design, implementation, and review of programs. 2-AP-16 Incorporate existing code, media, and libraries into original programs, and give attribution. 2-AP-17 Systematically test and refine programs using a range of test cases. 2-AP-19 Document programs in order to make them easier to follow, test, and debug.
Word Games with Lists 90 minutes	<ul style="list-style-type: none"> • Create multiple lists within a program to complete a word game • Program a color sensing model to coordinate with their word game 	2-CS-02 Design projects that combine hardware and software components to collect and exchange data. 2-AP-10 Use flowcharts and/or pseudocode to address complex problems as algorithms 2-AP-13 Decompose problems and subproblems into parts to facilitate the design, implementation, and review of programs. 2-AP-17 Systematically test and refine programs using a range of test cases. 2-AP-19 Document programs in order to make them easier to follow, test, and debug.
Ideas to Help with Word Games with Lists 30-45 minutes	<ul style="list-style-type: none"> • Give specific feedback on a peer's project. • Explore how to use feedback to improve a project. 	1B-IC-20 Seek diverse perspectives for the purpose of improving computational artifacts.

Unit 1: Hardware and Software

A LEGO® Education Unit

Unit Introduction

This introductory unit allows students to explore essential computer science principles and programming concepts of the text-based coding language, Python, to communicate in a variety of ways. Students will investigate combining hardware and software, with a focus on pair programming. The lessons are designed in an order that allows students to progress in their skills and knowledge in the following areas:

- Importing Python libraries and running programs
- Describe the function of hardware and software
- Practice pair programming and working in programming roles
- Program lights and sounds to communicate messages
- Decompose problems and utilize debugging strategies
- Provide specific feedback to others
- Utilize feedback to improve a project

Unit Learning Promise

In this unit, your students will explore how hardware and software work together while investigating ways to communicate ideas using lights and sounds. Additionally, students will focus on how to collaborate with their peers to participate in pair programming as well as how to give and receive feedback to improve the overall quality of a project.

Investigation Questions:

What are ways to communicate ideas to others? How can hardware and software work together to help us solve problems?

Unit Lessons

Lesson 1	Lesson 2	Lesson 3	Lesson 4	Lesson 5	Lesson 6	Lesson 7
Importing Libraries	Communicating with Lights	Pair Programming	Communicating with Sounds	Digital Sign	Ideas that Support Your Design	Career Connections
Time: 45 Min.	Time: 45 Min.	Time: 45 Min.	Time: 45 Min.	Time: 45-90 Min.	Time: 45 Min.	Time: 45 Min.

Assessment: We recommend assessing students on various skills throughout the unit.

- Use the progression of lessons as an opportunity to provide on-going feedback to prepare students for success for the open-ended project at the end of the unit.
- Each lesson includes a recommendation for teacher observations, student self-assessment, evaluation of success.

Unit Standards

CSTA
2-CS-02 Design projects that combine hardware and software components to collect and exchange data.
2-AP-13 Decompose problems and subproblems into parts to facilitate the design, implementation, and review of programs.
2-AP-16 Incorporate existing code, media, and libraries into original programs, and give attribution.
2-AP-19 Document programs in order to make them easier to follow, test, and debug.
1B-IC-20 Seek diverse perspectives for the purpose of improving computational artifacts.
2-CS-02 Design projects that combine hardware and software components to collect and exchange data.

Integrated Standards

NGSS		
MS-ETS1-2 Evaluate competing design solutions using a systematic process to determine how well they meet the criteria and constraints of the problem.		
Common Core English Language Arts (ELA)		
6 th Grade	7 th Grade	8 th Grade
SL.6.1 Engage effectively in a range of collaborative discussions (one-on-one, in groups, and teacher-led) with diverse partners on grade 6 topics, texts, and issues, building on others' ideas and expressing their own clearly	SL.7.1 Engage effectively in a range of collaborative discussions (one-on-one, in groups, and teacher-led) with diverse partners on grade 7 topics, texts, and issues, building on others' ideas and expressing their own clearly	SL.8.1 Engage effectively in a range of collaborative discussions (one-on-one, in groups, and teacher-led) with diverse partners on grade 8 topics, texts, and issues, building on others' ideas and expressing their own clearly
SL.6.2 Interpret information presented in diverse media and formats (e.g., visually, quantitatively, orally) and explain how it contributes to a topic, text, or issue under study	SL.7.2 Analyze the main ideas and supporting details presented in diverse media and formats (e.g., visually, quantitatively, orally) and explain how the ideas clarify a topic, text, or issue under study	SL.8.2 Analyze the purpose of information presented in diverse media and formats (e.g., visually, quantitatively, orally) and evaluate the motives (e.g., social, commercial, political) behind its presentation
SL.6.4 Present claims and findings, sequencing ideas logically and using pertinent descriptions, facts, and details to accentuate main ideas or themes; use appropriate eye contact, adequate volume, and clear pronunciation	SL.7.4 Present claims and findings, emphasizing salient points in a focused, coherent manner with pertinent descriptions, facts, details, and examples; use appropriate eye contact, adequate volume, and clear pronunciation	SL.8.4 Present claims and findings, emphasizing salient points in a focused, coherent manner with relevant evidence, sound valid reasoning, and well-chosen details; use appropriate eye contact, adequate volume, and clear pronunciation
RST.6-8.3 Follow precisely a multistep procedure when carrying out experiments, taking measurements, or performing technical tasks	RST.6-8.3 Follow precisely a multistep procedure when carrying out experiments, taking measurements, or performing technical tasks	<u>RST.6-8.3</u> Follow precisely a multistep procedure when carrying out experiments, taking measurements, or performing technical tasks
<u>L.6.6</u> Acquire and use accurately grade-appropriate general academic and domain-specific words and phrases; gather vocabulary knowledge when	L.7.6 Acquire and use accurately grade-appropriate general academic and domain-specific words and phrases; gather vocabulary knowledge when	L.8.6 Acquire and use accurately grade-appropriate general academic and domain-specific words and phrases; gather vocabulary knowledge when

considering a word or phrase important to comprehension or expression

considering a word or phrase important to comprehension or expression

considering a word or phrase important to comprehension or expression

Importing Libraries

Grade 6-9

45 minutes

Beginner

Importing Libraries

Students learn how to import libraries in Python and why this is important when connecting software and hardware.

Questions to investigate

- How can engineers and computer programmers work together to create a way to communicate ideas to others?

Materials needed

- SPIKE Prime sets ready for student use. Prior to the first lesson, please visit the following website for help with set up, kit organization and SPIKE App <https://education.lego.com/en-us/start/spike-prime/intro>
- Devices with the SPIKE App installed
- Student journals

Prepare

- Determine how students will be grouped (2 students work with one SPIKE Prime set).
- Determine your student expectations for teamwork.
- Check to make sure SPIKE Prime hubs are charged, especially if connecting through Bluetooth.

1. Engage

Ignite a discussion with students. Think about what is needed when you follow a recipe to cook. You can go to the store to purchase the ingredients you need. When you get home, you need to be sure you have the tools you need to prepare the food. Think about what you might need. You might need a stove or an oven. You might need a mixer and a spatula.

When writing a Python program, you will need to make sure you "gather" all the things you will use before you start to make a model run.

KEY OBJECTIVES

Students will:

- Learn why a Python program must have libraries imported.
- Import libraries.
- Run a program.

STANDARDS

CSTA

2-CS-02 Design projects that combine hardware and software components to collect and exchange data.
2-AP-16 Incorporate existing code, media, and libraries into original programs, and give attribution.

VOCABULARY

Library
Import

2. Explore

Students will learn what a library is and how and why you import libraries.

When using Python to program, students must import the libraries, the words, or terms, that signal the program to recognize the hardware that can be used. Because Python is a text-based coding language, capitalization and punctuation are important. For the SPIKE App to communicate with the SPIKE Prime components, you will need the right libraries that link the hardware components to the software.

Prompt students to open their SPIKE Prime sets and locate all the hardware pieces. Pull each hardware item out of the set and lay them on the table (1 Hub, 3 motors, 3 sensors).

Ask students to identify each piece of hardware.

Review each hardware item. Hold up each hardware piece and ask students to locate the same piece. Turn on the hub by pressing the large center button.

Ask students how they think they would import each piece of hardware into the software.

Show the introductory program in the SPIKE App Python canvas.

```
from hub import light_matrix
import runloop

async def main():
    # write your code here
    await light_matrix.write("Hi!")

runloop.run(main())
```

Introduce students to the line for importing a hardware piece which is called a library.

- Ask students to identify the hardware that matches the items in the import code
 - import _____
 - the blank will be filled in with the hardware item.

Note:

- light_matrix is the 5x5 grid on the hub

- Button refers to the three buttons located at the bottom on the front of the hub - left, center and right buttons
- light is the light color surrounding the center button
- force_sensor refers to the force sensor which contains a black button to push in
- motion_sensor is contained within the hub, like a gyro
- speaker is on the outside of the hub
- color_sensor is the small square color sensor that has one light
- App is the ability to play sounds
- distance_sensor is the rectangular ultrasonic sensor that appears to have two "eyes"
- motor refers to any size motor
- motor_pair refers to two motors that will work in tandem.

Have students return all materials to the SPIKE Prime set and correctly place them back into the proper areas.

3. Explain

Ask students questions like:

- Why do you think it is important to import a library before you begin writing code?
- What can be imported for use with the hub? How are they imported?
- How are motors imported?

4. Elaborate

Students will practice importing libraries and see what importing the different types of libraries looks like.

Ask students to open a new project in their SPIKE App, selecting Python as the programming type. Students should connect to their hub.

Students will see the program already written in the canvas.

```
from hub import light_matrix
```

```
import runloop
```

```
async def main():
```

```
# write your code here
```

```
await light_matrix.write("Hi!")
```

```
runloop.run(main())
```

Have students look carefully at which pieces of hardware are available to use in this code. Discuss the imported libraries with students.

Allow students to run the sample code by selecting the play button (yellow circle with white triangle) at the bottom of the screen. The light matrix should scroll Hi! Across the display. **Tips for troubleshooting:** Ensure the hub is connected properly. On the programming canvas in the upper left corner is an icon of the hub. If a green ring is around the hub icon, then it is connected with the USB cable. If a blue ring is around the hub icon, then it is connected with Bluetooth. If the hub icon is yellow, then the hub is disconnected.

Start Importing

To start learning how to import libraries, ask students to locate the Knowledge Base on the right-side panel.

Look at the code on screen. What libraries have been imported? Have students reference line 1 and 2 in the code.

Ask students to refer to line 6 of the code. What is being used during programming?

- What do you think the program is going to do? Talk with your partner and then start your program.

Refer students to the Getting Started section of the Knowledge Base. Read 1. Introduction to Python and then review the importing libraries information with students.

- Ask them what it means when they read, "The imported libraries are located at the beginning of the .py file and should appear only once in the program."
- Students may not know what a .py file is. The .py file is the Python program that is being coded. ".py" stands for Python.
- You do not see .py anywhere on the canvas because the .py file contains everything on the canvas.
- The name of the file or project is listed at the top left of the screen next to the picture of a house. If this is the first project created, the name of the Project is likely Project 1.

Ask students to open a new project using Python. Have students try different phrases to see how the program changes.

5. Evaluate

Teacher Observation:

Discuss the program with students.

Ask students questions like:

- What happened on your hub when you ran the program?
- What libraries did you use to run this program?
- Why do you need to import different parts of the hub?
- How can engineers and computer programmers work together to create a way to communicate ideas to others?

Remind students that they are responsible for materials management. Parts should not be shared between sets. If a part is missing, ask the teacher. Remember, the teacher has limited spare parts. Let the teacher know immediately if you cannot locate something.

Each day, students will make a journal entry about the materials management of their set. They grade themselves using a three-point scale. The goal is for all team members to obtain 3 points. Listed below is the self-scoring guide.

- 1) Materials are not all located in their correct tray; some parts are still together.
- 2) Materials are located correctly, but only one person helped put things away.
- 3) Both partners worked together and all parts in the correct locations.

Self-Assessment:

Have students answer the following in their journals:

- Why do you need to import libraries at the beginning of a Python program?
- What characteristics of a good teammate did I display today?
- Ask students to rate themselves on a scale of 1-3, on their time management today.
- Ask students to rate themselves on a scale of 1-3, on their materials (parts) management today.

Communicating with Light

Grade 6-9

45 minutes

Beginner

Communicating with Light

Students learn how to control the light matrix to show images and write words.

Questions to investigate

- How can technology help us solve problems?
- How can you plan for the use of hardware in a solution to a problem?

Materials needed

- SPIKE Prime sets
- Device with SPIKE App installed
- Student journal
- Sticky notes
- Scissors
- Colored paper

Prepare

Check to make sure SPIKE Prime hubs are charged, especially if connecting through Bluetooth.

1. Engage

What shapes can you create in a 5x5 grid? Could you make a rectangle? A triangle? What types of faces could you make?

Why would you create on paper before you code?

Students will investigate programming lights on the hub.

- Have students draw a 5x5 matrix in their journals.
- Ask students to cut 25 small squares of colored paper to fit inside their matrix.
- Draw a simple image or show a picture to the students that could be created on the light matrix on the hub. Examples could include a square or heart.

KEY OBJECTIVES

Students will:

- Describe the function of hardware and software.
- Program the light matrix and learn how to debug a simple program.

STANDARDS

CSTA

2-CS-02 Design projects that combine hardware and software components to collect and exchange data.

2-AP-13 Decompose problems and subproblems into parts to facilitate the design, implementation, and review of programs.

2-AP-16 Incorporate existing code, media, and libraries into original programs, and give attribution.

2-AP-19 Document programs in order to make them easier to follow, test, and debug.

VOCABULARY

Debug

Light Matrix

- Ask students to consider how to create an image on their hubs. Ask them to create an image or picture by placing their squares onto the matrix to represent what squares will be lit. They do not need to use all 25 paper squares at the same time.

2. Explore

Students will explore how to create images on the light matrix.

Prompt students to consider how to break this task down into steps that a computer could do. Ask students to write the steps in pseudocode in their journals. Encourage them to think about previous coding experiences to think about the steps needed. Pseudocode is the set of directions written in words that show step-by-step instructions for what should happen. Pseudocode is used to help write code.

Hint: What libraries will they need to import?

Allow students to share their ideas.

Direct students to open a new project in the Python programming canvas. Ask students to erase any code that is already in the programming area. Students should connect to their hub.

Provide students with the sample code below. Have students type in the code to the programming canvas. Ask students to run this program.

```
from hub import light_matrix
import runloop

async def main():
    # show a happy face
    light_matrix.show_image(light_matrix.matrix.IMAGE_HAPPY)
    await runloop.sleep_ms(5000)
    light_matrix.clear()

runloop.run(main())
```

Discuss with students what happens.

Ask students if the program is still running on their hub. If the smiley face is still showing on the hub, then the program is still running. Direct students to stop the

program by either clicking the red circle with the white square on the lower right side of the screen or pressing the large circle button on the hub.

Review the lines of code with students. Discuss what they see happening when they run the program and what each line of code is telling the hub to do.

Create a New Image

Have students change the image that is displayed on the light matrix.

Challenge students to change the display from HAPPY to HEART which will change the display on the light matrix. Ask students to indicate what needs to be changed in the code to make it display a heart instead of a smiley face. Discuss with students.

Have students change their code and the comment in green. Run the program.

```
# show a heart  
light_matrix.show_image(light_matrix.IMAGE_HEART)  
await runloop.sleep_ms(5000)  
light_matrix.clear()
```

Discuss the program with students. Ask students what the # signals in the program. The # signals a comment which is not a command for the computer but is something programmers add to help them remember what a section of code is supposed to do.

Review with students how long the heart image stays lit. The image was lit for 5 seconds because the `await runloop.sleep_ms(5000)` show a "5000" in the parenthesis immediately after the command. The value inside the parenthesis is the number of milliseconds.

3. Explain

Discuss the program with students. Ask students questions like:

- Which libraries do you need to import to run these programs?
- What is the purpose of the `runloop.sleep_ms` piece of the code?
- Do you need to rewrite the entire code each time to change the image that is shown?
- How do you recognize if a program is still running? What are ways to stop a program?

4. Elaborate

Investigate programming a word on the hub.

Display Two Images

Have students create two images to display on the light matrix.

Challenge students to copy the lines of code under #Light up a heart and paste it at the end of their code. Students will now have two images to code.

Have them insert HAPPY into the second section. Discuss with students what will happen when they run their code with both sections of code. Review the lines of code as a group. The code should show a heart for 5 seconds followed by a smiley face for 5 sections.

```
from hub import light_matrix

import runloop

async def main():

    # show a heart

    light_matrix.show_image(light_matrix.IMAGE_HEART)

    await runloop.sleep_ms(5000)

    light_matrix.clear()

    # show a happy face

    light_matrix.show_image(light_matrix.IMAGE_HAPPY)

    await runloop.sleep_ms(5000)

    light_matrix.clear()

runloop.run(main())
```

Discuss the program with students.

Debugging

Investigate how to read errors.

Ask students to click the two horizontal lines in the center at the bottom of the programming canvas. This allows the console to open. The console is where information or error messages are printed.

Ask students to run the program again.

```
# show a heart
light_matrix.show_image(light_matrix.IMAGE_Heart)
await runloop.sleep_ms(5000)
light_matrix.clear()

# show a happy face
light_matrix.show_image(light_matrix.IMAGE_HAPPY)
await runloop.sleep_ms(5000)
light_matrix.clear()
```

Ask students to explain what happened. This represents an error which can occur if the information is not input correctly.

Notice that the error message in the console points you to line 6. If you look at line 6, you can see the word Heart in not all in capital letters. Change Heart to HEART and run the program again.

Notice that the program works correctly but the error message in the console remains unchanged. If you make an additional error, the next message in the console will follow exactly underneath the previous one.

Additional Exploration

Allow students to explore all the options for displaying images on the hub. To access additional images that students can display on their hub light matrix, look in the Knowledge Base under Light Matrix(). Allow students time to explore changing the images as time allows.

Have students look at the code they have been working with and consider how to change this code to write a word out on the hub rather than just showing an image.

They need to code the light matrix to write text instead of showing an image. Students might initially think that they need to show the word as an image. Prompt students to think about writing words out rather than showing words like a picture.

Ask students to write the word hello on their hub light matrix.

```
from hub import light_matrix
import runloop

async def main():
    # write your code here
    await light_matrix.write("Hi!")

runloop.run(main())
```

Ask students to reflect on the difference in this code from the previous code used to show images. This code allows students to write text on the light matrix, one letter at a time, scrolling from right to left. This is a string, or an array of characters. For example, the word "Hello" consists of a string of characters "h - e - l - l - o." Five characters are in this string.

Challenge students to write their names on the hub using this code. What is the only thing that students need to change in the code?

Optional: Allow students to explore writing various words and challenge them to write a word and then show an image.

For additional reference, point students to the Knowledge Base under Light Matrix Actions write().

5. Evaluate

Teacher Observation:

Discuss the program with students.

Ask students questions like:

- What is the purpose of the `await runloop.sleep_ms`?
- What would happen if you left off the `await runloop.sleep_ms` from your program?
- What troubleshooting or debugging did you run into?
- What is the function of the green words starting with the # sign in your code?
- What is the difference between coding with "show_image" and coding with "write"?

Self-Assessment:

Have students answer the following in their journals:

- What did you learn today about programming the light matrix?
- What characteristics of a good teammate did I display today?
- Ask students to rate themselves on a scale of 1-3, on their time management today.

- Ask students to rate themselves on a scale of 1-3, on their materials (parts) management today.

Pair Programming

Grade 6-9

45 minutes

Beginner

Pair Programming

Students learn why it is important to have roles when programming together.

Questions to investigate

- Why do programmers often work together when creating final code?

Materials needed

- SPIKE Prime sets ready for student use. Prior to the first lesson, please visit the following website for help with set up, kit organization and SPIKE App <https://education.lego.com/en-us/start/spike-prime/intro>
- Devices with the SPIKE App installed.
- Student journals
- Sticky notes

Prepare

Check to make sure SPIKE Prime hubs are charged, especially if connecting through Bluetooth.

1. Engage

Ignite a discussion with students. Ask students to imagine they are riding in a car with 4 people. Discuss what the roles of each person in the car would be. While students might identify some fun roles the people in the backseat have, hopefully they identify that at least one person in the front seat as the driver. If students do not recognize that someone should navigate (even if just following a map app), prompt them with questions to identify this role.

Focus on the roles of the driver and navigator. Ask students what roles do the driver and navigator have?

2. Explore

Explore with students the roles of driver and navigator when working together to program.

KEY OBJECTIVES

Students will:

- Practice pair programming.
- Modify programs.

STANDARDS

CSTA

2-CS-02 Design projects that combine hardware and software components to collect and exchange data.
2-AP-16 Incorporate existing code, media, and libraries into original programs, and give attribution.

VOCABULARY

Driver

Navigator

Pair programming

Discuss that pair programming is more than just two people working on the same project. In Pair Programming, each partner will have a role to play. Review these roles together as a group.

- A "driver" is thinking about the programming needed and using the computer to type in the code.
- A "navigator" helps direct the driver, making sure all the code is correctly written – the ports are correct, the operators are precise, and so forth.

Have each team member take a role to practice working together.

Ask students to create an image on their hub. Students can view Hub – Light Matrix in the **Knowledge Base** to see all the preloaded images.

- Ask Partner 1 work as the "driver".
- Partner 2 should work as the "navigator." Choose his/her favorite image.
- Partner 1 should type the program to display the image chosen by Partner 2.
- Partner 2 verifies the programming as it is typed and before the program is tested.

Have students hold up the hubs to share the images with the class.

Have students switch roles and then complete the same activity.

Each student should obtain one sticky note.

- Have each person write the name of the image they liked best on the sticky note.
- Have students create a bar graph on a wall to show the popularity of images.
- Sticky notes containing the same image will be placed vertically to represent the number of times it was used. Each image name should have its own column.

3. Explain

Review the bar graph with students and discuss their favorite images.

Discuss how students worked in their roles. Ask students questions like:

- Why do you think it is important for each person have a role when programming?
- What is important about the driver's role?
- What is important about the navigator's role?
- How often do you think partners should switch roles?
- How important were the roles when creating the bar graph?
Why do you think this is different?

4. Elaborate

Students will work together to play a debugging game.

Explain to students that each partner will take a turn being the "driver" and then the "navigator" to practice these roles.

- The partner acting as the driver will create a new program that inserts either a word or image or combination of both.
- The driver should intentionally make an error.
- The role of the navigator will be to find the error and explain how to fix it.

Students should complete several rounds to ensure they each get practice in both roles.

As a class, discuss how students were able to be successful in each role.

5. Evaluate

Teacher Observation

Discuss the program with students.

- Ask students questions like:
 - Where did you find the list of images available for the light matrix?
 - How did you work together as a team?
 - What did you do in each role to help your teammate?

Self-Assessment

Have students answer the following in their journals:

- What did you learn today that might be helpful in working with a partner to program?
- What did you do in each role to help your teammate?
- What characteristics of a good teammate did I display today?
- Ask students to rate themselves on a scale of 1-3, on their time management today.
- Ask students to rate themselves on a scale of 1-3, on their materials (parts) management today.

Communicating with Sounds

Grade 6-9

45 minutes

Beginner

Controlling Sounds

Students will learn to program different types of sounds.

Questions to investigate

- How can sound help someone to communicate an idea?

Materials needed

- SPIKE Prime sets
- Device with SPIKE App installed
- Student journal

Prepare

Check to make sure SPIKE Prime hubs are charged, especially if connecting through Bluetooth.

1. Engage

There are many ways to communicate with each other. Allow students to share some ways that they frequently communicate.

Engage students in a challenge to communicate without talking or writing/texting.

2. Explore

Students will explore how to program sounds.

Each partner in the team will need a matching set of three bricks of three different colors.

- Ensure that each student in the pair has the same three colors bricks.
- Partner A will stack the three bricks in any order. For example, if the student's bricks are green, red, and blue, then the stack might be green on the bottom, red in the middle, and blue on top.

KEY OBJECTIVES

Students will:

- Describe the function of hardware and software.
- Program sounds and beeps and learn how to debug a simple program.
- Create a sound pattern.

STANDARDS

CSTA

2-CS-02 Design projects that combine hardware and software components to collect and exchange data.

2-AP-13 Decompose problems and subproblems into parts to facilitate the design, implementation, and review of programs.

2-AP-16 Incorporate existing code, media, and libraries into original programs, and give attribution.

2-AP-19 Document programs in order to make them easier to follow, test, and debug.

VOCABULARY

Debug

Communicate

- Partner A should communicate to Partner B how to build the stack without talking. The simplest approach here would be to show Partner B how to build the stack.

Next, challenge the students to repeat the exercise with Partner B creating the stack. However, this time students cannot show the stack to each other.

- Ask the teams to come up with a way to communicate with each other using sounds.
- Demonstrate the idea by saying that the green brick is represented by a clap, the red brick by a snap, and the blue brick by a stomp.
- Make each sound as you create your stack.
- Students should create their own sounds for each brick and try to work together to build the stacks.

Discuss with students how sound can be used to communicate other than talking. Examples may include whistles or honking a horn to get someone's attention, clapping hands to show approval, or a scream that means you are very scared of something.

Creating sounds is a great way to communicate.

Beep, Beep

Investigate producing beep sounds on the hub.

Direct students to open a new project in the Python programming canvas. Ask students to erase any code that is already in the programming area. Students should connect their hub.

Allow students to investigate how sounds can be made using the hub with this sample program.

Provide students with the sample code below. Have students type the code into the programming canvas.

```
# import the sound
from hub import sound
import runloop

async def main():
    # play a beep for one second
    await sound.beep(400, 1000, 100)

runloop.run(main())
```

Review the code with students to identify what sections of the code tell the software what to do (importing the speaker), what portions of the code tell the hub what to do (beep), and what portions are just notes to the programmer (# green sections).

Ask students to identify areas that they could change to play different sounds. Encourage students to try different numbers.

Allow students to share the new sounds they have created by changing the numbers. Ask students to identify what the number 400, 1000 and 100 represents in the code. Explain that the number 400 represents the frequency, the number 1000 represents the length of time in milliseconds and the number 100 represents the volume.

Debugging

Ask students to replace the number 400 (frequency) with a number lower than 100. Discuss what happens.

Note: Students may have already discovered this on their own. Discuss as it becomes relevant.

```
# import the sound
from hub import sound
import runloop

async def main():
    # play a beep for one second
    await sound.beep(100, 1000, 100)
runloop.run(main())
```

Students will encounter that even though there isn't an error shown, you cannot hear any beep.

Ask students to explain what happened.

Play a Song

Challenge students to take what they learned in their investigation to add additional lines of code to create a song. Students should add several additional lines of code with different sounds to make a song.

Guide students to reference the sound options through the **Speaker** section in the **Knowledge Base** and select `beep()`. This guide will provide a range of numbers to use in the code to make different sounds as well as some guidance on errors.

Allow students time to explore as time allows.

Here is a sample song for students that might struggle.

```
# import the speaker
from hub import sound
import runloop

async def main():
    # play a song
    await sound.beep(400, 1000, 100)
    await sound.beep(450, 500, 100)
    await sound.beep(500, 1000, 100)

runloop.run(main())
```

3. Explain

Allow students to share their final song programs with each other and explain what their code shows.

Ask them questions like:

- What libraries need to be imported?
- How did you add additional lines to your code?
- What does each line of code in your program represent?
- What was challenging about this task?
- Where did you run into errors in programming? How did you fix or debug them?

Explain to students that the numbers included in the beep code are a float type because they can be whole numbers or include decimals. For example, the seconds could be set for 1.5 to be more exact.

4. Elaborate

Playing Sounds

Investigating playing pre-recorded sounds.

Explain to students that in addition to playing beeps, the hub can also be programmed to play pre-set sounds such as a cat's meow or a dog barking. Have students look at the code they have been working with and consider how to change this code to play a pre-set sounded hub rather than just a beep.

Help students recognize that they will still need to import the sound module. Students will need to know what pre-set sounds are available. Students can locate all the available sounds in the play sound word block. Students can access these later.

Ask students to create a code to investigate how sounds can be made by using the sample program Below. Prompt students to either copy this code to the programming canvas or type it in by modifying their existing project.

```
import runloop

from app import sound

async def main():

    await sound.play('Cat Meow 1')

runloop.run(main())
```

Troubleshooting tip: Ensure students are bringing in the proper libraries to program both parts of the challenge. For additional debugging information, reference the Knowledge Base.

Note: Because the sounds are coming from the device and not the hub, ensure the speakers on the device are turned on.

Discuss with students what is different about this code than the code for playing beeps. Students should see that the sounds are based on names and not numbers.

Students should recognize that the sound is coming from their device (app) and not the hub. Prompt students to think about why. When they import speaker, it allows the beeps to be heard. When they import sound, it allows sounds to be heard through the device.

Ask students why they think a "1" is located after the words cat meow. In this case, the 1 is to differentiate between other similar sounds Cat Meow 2 and Cat Meow 3. The number is part of the string or name for that variable. Ask students if they remember what a string is. You may need to review this term.

Allow students to try other sounds by changing the code from 'Cat Meow 1' to 'Triumph' or 'Doorbell1'. Prompt students to reference a word block lesson to locate the available sounds.

Challenge students to try different sounds from the library of options provided to create a pattern. Allow students to share their sounds with other groups.

Discuss with students how they can use sounds to communicate in a variety of ways.

5. Evaluate

Teacher Observation:

Discuss the program with students. Ask students questions like:

- How does programming beeps differ from programming sounds?
- Where do you hear the sound when the beep plays?
- Where do you hear the sound when the sound plays?

Self-Assessment:

Have students answer the following in their journals:

- How can you use beeps and sounds to communicate?
- What characteristics of a good teammate did I display today?
- Ask students to rate themselves on a scale of 1-3, on their time management today.

Ask students to rate themselves on a scale of 1-3, on their materials (parts) management today.

Digital Sign

Grade 6-9

45-90 minutes

Beginner

Digital Sign

Students will apply their knowledge of using the light matrix and programming sounds to design, build, and program a digital sign for advertisement.

Questions to investigate

- How can light and sound be combined to communicate an idea?

Materials needed

- SPIKE Prime sets
- Device with SPIKE App installed
- Student journal
- Sticky notes
- Scissors
- Colored paper

Prepare

Check to make sure SPIKE Prime hubs are charged, especially if connecting through Bluetooth.

1. Engage

Kate and Kyle created a popcorn stand, but no one seems to be buying their popcorn. Kate has an idea to create a sign that will get people's attention. First, they think about making a paper sign. Then, they think about the cool flashing sign they always see at the fair. Can your team help Kate and Kyle create a digital sign for their popcorn stand?

2. Explore

Students will design and build a sign that uses the light matrix then program their sign to display a message to advertise popcorn or another item. Consider allowing students choice in what item they might be selling and need to advertise.

KEY OBJECTIVES

Students will:

- Describe the function of hardware and software.
- Program lights and sounds to communicate a message.

STANDARDS

CSTA

2-CS-02 Design projects that combine hardware and software components to collect and exchange data.

2-AP-13 Decompose problems and subproblems into parts to facilitate the design, implementation, and review of programs.

2-AP-16 Incorporate existing code, media, and libraries into original programs, and give attribution.

2-AP-19 Document programs in order to make them easier to follow, test, and debug.

VOCABULARY

Advertise

Challenge students to think of different ways that the light matrix on their hub can be used to create a digital sign. Show images or videos of digital signs. Ask student to name any examples of digital signs they have seen before.

Students should sketch their sign design including writing out the message they play to display on the light matrix. Students need to consider the design of the sign so it will hold the hub in a way that allows their message to be displayed.

Hint: Consider orientation of the hub. Students should consider if they want words and images included in their sign.

Remind students to test their program several times to ensure the message is clear. Students should consider tradeoffs in their design such as the length of a message (easy to read) while making sure their advertisement is clear (people understand the message).

Requirements for this challenge:

- The light matrix cannot stand on its own. Students must build a frame for it.
- The digital sign must be advertising popcorn or the item of their choice.
- Prior to writing code, students must show a sketch of their design and write an explanation of what they want to the code.
- Students should use the comment feature in their Python code to explain what the lines of code are meant to do.

Note: If students struggle with programming their light matrix to create a digital sign remind them to use the Knowledge Base. You can find the information under Hub, then choose Light Matrix and look for show_image for inspiration on ideas and support in programming.

3. Explain

Students should share their sign design and explain how it works.

Ask students:

- How did you program your digital sign? Ask students to share their program comments to explain.
- What decisions did you have to make while creating your sign?
- What were areas that you had to debug or troubleshoot?
- What was difficult about this challenge?

4. Elaborate

Students can also draw attention to the popcorn stand or item of their choice by adding sound to their sign. Ask students to add a melody using the beep sound

to their sign that plays only while the lights are used on the hub (i.e. their message is scrolling).

Allow students to share their final design with lights and sounds with other groups.

5. Evaluate

Teacher Observation:

Discuss the program with students. Ask students questions like:

- What libraries were imported for your digital sign program?
- Where did you hear the sounds and the beeps?

Self-Assessment:

Have students answer the following in their journals:

- How can you use beeps and sounds to communicate?
- What characteristics of a good teammate did I display today?
- Ask students to rate themselves on a scale of 1-3, on their time management today.
- Ask students to rate themselves on a scale of 1-3, on their materials (parts) management today.

Ideas to Support Your Design

Grade 6-9

30-45 min.

Intermediate

Ideas to Support your Design

Practice giving and using feedback from others.

Questions to investigate

- How can input from others help me make a better design and program?

Materials needed

- SPIKE Prime sets ready for student use. Prior to the first lesson, please visit the following website for help with set up, kit organization and SPIKE App <https://education.lego.com/en-us/start/spike-prime/intro>
- Devices with the SPIKE App installed
- Student journals

Prepare

- Ensure SPIKE Prime hubs are charged, especially if connecting through Bluetooth.
- Ensure students have their built model from the Digital Signs lesson.

1. Engage

Review the model for providing feedback with students.

Explain to students the following guidelines for giving feedback. Consider posting the guidelines for student reference.

- Feedback is not doing something for someone else.
- You should not rebuild a model for someone else.
- You should not type into someone's program.
- You should ask questions of each other.
- You should share your ideas and show your own programming, explaining why and how you did something.
- You should be encouraging and helpful to others and not provide negative or mean comments.

KEY OBJECTIVES

Students will:

- Give specific feedback on a peer's project.
- Explore how to use feedback to improve a project.

STANDARDS

1B-IC-20 Seek diverse perspectives for the purpose of improving computational artifacts.

VOCABULARY

Feedback
Specific
Positive
Negative

2. Explore

Have students work together to provide feedback to each other. Use the digital sign models.

Have two teams work together to provide feedback to each other. Teachers should model the process and demonstrate what specific feedback looks like and sounds like.

Review the procedure with students. Then have students take turns providing feedback.

- Team B will show their working model.
- Team A provides feedback while Team B takes notes in their journal.
- Then teams can switch roles. Team A will show their working model and take notes while Team B provides feedback.

Feedback should include:

1. Tell something they really like. This could be the model, program, or design.
2. Tell something that worked well.
3. Share something the group could try differently.
4. Share anything that is confusing, did not work or that could be improved,
 - Remind students to be kind and clear in explaining why it is not clear or could be improved.
 - Let the team receiving the feedback ask questions as needed for more clarity.
 - The team giving feedback can also share ideas for improvement.

Teacher tip – Model providing feedback for the class frequently to help them learn to use positive language instead of negative language when providing feedback. Also practice taking feedback and thinking about how to use it rather than becoming defensive.

3. Explain

Have students discuss what they learned from their feedback session.

Ask students questions like:

- What did you notice in models that worked well?
- What ideas did you get from others?
- What is something you can do with your feedback?

4. Elaborate

Students should incorporate the feedback they were given.

Give students time to modify their designs and program based on the feedback they received. Have students document their changes in their journal.

Allow students to share their updated models and programs. Ask students to share what changes they incorporated and how they were able to make the changes.

5. Evaluate

Teacher Observation:

Discuss the program with students.

Ask students questions like:

- How did you use the feedback given?
- How did it feel to give feedback to others? And how did it feel when receiving it?
- How did you work to provide good feedback today?

Self-Assessment:

Have students answer the following in their journals:

- What did you learn today about providing good feedback?
- What did you learn today about how feedback can help in your work?
- What characteristics of a good teammate did I display today?
- Ask students to rate themselves on a scale of 1-3, on their time management today.
- Ask students to rate themselves on a scale of 1-3, on their materials (parts) management today.

Career Connections – Lesson Extension

Students will extend learning to explore and research careers related to the topic explored

STEM, Computer Science

6-8 grade

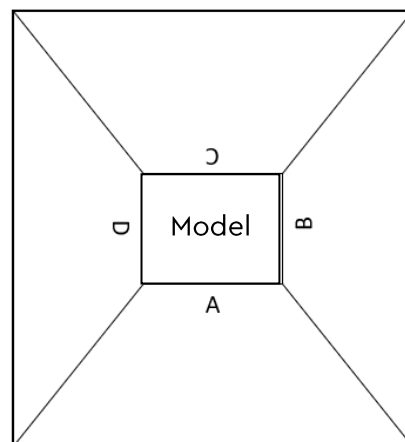
60-90 min.

Beginner

Career Connections

Prepare

- LEGO® Education model from previous lesson.
- Have chart paper and markers available for student use. Prepare the charts ahead of time by drawing lines to separate the chart paper into four sections and label each section with a letter: A, B, C, D. The chart would like this:



You may also want to include the name of the career area in the center.

- Prior to starting the lesson, make sure you have enough devices and access to the Internet for student use during this lesson.

1. Engage

Have students brainstorm possible jobs they see connected to the Digital Sign lesson. Students should consider all the possible jobs involved in creating an advertisement such as this. Prompt students to think about the model they

KEY OBJECTIVES

Students will:

- Articulate their personal interests and goals.
- Relate their personal interests and goals into possible career pathways.
- Explore various careers in career pathways.

STANDARDS

Career Ready Practice 10- Plan education and career path aligned to personal goals. (CCTC)

created, the programming, and deciding on the message to include.

2. Explore

Working in groups, students will explore one career cluster to identify jobs that relate to the Digital Sign lesson they completed. Students will use the model and programs created in that lesson to tie into the relevant jobs they find during their investigation.

Review the 16 career clusters if needed. Explain to students that they are going to look at specific jobs within the career clusters that relate to the concepts they have been investigating using the model they still have built.

- Consider assigning career cluster areas or letting students choose.
- Possible career areas to consider are Marketing, Business, IT, and STEM.

Ask students to consider where they can find the most reliable information on these career areas. Provide students with resources/websites as needed. Provide the students with a piece of chart paper with the 4 quadrants marked off. The information students should provide in each quadrant is

A. What career areas are relevant to the model/concepts you have been learning. Students can place the model in the center of the chart paper which will allow them to easily reference it in this section.

B. What skills are needed for this career area or job? Did you use any of these skills when working with the model/concept from the lesson? If so, which ones and how did you use them?

C. What are the types of jobs you might have in this career area? What specialized training or certifications would you need to get this job?

D. What about this career area or job interests you and why?

3. Explain

Have each group share their findings with the whole class which will allow the class to learn about jobs in each career area. Ask students questions such as:

- How are the concepts we are learning now related to the jobs you investigated?
- What career cluster does this job belong in?
- What kind of qualifications are needed for this job?
- What kind of day-to-day responsibilities are associated with this job?

- What similar jobs are related to this (specific job)?
- Are you interested in this job?

4. Elaborate

Ask students to reflect on these career areas. How do you see these concepts overlapping across career clusters?

Ask students to reflect in a journal or other appropriate ways on all the career areas and jobs discussed to consider which might be most interesting to them and why.

5. Evaluate

Teacher Observation:

Discuss the program with students.

Ask students questions like:

- What are some careers involved with creating digital signs?
- What from the careers discussed interests you?
- What else would you be interested in knowing about these careers?

Self-Assessment:

Have students answer the following in their journals:

- What did I learn today that interested me about different careers?
- What characteristics of a good teammate did I display today?
- Ask students to rate themselves on a scale of 1-3, on their time management today.
- Ask students to rate themselves on a scale of 1-3, on their materials (parts) management today.

Career Exploration Activity

C	
<p>What are the types of jobs you might have in this career areas? What training or specialized certifications would you need to get this job?</p>	
D	B
<p>What about this career area or job interests you and why?</p>	<p>What skills are needed for this career area or job? Did you use any of these skills when working with the model/concept from the lesson? If so, which ones and how did you use them?</p>
A	
<p>What career areas are relevant to the model/concepts you have been learning. Students can place the model in the center of the chart paper which will allow them to easily reference it in this section.</p>	
<p>LEGO Education Model</p> <p>Career Cluster</p>	

Unit 2: Motors

A LEGO® Education Unit

Unit Introduction

This unit allows students to explore essential computer science principles and programming concepts of the text-based coding language, Python, while creating models that move autonomously. Students will investigate how robots are motorized and can help automate actions. The lessons are designed in an order that allows students to progress in their skills and knowledge in the following areas:

- Program single and multiple motors run using variables for time, speed, and degrees
- Program a motor to move to a specific position using the shortest path
- Build and program a model that automates a task
- Use and modify existing code to explore new ideas
- Create pseudocode to support creating algorithms
- Utilize code comment features to document parts of a program
- Define and decompose a problem

Unit Learning Promise

In this unit, students will explore how to control motors in different ways, how to effectively utilize motors in robots, and how to utilize motors to automate tasks. Students will utilize pseudocode to support creating algorithms and code comments to document their programs.

Investigation Questions:

How do robots move? How can automation make tasks easier?
Why is documenting programs important?

Unit Lessons

Lesson 1	Lesson 2	Lesson 3	Lesson 4	Lesson 5	Lesson 6
Making Moves with Motors	New Moves with Motors	Automating Action	Hopper Run	Race Day	Ideas to Help
Time: 45 min.	Time: 45 min.	Time: 45 min.	Time: 45 min.	Time: 45 min.	Time: 45 min.

Assessment: We recommend assessing students on various skills throughout the unit.

- Use the progression of lessons as an opportunity to provide on-going feedback to prepare students for success for the open-ended project at the end of the unit.
- Each lesson includes a recommendation for teacher observations, student self-assessment, evaluation of success.

Unit Standards

CSTA
2-CS-02 Design projects that combine hardware and software components to collect and exchange data.
2-AP-10 Use flowcharts and/or pseudocode to address complex problems as algorithms.
2-AP-13 Decompose problems and subproblems into parts to facilitate the design, implementation, and review of programs.
2-AP-16 Incorporate existing code, media, and libraries into original programs, and give attribution.
2-AP-17 Systematically test and refine programs using a range of test cases.
2-AP-19 Document programs in order to make them easier to follow, test, and debug.
1B-IC-20 Seek diverse perspectives for the purpose of improving computational artifacts.

Integrated Standards

NGSS		
MS-ETS1-4 Develop a model to generate data for iterative testing and modification of a proposed object, tool or process such that an optimal design can be achieved.		
Common Core English Language Arts (ELA)		
6 th Grade	7 th Grade	8 th Grade
SL.6.1 Engage effectively in a range of collaborative discussions (one-on-one, in groups, and teacher-led) with diverse partners on grade 6 topics, texts, and issues, building on others' ideas and expressing their own clearly	SL.7.1 Engage effectively in a range of collaborative discussions (one-on-one, in groups, and teacher-led) with diverse partners on grade 7 topics, texts, and issues, building on others' ideas and expressing their own clearly	SL.8.1 Engage effectively in a range of collaborative discussions (one-on-one, in groups, and teacher-led) with diverse partners on grade 8 topics, texts, and issues, building on others' ideas and expressing their own clearly
SL.6.2 Interpret information presented in diverse media and formats (e.g., visually, quantitatively, orally) and explain how it contributes to a topic, text, or issue under study	SL.7.2 Analyze the main ideas and supporting details presented in diverse media and formats (e.g., visually, quantitatively, orally) and explain how the ideas clarify a topic, text, or issue under study	SL.8.2 Analyze the purpose of information presented in diverse media and formats (e.g., visually, quantitatively, orally) and evaluate the motives (e.g., social, commercial, political) behind its presentation
SL.6.4 Present claims and findings, sequencing ideas logically and using pertinent descriptions, facts, and details to accentuate main ideas or themes; use appropriate eye contact, adequate volume, and clear pronunciation	SL.7.4 Present claims and findings, emphasizing salient points in a focused, coherent manner with pertinent descriptions, facts, details, and examples; use appropriate eye contact, adequate volume, and clear pronunciation	SL.8.4 Present claims and findings, emphasizing salient points in a focused, coherent manner with relevant evidence, sound valid reasoning, and well-chosen details; use appropriate eye contact, adequate volume, and clear pronunciation
RST.6-8.3 Follow precisely a multistep procedure when carrying out experiments, taking measurements, or performing technical tasks	RST.6-8.3 Follow precisely a multistep procedure when carrying out experiments, taking measurements, or performing technical tasks	<u>RST.6-8.3</u> Follow precisely a multistep procedure when carrying out experiments, taking measurements, or performing technical tasks.
<u>L.6.6</u> Acquire and use accurately grade-appropriate general academic and domain-specific words and phrases; gather vocabulary knowledge when	L.7.6 Acquire and use accurately grade-appropriate general academic and domain-specific words and phrases; gather vocabulary knowledge when	L.8.6 Acquire and use accurately grade-appropriate general academic and domain-specific words and phrases; gather vocabulary knowledge when

considering a word or phrase important to comprehension or expression

considering a word or phrase important to comprehension or expression

considering a word or phrase important to comprehension or expression

Making Moves with Motors

Grade 6-9

45 minutes

Beginner

Making Moves with Motors

Students will work with individual motors to create a dance party.

Questions to investigate

- How can engineers and programmers work together to make something move?

Materials needed

- SPIKE Prime sets ready for student use. Prior to the first lesson, please visit the following website for help with set up, kit organization and SPIKE App <https://education.lego.com/en-us/start/spike-prime/intro>
- Devices with the SPIKE App installed.
- Student journals

Prepare

Ensure SPIKE Prime hubs are charged, especially if connecting through Bluetooth.

1. Engage

Engage students in thinking about how motors move. Have students stand to act as motors. Have students create a line and ask them to walk forward 5 steps. Have them try this a couple of times. Ask students why everyone does not move the same distance.

- Ask students to raise their right arm when you say "Raise". Wait a couple of seconds and say "Raise." Why didn't everyone move at the exact same time?
- Have them do this a couple of times. Ask students why everyone does not move the same way.

KEY OBJECTIVES

Students will:

- Program motors to turn individually using parameters of time and speed
- Create a robot dance party

STANDARDS

CSTA

2-CS-02 Design projects that combine hardware and software components to collect and exchange data.
2-AP-10 Use flowcharts and/or pseudocode to address complex problems as algorithms
2-AP-16 Incorporate existing code, media, and libraries into original programs, and give attribution.
2-AP-17 Systematically test and refine programs using a range of test cases.
2-AP-19 Document programs in order to make them easier to follow, test, and debug.

VOCABULARY

Pseudocode
Initialize
Motor
Speed

- Ask students to clap when you say "clap." Wait a couple of seconds and say clap. Have them try this a couple of times. Ask students why everyone does not clap at the same time.
- Discuss how a group of robots programmed with the same code would react given the same command simultaneously.
- Discuss how a robot moves and why movement for a given command would be the same every time. Ask students when this would be useful or helpful.

2. Explore

Students will explore working with motors and ways to code movement.

Direct students to the **GETTING STARTED** section of the Knowledge Base in the SPIKE App. Here students can access **4. Controlling Motors**. This Getting Started lesson provides students with an early experience in building and coding with SPIKE Prime.

To investigate the programming of motors in more depth, open a new project in the Python programming canvas. Ask students to erase any code that is already in the programming area. Students should connect their hub.

Plug the large motor into port A of the hub.

Run Single Motor for Degrees

Running a motor for degrees allows you to set how far the motor will run.

Brainstorm with students on ways to program the motor to move. Discuss what information the software needs to run the hardware correctly.

Write a pseudocode program for making the motor run for 2 seconds with students. Pseudocode is writing in words what you want the program to do.

An example could be:

- Import motor
- Turn motor on
- Move clockwise for 2 rotations

Note: The code written in steps does not need to match exactly what will be put in the program. Writing pseudocode helps students determine what is needed.

Provide students with the sample code below to move the motor. Have students type the code into the programming canvas. (Students can copy and paste the

sample code from the **Knowledge Base Getting Started Part 4: Controlling the Motors** section.)

```
import motor
from hub import port
import runloop

async def main():
    # Run a motor on port A for 360 degrees at 720 degrees per second.
    await motor.run_for_degrees(port.A, 360, 720)

runloop.run(main())
```

Troubleshooting Tip: Ensure students have the motor plugged into the A port or change port.A to the correct port.

Discuss what the 360 and 720 represent in the parenthesis in the last line of code. The code comment provides a hint.

- Prompt students to put different numbers in for these two values to investigate how it changes how the motor moves.
- Allow students to share the new examples they create and discuss ways to program the motor.

Run Multiple Motors

Prompt students to add an additional medium motor to their hub. Having multiple motors attached will allow students to run more than one motor at a time. Ask students to consider what type of robots they could create that would need more than one motor. What function could multiple motors serve?

Brainstorm with students about what they need to add to their program to allow for two motors to run. The task is to make the two motors run (one in port A and the second in port B).

Remind students that they can plug the motors into any port. However, the program needs to match the chosen port, which might not match the sample program.

Ask students to add to the pseudocode used previously. Add wording to make the motors run at the same time.

Have students use the pseudocode as inspiration to help write their new program that will run both motors.

```
import motor
from hub import port
import runloop

async def main():
    # Run a motor on port A for 360 degrees at 720 degrees per second.
    await motor.run_for_degrees(port.A, 360, 720)
    # Run a motor on port B for 360 degrees at 720 degrees per second.
    await motor.run_for_degrees(port.B, 360, 720)

runloop.run(main())
```

3. Explain

Discuss with students how the program worked. Did students notice that the motors turned in the opposite direction? Did students notice that the first motor ran and then the second ran? This is because we added an await on an awaitable.

Ask students to modify the code by removing the await that is before each `motor.run_for_degrees(port.B, 360, 720)` and run the program again. Did students notice that the motors now ran at the same time? That's how an awaitable works.

You can find more about awaitables in the Knowledge Base in the Getting Started, 4. Controlling Motors.

What different combinations of the degrees and speed did students choose?

Ask students questions like:

- Explain the difference between importing the library for only one motor and two motors.
- What parameters for the motor can you change to make it move in different ways?

- How did the two motors move? Did they move at the same time? Why or why not?
- How can you troubleshoot errors that occur when working with motors?

Explain to students that the number specified for velocity is an integer type input. An integer type input can be a positive or negative whole number ranging from -1000 to 1000.

4. Elaborate

Dance Party

Challenge students to create a short program to run the 2 medium motors to create a dance party. Play some music to get kids creative. For example, students could have both motors moving in one direction or have each motor moving in opposite directions.

- Allow students to add additional bricks to their motors to create something new.
- Prompt students to add sounds and hub lights to their program to complete the dance party.
- Remind them to think about which libraries need to be imported to support their programs and where to find help in the **Knowledge Base** as needed.

Students should share their dancing robots and explain their programs.

Disconnect the motors and return all parts to the correct bin location.

5. Evaluate

Teacher Observation:

Discuss the program with students.

Ask students questions like:

- What happened on your hub when you ran the program?
- What libraries did you use to run this program?
- How can engineers and computer programmers work together to create a way to communicate ideas to others?
- How can engineers and programmers work together to make something move?

Self-Assessment:

Have students answer the following in their journals:

- How do you determine which motor you are programming to move if multiple motors are attached to the hub?
- What characteristics of a good teammate did I display today?
- Ask students to rate themselves on a scale of 1-3, on their time management today.
- Ask students to rate themselves on a scale of 1-3, on their materials (parts) management today.

New Moves with Motors

Grade 6-9

45 minutes

Beginner

New Moves with Motors

Students program their motors to move to exact positions and for exact degrees.

Questions to investigate

- How do robots move precisely?

Materials needed

- SPIKE Prime sets
- Devices with the SPIKE App installed.
- Student journals

Prepare

- Ensure SPIKE Prime hubs are charged, especially if connecting through Bluetooth.
- Ensure students have built the Getting Started 2: Motors and Sensors model through step 19 utilized in the Making Moves with Motors Lesson.

1. Engage

Using a tire from the SPIKE Prime set, ask students to think of different ways to measure how far the tire would drive if it went around one rotation. Students should come up with several methods to determine this distance. Prompt student to think about the wheel as a circle. Discuss with them how to calculate the circumference to determine the distance. Ask students how they could measure the circumference of the wheel using a flat surface if they are struggling.

Ask students to imagine they have a robot with wheels. They need to code the motors to move a certain distance. Ask students how could they program the motor to move other than for time? How would this help move the robot more accurately? Discuss ideas with the students about how to move only $\frac{1}{2}$ a turn (0.5 rotation), a full turn (1 rotation), 2 turns (2 rotations).

KEY OBJECTIVES

Students will:

- Program a motor to move to position using the shortest path.
- Program a motor to move to a specific position.
- Program a motor to move a defined number of degrees.

STANDARDS

CSTA

2-CS-02 Design projects that combine hardware and software components to collect and exchange data.
2-AP-10 Use flowcharts and/or pseudocode to address complex problems as algorithms
2-AP-16 Incorporate existing code, media, and libraries into original programs, and give attribution.
2-AP-17 Systematically test and refine programs using a range of test cases.
2-AP-19 Document programs in order to make them easier to follow, test, and debug.

VOCABULARY

Degrees
Position

Note: The wheel has a diameter of 5.6 cm (2.2 in.) and travels 17.6 cm (6.9 in) per rotation or a complete 360° degree circle.

2. Explore

Students will explore how to program motors to move using degrees.

Run Single Motor for Degrees

Running a motor for degrees allows you to determine how far the motor will turn in degree increments – 1/360th of a rotation. Brainstorm with students how they could program the motor to move.

Students should think about what information the software needs to run the hardware correctly. Write a pseudocode program with students for making the motor run for one complete circle.

An example could be:

- Import motor
- Turn motor on
- Move motor 360 degrees

Note: The code written step-by-step does not need to match exactly what will be in the program. Pseudocode is used to help students think through what steps are needed so programming can be coded linearly.

Direct students to open a new project in the Python programming canvas. Ask students to erase any code that is already in the programming area. Students should connect their hub and plug a motor into the A port of their hub.

Provide students with the sample code to move the motor. Ask students to run the program. Students can also copy and paste this code from the **Knowledge Base Getting Started 4: Controlling Motors** section into the programming canvas.

```
import motor
```

```
from hub import port
```

```
import runloop
```

```
async def main():
```

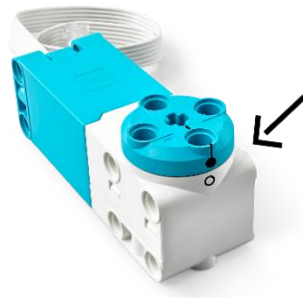
```
    # Run a motor on port A for 360 degrees at 720 degrees per second.
```

```
    await motor.run_for_degrees(port.A, 360, 720)
```

```
runloop.run(main())
```

Discuss with students how the motor moved. Help students identify the marker on the motor that allows you to set the position to 0 degrees. This will allow students to measure how far the motor moves.

Ask students to set the mark to the 0 position. Run the program again. Ask students if the marker return to the same position.



Prompt students to put different numbers into the degree values to see how the motor moves. Have them try large and small numbers.

Ensure students specifically try to run the motor using a negative value. Ask them what changed. (The direction reversed.)

```
# Run a motor on port A for -360 degrees at 720 degrees per second.
```

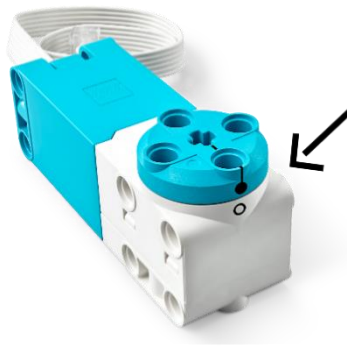
```
motor.run_for_degrees(port.A, -360, 720)
```

Ask students to enter a number larger than 360. What happens?

Discuss what students discover after their investigations.

Run Single Motor to Position

Students will explore how to run a single motor to a position.



Have students locate the position marker again and ensure it is lined up. When aligned, the motor is in the 0-degree position. Discuss with students how to program the motor to move to a specific position (the motor will stop at an exact position). Prompt students to change their pseudocode to use position instead of degrees.

Provide students with the sample code to move the motor.

```
import motor
from hub import port
import runloop

async def main():
    # Run a motor on port A to 0 degrees at 720 degrees per second.
    await motor.run_to_absolute_position(port.A, 0, 720, direction=motor.SHORTEST_PATH )

runloop.run(main())
```

Discuss the program with students to identify what happens when they run the code. If students start at the 0 degrees position, the motor will not move.

Ask students to move the motor so it is no longer in the 0 degrees position. Run the program. The motor should move to the 0 degrees position. Which direction did it move to get there?

Ask students to run the program several more times with the motor starting in different positions each time. Discuss what happens. Students should identify that the motor is not moving the same each time. The motor moves in the direction that makes the shortest path back to the 0 degrees position.

Review the lines of code together to highlight why the motor is moving in this way.

What is the longest move a motor can make to return to 0 position?

Ask students to change the code to make the motor stop in different positions. You can provide this sample code to them and then let them explore ideas of how they might change the code.

```
import motor

from hub import port

import runloop

async def main():

    # Run a motor on port A to 0 degrees at 720 degrees per second. Then it will wait 2
    # seconds and move to 90 degrees in the clockwise direction.

    await motor.run_to_absolute_position(port.A, 0, 720, direction=motor.SHORTEST_PATH)

    await runloop.sleep_ms(2000)

    await motor.run_to_absolute_position(port.A, 90, 720, direction=motor.CLOCKWISE)

runloop.run(main())
```

Remind students that when exploring there is a high likelihood of a receiving an error message in the bottom console and the program not executing.

3. Explain

Allow students to share the new example programs they created for degrees and position and discuss ways to program the motor. Discuss how the motor can be set to move for degrees and when this might be useful in a program.

- How does this program work?
- When would it be useful to program a motor to move using degrees or to position?
- Why does using degrees or to position allow for more precision in movement than programming for seconds?
- How did the motor move when you put a number in larger than 360?
- How can you calculate how many degrees you will need to program?
- When using the shortest path in your program for position, why does the motor not move if you start at the 0 degree position?
- How can you program the motor to move in the opposite direction?

4. Elaborate

Work together to work on debugging. Provide students with each of the following codes. Discuss what is wrong with each code. Make sure students run each code and review the error message in the console.

1. What is missing?

```
import motor

import runloop

async def main():

    # Run a motor on port A to 0 degrees at 720 degrees per second. Then it will wait 2
    # seconds and move to 90 degrees in the clockwise direction.

    await motor.run_to_absolute_position(port.A, 0, 720, direction=motor.SHORTEST_PATH )

    await runloop.sleep_ms(2000)

    await motor.run_to_absolute_position(port.A, 90, 720, direction=motor.CLOCKWISE)

runloop.run(main())
```

Students should identify that the port is not imported and therefore cannot set the motor port. The error message in the console is

```
Traceback (most recent call last):
File "Project 1", line 8, in <module>
File "Project 1", line 6, in main
NameError: name 'port' isn't defined
```

Students need to add `from hub import port` after the `import motor` line. Have students add this in and run the program again to check their program.

2. What number is wrong and why?

```
import motor

from hub import port

import runloop
```



```
async def main():
```

```
    # Run a motor on port A to 0 degrees at 720 degrees per second. Then it will wait 2 seconds and move to 90 degrees in the clockwise direction.
```

```
    await motor.run_to_absolute_position(port.A, 0, 7200)
```

```
    await runloop.sleep_ms(2000)
```

```
    await motor.run_to_absolute_position(port.A, 90, 720, direction=motor.CLOCKWISE)
```

```
runloop.run(main())
```

Students should identify that the motor velocity is set to 7200 which is outside the allowable -1000 to 1000 range. The program, however, will still run with no error message in the console. The power will revert to -1000 and run the program.

3. What is wrong?

```
import motor
```

```
from hub import port
```

```
import runloop
```

```
async def main():
```

```
    # Run a motor on port A to 0 degrees at 720 degrees per second. Then it will wait 2 seconds and move to 90 degrees in the clockwise direction.
```

```
    await Motor.run_to_absolute_position(port.A, 0, 720)
```

```
runloop.run(main())
```

Students should recognize that the word Motor is capitalized in line 7. The error message indicates that there is an issue in line 7 and that the error is a value that is not defined.

Traceback (most recent call last):

File "Project 1", line 9, in <module>

File "Project 1", line 7, in main

NameError: name 'Motor' isn't defined

5. Evaluate

Teacher Observation:

Discuss the program with students.

Ask students questions like:

- What are different ways you can program your motors to move?
- Which methods allow for more precise movements?
- What are ways you can program to move using position?

Self-Assessment:

Have students answer the following in their journals:

- What did you learn today that might be helpful in programming robots to move with precision?
- What characteristics of a good teammate did I display today?
- Ask students to rate themselves on a scale of 1-3, on their time management today.

Ask students to rate themselves on a scale of 1-3, on their materials (parts) management today.

Automating Action

Grade 6-9

45 minutes

Beginner

Automating Action

Students will add a motor to a model to automate the action.

Questions to investigate

- How can a mechanism be motorized to add automation? How can automation make tasks easier?

Materials needed

- SPIKE Prime sets
- Devices with the SPIKE App installed.
- Student journals

Prepare

Ensure SPIKE Prime hubs are charged, especially if connecting through Bluetooth.

1. Engage

Engage students in a discussion about automation. Prompt students to provide ideas about what automation is and provide examples of everyday items that provide automation. Consider showing images or videos as needed for students to understand how automation happens.

Show students the video from the SPIKE Prime **Extra Resources - Ideas, the LEGO Way lesson**. This lesson is located at <https://education.lego.com/en-us/lessons/prime-extra-resources/ideas-the-lego-way#ignite-a-discussion>. Use this video to prompt a discussion about what could be automated in the model shown. Discuss ideas for adding a motor to automate the movement provided by the two people in the video.

KEY OBJECTIVES

Students will:

- Build and program a model that automates a task.

STANDARDS

CSTA

2-CS-02 Design projects that combine hardware and software components to collect and exchange data.

2-AP-10 Use flowcharts and/or pseudocode to address complex problems as algorithms

2-AP-16 Incorporate existing code, media, and libraries into original programs, and give attribution.

2-AP-17 Systematically test and refine programs using a range of test cases.

2-AP-19 Document programs in order to make them easier to follow, test, and debug.

NGSS

MS-ETS1-4 Develop a model to generate data for iterative testing and modification of a proposed object, tool or process such that an optimal design can be achieved.

VOCABULARY

Automate

2. Explore

Challenge students to add a motor to the tower arm in order to automate the movement to the character rather than needing to move it manually.

Students will design an idea for adding a single motor to the tower arm. Students should sketch an idea for adding the motor then write out their pseudocode for how to program the tower arm to move. Students need to consider the design of the tower arm, understanding how it can move, in order to program the movement to be automated.

Open a new project in the Python programming canvas. Ask students to erase any code that is already in the programming area. Students should connect their hub.

Students will program the movement of the arm. Use the pseudocode to help create the program. Remind students to test their program several times in order to ensure it moves as expected. Ask students to add code comments using # to explain the steps of the program.

3. Explain

Students should share their final design and program, explaining how the program automates the movement.

Ask students questions like:

- How did you program your model? Ask students to share their program comments to explain.
- What decisions did you have to make while adding the motor?
- What decisions did you have to make when programming your movement?
- How does adding the motor to automate the movement make the task easier?
- What were areas that you had to debug or troubleshoot?
- What was difficult about this challenge?

4. Elaborate

Have two groups of students come together to form a larger group.

Group 1 will present their model and program to Group 2. Group 2 will then present their model and program to Group 1.

Both groups should focus on the code comments provided in each program. The groups will work together to make sure the code comments clearly outline each program.

Working together, groups should create basic user instructions for their

automation. Each group can add these instructions to the top of their program using a code comment.

5. Evaluate

Teacher Observation:

Discuss the program with students.

Ask students questions like:

- How were you able to motorize your mechanism?
- What decisions did you need to make in order to add the motor?
- Why are code comments, including user instructions, important to create?

Self-Assessment:

Have students answer the following in their journals:

- What did you learn today about adding automation to movement?
- When can automation be helpful?
- What characteristics of a good teammate did I display today?
- Ask students to rate themselves on a scale of 1-3, on their time management today.
- Ask students to rate themselves on a scale of 1-3, on their materials (parts) management today.

Hopper Run

Grade 6-9

45 minutes

Beginner

Hopper Run

Students build and program a robot without wheels to go forward.

Questions to investigate

How does the design of a robot determine how it can move?

Materials needed

- SPIKE Prime sets
- Devices with the SPIKE App installed.
- Student journals

Prepare

Ensure SPIKE Prime hubs are charged, especially if connecting through Bluetooth.

1. Engage

Engage students to think about how to move when using more than one motor. Put students in groups of 4. Have them link together either by holding arms or a ruler/stick between them. Ask groups to move together in unison in a line. Discuss what happens.

Challenge students to become synced in the way they move (stepping forward together at the same time, for the same distance). See how students communicate together to make these moves happen. Discuss as a group.

2. Explore

Students will build a hopper model to investigate different ways to move using the motors.

Direct students to the **BUILD** section in the SPIKE App. Here students can access

KEY OBJECTIVES

Students will:

- Program two motors to move simultaneously.
- Build and program a robot without wheels to move forward.

STANDARDS

CSTA

2-CS-02 Design projects that combine hardware and software components to collect and exchange data.

2-AP-10 Use flowcharts and/or pseudocode to address complex problems as algorithms

2-AP-13 Decompose problems and subproblems into parts to facilitate the design, implementation, and review of programs.

2-AP-16 Incorporate existing code, media, and libraries into original programs, and give attribution.

2-AP-17 Systematically test and refine programs using a range of test cases.

2-AP-19 Document programs in order to make them easier to follow, test, and debug.

VOCABULARY

Motor Pair

the building instructions for **Hopper**. Ask students to build the model. The building instructions are also available at <https://education.lego.com/en-us/support/spike-prime/building-instructions>.

Direct students to open a new project in the Python programming canvas. Ask students to erase any code that is already in the programming area. Students should connect their hub.

Get Moving

Challenge students to have their hopper run forward for 5 seconds.

Discuss with students how the hopper model will move using the two motors attached to move simultaneously. Ask students to write a pseudocode program to explain how they need to program hopper to move forward. Discuss the pseudocode examples students create.

Example pseudocode:

- Import motors
- Run motors for 5 seconds

Note: This is a sample of what students might be thinking and does not represent an actual code.

Share this sample program with students. Students will need to type this program into the programming canvas.

```
from hub import port
import runloop
import motor_pair

async def main():
    # Pair motors on port E and F
    motor_pair.pair(motor_pair.PAIR_1, port.E, port.F)

    # Move straight at default velocity for 5 second
    await motor_pair.move_for_time(motor_pair.PAIR_1, 5000, 0)

runloop.run(main())
```

Note: Remind students to watch for errors in the console. Students can reference the line from the error message to pinpoint where a typing error

might have occurred.

Review how the hopper model moved with this code. Then, take students to the **Knowledge Base** and open the **Motor Pairs section**. This section is an in-depth review of how to program motors together.

Review the `move_for_time` section with students. Note to students that they can set variables for time, steering and velocity (speed). All are integer type inputs (whole numbers). Time can be any number greater than 0. Steering has a range from -100 to 100. steering has a range from -100 to 100. Velocity (Speed) has a range of -1000 to 1000.

Challenge students to create a 3-2-1 countdown on their hubs and the hopper to travel a distance of 50 cm. Remind students to consider the needed libraries and the time needed to move 50 cm when writing their program.

Sample program:

```
from hub import port, light_matrix
import runloop
import motor_pair

async def main():
    # Countdown
    await light_matrix.write("3")
    await runloop.sleep_ms(1000)
    await light_matrix.write("2")
    await runloop.sleep_ms(1000)
    await light_matrix.write("1")
    await runloop.sleep_ms(1000)

    # Pair motors on port A and B
    motor_pair.pair(motor_pair.PAIR_1, port.E, port.F)

    # Move straight at default velocity for 5 second
    await motor_pair.move_for_time(motor_pair.PAIR_1, 7000, 0, velocity=500)
```



```
runloop.run(main())
```

3. Explain

Discuss with students how they were able to move their models and review the different code combinations together.

Ask students questions like:

- How did your program work?
- How did programming two motors differ from programming one motor?
- How does the design of the hopper model determine how it is able to move?
- What was difficult about getting hopper to move?
- What was difficult about moving hopper to 50 cm when programming with seconds?

4. Elaborate

Investigate moving two motors using distance and turning.

Introduce three new lines of code to students:

```
from hub import port
```

```
import runloop
```

```
import motor_pair
```

```
async def main():
```

```
    # Pair motors on port A and B
```

```
    motor_pair.pair(motor_pair.PAIR_1, port.E, port.F)
```

```
    # Turn for different amounts
```

```
    await motor_pair.move_for_time(motor_pair.PAIR_1, 3000, -20, velocity=280)
```

```
    await motor_pair.move_for_time(motor_pair.PAIR_1, 3000, 90, velocity=280)
```

```
    await motor_pair.move_tank_for_time(motor_pair.PAIR_1, 500, 1000, 3000)
```

```
runloop.run(main())
```

Allow students time to explore these new lines of code and investigate how they can move their hopper model in various ways.

Discuss with students how they were able to move their models and review the different code combinations together.

Ask students questions like:

- How did each line of code work?
- How did your program allow the motor to move in a different direction?
- What do the two numbers after start represent?
- What are different ways you can set the speed of the motors?
- What happened when you set the motors to move for cm?
- When might you use these types of codes?

Point out to students that the numbers used in the line `motor_pair.move_for_time` are for time, steering and velocity respectively. All numbers are integer types meaning that only whole numbers can be used.

Review this line of code specifically with students to discuss how to move the motors differently. 500 is the velocity of the left motor and 1000 is the velocity of the right motor. 3000 is the amount of time the hopper will run.

```
await motor_pair.move_tank_for_time(motor_pair.PAIR_1, 500, 1000, 3000)
```

5. Evaluate

Teacher Observation:

Discuss the program with students.

Ask students questions like:

- How were you able to program multiple motors to move the hopper model in a variety of ways?
- How does the design of the hopper model determine how it can move?
- What was difficult about programming hopper?

Self-Assessment:

Have students answer the following in their journals:

- What did you learn today about programming multiple motors?
- How does the design of a robot determine how it moves?
- What characteristics of a good teammate did I display today?
- Ask students to rate themselves on a scale of 1-3, on their time management today.

Ask students to rate themselves on a scale of 1-3, on their materials (parts) management today.

Race Day

Grade 6-9

45 minutes

Beginner

Race Day

Students program their model to move through a course considering the best use of motor controls.

Questions to investigate

How does the task to complete determine the way a robot needs to move? Why is documenting moves important?

Materials needed

- SPIKE Prime sets
- Devices with the SPIKE App installed.
- Student journals

Prepare

- Ensure SPIKE Prime hubs are charged, especially if connecting through Bluetooth.
- Ensure students have built the Hopper model, which was used in the Hopper Run lesson.

1. Engage

Ignite a discuss with students about different types of racetracks – tracks for cars, tracks for people, tracks for bicycles, etc. There are many different types of tracks that you can race across. Consider showing videos or images of different types of racetracks.

Challenge students to create a track for the race that includes straight areas and turns.

Each group should design a basic racecourse for Hopper to move through. The course should have at least 5 steps including straight moves and turns.

2. Explore

Students will program their Hopper to run the course.

KEY OBJECTIVES

Students will:

- Create a program to move through a series of steps and turns.
- Utilize motor pair in multiple ways.

STANDARDS

CSTA

2-CS-02 Design projects that combine hardware and software components to collect and exchange data.

2-AP-10 Use flowcharts and/or pseudocode to address complex problems as algorithms

2-AP-13 Decompose problems and subproblems into parts to facilitate the design, implementation, and review of programs.

2-AP-16 Incorporate existing code, media, and libraries into original programs, and give attribution.

2-AP-17 Systematically test and refine programs using a range of test cases.

2-AP-19 Document programs in order to make them easier to follow, test, and debug.

VOCABULARY

Discuss with students how Hopper can be programmed to move in the needed ways to run along the trail. Identify different ways that they could get the motors to move without changing the design of the model.

Prompt students to think of different ways that each motor can move to make the model go straight or turn.

- For example, motor E might move quickly while motor F moves slowly using a Motor Pair tank move.
- For example, one motor could move using degrees while the other is stopped.

Ask students to write a pseudocode program to explain the needed steps and programming elements to complete the race.

Open a new project in the Python programming canvas.

- Ask students to erase any code that is already in the programming area.

For this challenge, tell students to include a code comment using the # in their code for each step of the track moves to explain the movement of the model (i.e. moving straight, way it turns, moving backwards).

Encourage students to complete one step of the race at a time. Testing and iterating on the program will be important during this challenge. Remind students to watch their console for error messages and to reference the **Knowledge Base** as needed for help.

3. Explain

Allow students to share their final programs and how they programmed Hopper.

Ask students

- How did you program your Hopper to move through the different parts of the race?
- What debugging issues did you have? Did you have any error messages during programming?
- What was difficult about this challenge?

Ask students to review the code comments used in their program to determine if the code was documented well and is easy to follow. Discuss several examples as a class to think about best practices in documenting.

4. Elaborate

Allow students to try racing on other courses developed by other groups.

Students will need to create a new program in order to complete the new course.

Remind students to test their program several times in order to ensure it

moves as expected from the pseudocode. Ask students to add code comments using # to explain the steps of the program.

5. Evaluate

Teacher Observation:

Discuss the program with students.

Ask students questions like:

- How did you approach programming each step for the challenge?
- Why was documenting each step of your program important?
- Why is testing your programming at each step important?

Self-Assessment:

Have students answer the following in their journals:

- Why is checking each step of a long process one way to save time debugging?
- What was difficult about this challenge?
- What characteristics of a good teammate did I display today?
- Ask students to rate themselves on a scale of 1-3, on their time management today.
- Ask students to rate themselves on a scale of 1-3, on their materials (parts) management today.

Ideas to Help with Race Day

Grade 6-9

30-45 min.

Intermediate

Ideas to Help

Practice giving and using feedback from others.

Questions to investigate

- How can input from others help me make a better design and program?

Materials needed

- SPIKE Prime sets ready for student use. Prior to the first lesson, please visit the following website for help with set up, kit organization and SPIKE App <https://education.lego.com/en-us/start/spike-prime/intro>
- Devices with the SPIKE App installed
- Student journals
- Models from the Leading the Team lesson

Prepare

- Ensure SPIKE Prime hubs are charged, especially if connecting through Bluetooth.
- Ensure students have their built model and race course from the Race Day lesson.

1. Engage

Review the model for providing feedback with students.

Explain to students the following guidelines for giving feedback. Consider posting the guidelines for student reference.

- Feedback is not doing something for someone else.
- You should not rebuild a model for someone else.
- You should not type into someone's program.
- You should ask questions of each other.
- You should share your ideas and show your own programming, explaining why and how you did something.
- You should be encouraging and helpful to others and not provide negative or mean comments.

KEY OBJECTIVES

Students will:

- Give specific feedback on a peer's project.
- Explore how to use feedback to improve a project.

STANDARDS

1B-IC-20 Seek diverse perspectives for the purpose of improving computational artifacts.

VOCABULARY

Feedback, Specific, Positive, Negative

2. Explore

Have students work together to provide feedback to each other about the Race Day program.

Have two teams work together to provide feedback to each other. Teachers should model the process and what specific feedback looks and sounds like.

Review the procedure with students. Then have students take turns providing feedback.

- Team B will show their working model and explain their program.
- Team A provides feedback while Team B takes notes in their journal.
- Then teams can switch roles. Team A will show their working model and program while taking notes and listening to the feedback from Team B.

Feedback should include:

1. Tell something they really like. This could be the model, program, or design.
2. Tell something that worked well.
3. Share something the group could try differently.
4. Share anything that is confusing, did not work or that could be improved,
 - Remind students to be kind and clear in explaining why it is not clear or could be improved.
 - Let the team receiving the feedback ask questions as needed for more clarity.
 - The team giving feedback can also share ideas for improvement.

Teacher tip – Model providing feedback for the class frequently to help them learn to use positive language instead of negative language when providing feedback. Also practice taking feedback and thinking about how to use it rather than becoming defensive.

3. Explain

Have students discuss what they learned from their feedback session.

Ask students questions like:

- What did you notice in program that worked well?
- What ideas did you get from others?
- What is something you can do with your feedback?

4. Elaborate

Students should incorporate the feedback they were given.

Give students time to modify their program based on the feedback they received. Have students document their changes in their journal.

Allow students to share their updated their programs. Ask students to share what changes they incorporated and how they were able to make the changes.

5. Evaluate

Teacher Observation:

Discuss the program with students.

Ask students questions like:

- How did you use the feedback given?
- How did it feel to give feedback to others? And to receive it?
- How did you work to provide good feedback today?

Self-Assessment:

Have students answer the following in their journals:

- What did you learn today about providing good feedback?
- What did you learn today about how feedback can help in your work?
- What characteristics of a good teammate did I display today?
- Ask students to rate themselves on a scale of 1-3, on their time management today.
- Ask students to rate themselves on a scale of 1-3, on their materials (parts) management today.

Sensing Trouble: Exploring Sensors

A LEGO® Education Unit

Unit Introduction

This unit allows students to explore essential computer science principles and programming concepts of the text-based coding language, Python, to create safe movements in their robots. Students will investigate how sensors take in information and how to program the robot to act based on this information. The lessons are designed in an order that allows students to progress in their skills and knowledge in the following areas:

- Program a force sensor and investigate ways to use this type of sensor
- Program a distance sensor and investigate ways to use this type of sensor
- Investigate how a robot's movements affects the accuracy of sensors
- Decide on appropriate sensors to use for given situations
- Use and modify existing code to explore new ideas
- Create pseudocode to support creating algorithms
- Utilize code comment features to documents parts of a program
- Define and decompose a problem

Unit Learning Promise

In this unit, students will explore how to control sensors and understand how sensors are used to detect and provide information to create safe movements. Students will experience how to effectively utilize and program sensors in robots. Students will utilize pseudocode to support creating algorithms and code comments to document their programs.

Investigation Questions:

How can sensors provide information to make decisions?

How can sensors provide safety? When are sensors appropriate to use?

Unit Lessons

Lesson 1	Lesson 2	Lesson 3	Lesson 4	Lesson 5	Lesson 6
Start Sensing	Charging Rhino	Cart Control	Safe Delivery	Grasshopper Troubles	Ideas to Help Your Grasshopper
Time: 45 min.	Time: 45 min.	Time: 45 min.	Time: 45 min.	Time: 45 min.	Time: 45 min.

Assessment: We recommend assessing students on various skills throughout the unit.

- Use the progression of lessons as an opportunity to provide on-going feedback to prepare students for success for the open-ended project at the end of the unit.
- Each lesson includes a recommendation for teacher observations, student self-assessment, evaluation of success.

Unit Standards

CSTA
2-CS-02 Design projects that combine hardware and software components to collect and exchange data.
2-AP-10 Use flowcharts and/or pseudocode to address complex problems as algorithms.
2-AP-13 Decompose problems and subproblems into parts to facilitate the design, implementation, and review of programs.
2-AP-16 Incorporate existing code, media, and libraries into original programs, and give attribution.
2-AP-17 Systematically test and refine programs using a range of test cases.
2-AP-19 Document programs in order to make them easier to follow, test, and debug.
1B-IC-20 Seek diverse perspectives for the purpose of improving computational artifacts.

Integrated Standards

NGSS		
MS-ETS1-4 Develop a model to generate data for iterative testing and modification of a proposed object, tool or process such that an optimal design can be achieved.		
Common Core English Language Arts (ELA)		
6 th Grade	7 th Grade	8 th Grade
SL.6.1 Engage effectively in a range of collaborative discussions (one-on-one, in groups, and teacher-led) with diverse partners on grade 6 topics, texts, and issues, building on others' ideas and expressing their own clearly	SL.7.1 Engage effectively in a range of collaborative discussions (one-on-one, in groups, and teacher-led) with diverse partners on grade 7 topics, texts, and issues, building on others' ideas and expressing their own clearly	SL.8.1 Engage effectively in a range of collaborative discussions (one-on-one, in groups, and teacher-led) with diverse partners on grade 8 topics, texts, and issues, building on others' ideas and expressing their own clearly
SL.6.2 Interpret information presented in diverse media and formats (e.g., visually, quantitatively, orally) and explain how it contributes to a topic, text, or issue under study	SL.7.2 Analyze the main ideas and supporting details presented in diverse media and formats (e.g., visually, quantitatively, orally) and explain how the ideas clarify a topic, text, or issue under study	SL.8.2 Analyze the purpose of information presented in diverse media and formats (e.g., visually, quantitatively, orally) and evaluate the motives (e.g., social, commercial, political) behind its presentation
SL.6.4 Present claims and findings, sequencing ideas logically and using pertinent descriptions, facts, and details to accentuate main ideas or themes; use appropriate eye contact, adequate volume, and clear pronunciation	SL.7.4 Present claims and findings, emphasizing salient points in a focused, coherent manner with pertinent descriptions, facts, details, and examples; use appropriate eye contact, adequate volume, and clear pronunciation	SL.8.4 Present claims and findings, emphasizing salient points in a focused, coherent manner with relevant evidence, sound valid reasoning, and well-chosen details; use appropriate eye contact, adequate volume, and clear pronunciation
RST.6-8.3 Follow precisely a multistep procedure when carrying out experiments, taking measurements, or performing technical tasks	RST.6-8.3 Follow precisely a multistep procedure when carrying out experiments, taking measurements, or performing technical tasks	<u>RST.6-8.3</u> Follow precisely a multistep procedure when carrying out experiments, taking measurements, or performing technical tasks.
<u>L.6.6</u> Acquire and use accurately grade-appropriate general academic and domain-specific words and phrases; gather vocabulary knowledge when	L.7.6 Acquire and use accurately grade-appropriate general academic and domain-specific words and phrases; gather vocabulary knowledge when	L.8.6 Acquire and use accurately grade-appropriate general academic and domain-specific words and phrases; gather vocabulary knowledge when

considering a word or phrase important to comprehension or expression

considering a word or phrase important to comprehension or expression

considering a word or phrase important to comprehension or expression

Start Sensing

Grade 6-8

45 minutes

Beginner

Start Sensing

Students explore conditional statements using the force sensor and how to program sensors.

Questions to investigate

- How can sensors interact with or control motors?

Materials needed

- SPIKE Prime sets ready for student use. Prior to the first lesson, please visit the following website for help with set up, kit organization and SPIKE App <https://education.lego.com/en-us/start/spike-prime/intro>
- Devices with the SPIKE App installed.
- Student journals

Prepare

Ensure SPIKE Prime hubs are charged, especially if connecting through Bluetooth.

1. Engage

Engage students in thinking about how sensors work. Have students play a game of freeze dance. Hold up a force sensor and explain to students that they can only move when you are holding the sensor button down. Be sure to make it obvious when you are pushing and releasing the button. Students can move in any way they like when it is time to dance but must freeze when you release the button.

Discuss with students how the sensor was used to provide information on how they should move. How do sensors work?

2. Explore

Students will explore working with sensors by coding the force sensor.

Direct students to the Getting Started section of the knowledge base in the SPIKE App. Here students can access **7. Sensor Control**. This gives information and an example of how you can use the force sensor with the SPIKE Prime.

KEY OBJECTIVES

Students will:

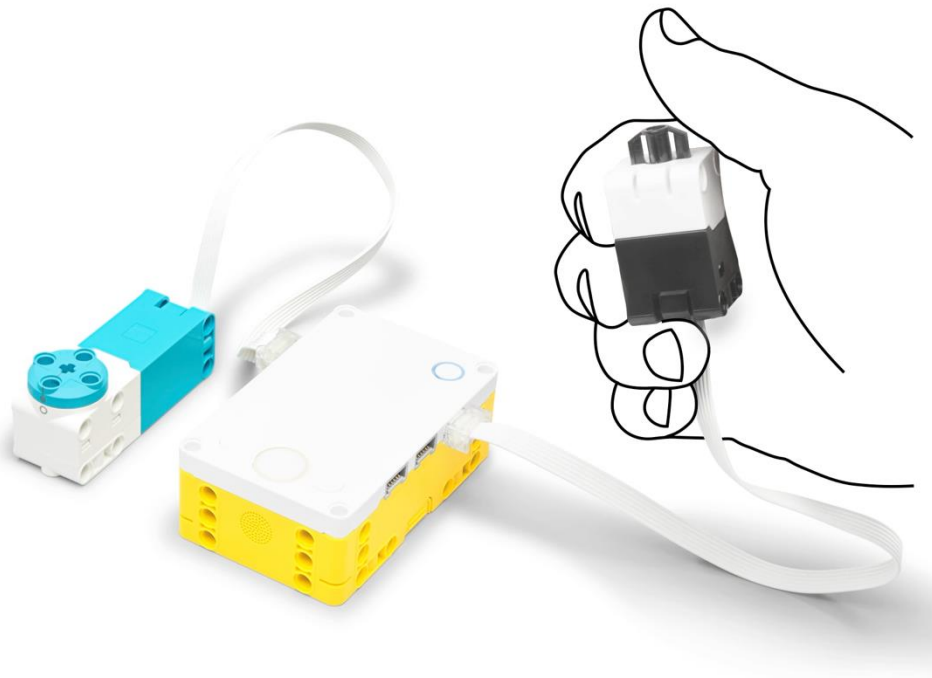
- Program the force sensor.
- Create conditional statements.

STANDARDS

CSTA

2-CS-02 Design projects that combine hardware and software components to collect and exchange data.
2-AP-10 Use flowcharts and/or pseudocode to address complex problems as algorithms
2-AP-16 Incorporate existing code, media, and libraries into original programs, and give attribution.
2-AP-17 Systematically test and refine programs using a range of test cases.
2-AP-19 Document programs in order to make them easier to follow, test, and debug.

Use this information to guide students in connecting the motor and force sensor to the hub.



Open a new project in the Python programming canvas. Ask students to erase any code that is already in the programming area. Students should connect their hub.

Push, Start, Stop

Sensors can be used in a variety of ways. The force sensor can be used like a "button" you can program to start and stop actions.

Brainstorm with students how they think you should program the sensor to work. What information will the software need to run the hardware correctly? Write a pseudocode program for making the force sensor work to start and stop the motor when pushed. Pseudocode is writing in words what you want the program to do.

An example could be:

- Import force sensor, motor, and hub
- Receive input from force sensor
- Motor turn on
- Move clockwise for time 2 seconds

Note: The code you write in steps (pseudocode) does not need to match exactly what will be put in the program. Pseudocode helps students think through what code is needed.

Provide students with the sample code to use the force sensor to stop and start the motor. Students can type this program into the programming canvas.

Ask students to run the program.

```
import runloop
import motor
import force_sensor
from hub import port

def is_force_sensor_pressed():
    # collect input from force sensor
    return force_sensor.pressed(port.B)

def is_force_sensor_not_pressed():
    # collect input from force sensor
    return not force_sensor.pressed(port.B)

async def main():
    # wait until the force sensor is pressed
    await runloop.until(is_force_sensor_pressed)

    motor.run(port.A, 750)
    # wait until the force sensor is not pressed
    await runloop.until(is_force_sensor_not_pressed)

    motor.stop(port.A)

runloop.run(main())
```

Discuss with students what happened when they ran the code. Students should identify that when the force sensor was pushed, the motor started running at 750 degrees per second. The motor continued to run until the force sensor was released, which caused the motor to stop. Review the program with the students to ensure they understand each line of code.

Point out to students where the force sensor input was stored. Explain to them that we need to be thoughtful of how to name parts of the program like functions. We are creating a function to tell the until what to wait for.

Prompt students to think about how to change this code to start the motor running, then stop it when the button is pushed. Ask students to change their code and run the program.

```
import runloop
import motor
import force_sensor
from hub import port

def is_force_sensor_pressed():
    # collect input from force sensor
    return force_sensor.pressed(port.B)

async def main():
    # turn on motor
    motor.run(port.A, 1000)

    # wait until the force sensor is pressed
    await runloop.until(is_force_sensor_pressed)

    # stop motor
    motor.stop(port.A)

runloop.run(main())
```

Note: Students need to change the code to move the motor.run line before the until line. This will start the motor running and then wait until the force sensor is pressed. Also notice that the function is force sensor not pressed is not needed.

Allow students additional time to explore with their force sensor and motor.

3. Explain

Have students share their new code and discuss how they used the force sensor.

Ask students questions like:

- How can you use a sensor to control the motors' actions?

- What are different ways that you can use or program a force sensor?
- Why do we not set a time or distance for the motor to move when we start the motor?
- What are some errors you think might occur with the force sensor?

4. Elaborate

Challenge students to try something new with their force sensor. Ask students to open the console. Students can print messages in the console using the `print()` function.

Introduce a new line of code to students. They can modify their previous code or type it in.

```
import runloop
import motor
import force_sensor
from hub import port

def is_force_sensor_pressed():
    # collect input from force sensor
    return force_sensor.pressed(port.B)

async def main():
    # turn on motor
    motor.run(port.A, 1000)

    # wait until the force sensor is pressed
    await runloop.until(is_force_sensor_pressed)

    # stop motor
    motor.stop(port.A)

    # print Hello! in the console
    print('Hello!')

runloop.run(main())
```

Discuss with students what happens when the new line of code `print('Hello!')` is run.

Now students can include messages with their physical robot as it moves. Students can use the `print` function to document what is happening in the program as an output. It can also be fun.

For debugging help, remind students that the printed messages have to be included in parenthesis () with quotation marks – (' ') or (" ") works.

Challenge students to create a question-and-answer game where one person pushes the button to answer a question and then the right answer pops up (or is printed) on screen.

5. Evaluate

Teacher Observation

Discuss the program with students.

- Ask students questions like:
 - What are some ways you were able to program the force sensor to work?
 - What are some ways that the sensor and motor can interact?
 - What are some ideas you have for using the print function?

Self-Assessment

Have students answer the following in their journals:

- What did you learn today about using a force sensor?
- When can you use the print function?
- What characteristics of a good teammate did I display today?
- Ask students to rate themselves on a scale of 1-3, on their time management today.
- Ask students to rate themselves on a scale of 1-3, on their materials (parts) management today.

Charging Rhino

Grade 6-8

45 minutes

Beginner

Charging Rhino

Students will explore using the force sensor to control movement.

Questions to investigate

How can a force sensor be used to help control movements and actions?

Materials needed

- SPIKE Prime sets ready for student use. Prior to the first lesson, please visit the following website for help with set up, kit organization and SPIKE App <https://education.lego.com/en-us/start/spike-prime/intro>
- Devices with the SPIKE App installed.
- Student journals

Prepare

Ensure SPIKE Prime hubs are charged, especially if connecting through Bluetooth.

1. Engage

Engage students in a discussion about how to stop a charging rhinoceros. Consider viewing videos that show a rhino when charging or running into an object. Discuss how the rhino moves and what it would take to stop it. Also consider discussing the senses that the rhino might use and how these might be similar to the sensors that are available for students to program.

2. Explore

Students will build a Rhino model to investigate different ways to move with the force sensor.

Direct students to the **BUILD** section in the SPIKE App. Here students can access the building instructions for the Rhino model. Ask students to build the model.

The building instructions are also available at

<https://education.lego.com/en-us/support/spike-prime/building-instructions>.

KEY OBJECTIVES

Students will:

- Explore the force sensor
- Understand effects of power on movement

STANDARDS

CSTA

- 2-AP-10 Use flowcharts and/or pseudocode to address complex problems as algorithms
- 2-AP-13 Decompose problems and subproblems into parts to facilitate the design, implementation, and review of programs.
- 2-AP-16 Incorporate existing code, media, and libraries into original programs, and give attribution.
- 2-AP-17 Systematically test and refine programs using a range of test cases.
- 2-AP-19 Document programs in order to make them easier to follow, test, and debug.

Direct students to open a new project in the Python programming canvas. Ask students to erase any code that is already in the programming area. Students should connect their hub.

Rhino Run

Challenge students to have their rhino run forward. Start students with this program.

```
import runloop
import motor_pair
from hub import port

async def main():
    # pair motors with the left motor on port B and right motor on port A
    motor_pair.pair(motor_pair.PAIR_1, port.B, port.A)
    # move motor pair straight forward
    motor_pair.move(motor_pair.PAIR_1, 0, velocity=750)

runloop.run(main())
```

Ask students to stop the program. Discuss the fact that there is nothing in the program that tells the Rhino to stop. Eventually, the Rhino runs into something.

Discuss ideas for making the rhino stop. One way is to add a stop to the program or set the start for a certain time, distance, etc. Another way is to use the force sensor attached as the Rhino's nose.

Prompt students to stop the charging Rhino when he runs into the wall using the force sensor. Students should position the Rhino facing the wall or another strong object a foot or more away. Share this sample program with students. Students will need to type this program into the programming canvas.

```
import runloop
import motor_pair
```

```

import force_sensor
from hub import port

def is_force_sensor_pressed():
    # collect input from force sensor
    return force_sensor.pressed(port.E)

async def main():
    # pair motors with the left motor on port B and right motor on port A
    motor_pair.pair(motor_pair.PAIR_1, port.B, port.A)

    # move motor pair straight forward
    motor_pair.move(motor_pair.PAIR_1, 0, velocity=750)

    # wait until the force sensor is pressed
    await runloop.until(is_force_sensor_pressed)

    # stop the rhino
    motor_pair.stop(motor_pair.PAIR_1)

runloop.run(main())

```

Note: Remind students to watch for errors in the console. Students can reference the line from the error message to pinpoint where a typing error might have occurred.

Ask students to run the program several more times. Each time, the students should move the Rhino model further away from the wall. Allow students to investigate how moving the model further back does not change the way the program runs.

3. Explain

Discuss with students how the Rhino model moved and review the code as a group.

Ask students questions like:

- How did the program work?
- What do the 0 and 750 represent in the motor_pair.move line of code?
- How did the force sensor work?
- What happened when you moved the Rhino further from the wall? Did running the model at different distances change the way the program worked?
- What was difficult about this challenge?

4. Elaborate

Challenge students to change their program to investigate how the Rhino stops when he is charging (moving fast) versus when he is moving slowly.

Ask students to run their program two more times. Once with a high velocity (velocity = 1000) and a low velocity (velocity = 250).

```
# move motor pair at high velocity
```

```
motor_pair.move(motor_pair.PAIR_1, 0, velocity=1000)
```

```
# move motor pair at low velocity
```

```
motor_pair.move(motor_pair.PAIR_1, 0, velocity=250)
```

Discuss what happens after each program is run. Students should note that the Rhino just stops, while the force sensor is still touching the object, when the velocity is set at 250. However, when the velocity is set to 1000, the Rhino hits the object and bounces back. Discuss as a group why this happens.

Allow students to build a small wall from the additional LEGO elements in their set. Challenge students to program the Rhino to charge through the wall and then play an appropriate sound. Students can also put a message in the console using the print() function to add the Rhino saying something as he hits the wall (ouch!).

Allow students to share and discuss their final programs.

5. Evaluate

Teacher Observation

Discuss the program with students.

Ask students questions like:

- How did your force sensor work to control your Rhino?
- Why does the power of the motor affect the way the model reacts when stopping with the force sensor?
- How did the force sensor provide information to the program to tell the Rhino what to do?

Self-Assessment

Have students answer the following in their journals:

- What did you learn today about using the force sensor to control your Rhino?
- What characteristics of a good teammate did I display today?
- Ask students to rate themselves on a scale of 1-3, on their time management today.
- Ask students to rate themselves on a scale of 1-3, on their materials (parts) management today.

Cart Control

Grade 6-8

45 minutes

Beginner

Cart Control

Students will explore the distance sensor to control movements.

Questions to investigate

- How can a distance sensor be used to provide information for decisions?
- How can sensors be used for more precise moves?

Materials needed

- SPIKE Prime sets ready for student use. Prior to the first lesson, please visit the following website for help with set up, kit organization and SPIKE App <https://education.lego.com/en-us/start/spike-prime/intro>
- Devices with the SPIKE App installed.
- Student journals

Prepare

- Ensure SPIKE Prime hubs are charged, especially if connecting through Bluetooth.

1. Engage

Engage students in thinking about ways we use automated machines. Prompt students to think about how these machines use sensors to keep from bumping into objects. Consider showing images and videos of different packing or shipping machines as examples. You might also prompt students to research and find their own examples.

Discuss with students how they see the automated machines moving around successfully.

KEY OBJECTIVES

Students will:

- Program the distance sensor.
- Explore movements with distance.
- Understand ultrasonic.

STANDARDS

CSTA

2-CS-02 Design projects that combine hardware and software components to collect and exchange data.
2-AP-10 Use flowcharts and/or pseudocode to address complex problems as algorithms.
2-AP-13 Decompose problems and subproblems into parts to facilitate the design, implementation, and review of programs.
2-AP-16 Incorporate existing code, media, and libraries into original programs, and give attribution.
2-AP-17 Systematically test and refine programs using a range of test cases.
2-AP-19 Document programs in order to make them easier to follow, test, and debug.

VOCABULARY

Float

Integer

2. Explore

Students will build a deliver cart model to investigate different ways to move with the distance sensor.

Direct students to the **BUILD** section in the SPIKE App. Here students can access the building instructions for the **Delivery Cart** model. Ask students to build the model. The building instructions are also available at <https://education.lego.com/en-us/support/spike-prime/building-instructions>.

Direct students to open a new project in the python programming canvas. Ask students to erase any code that is already in the programming area. Students should connect their hub.

Cart Under Control

Students will investigate how to use the distance sensor.

Ask students to review this program. Discuss with them if there are any bugs. When they realize there are not, ask them why this program is not going to work well. Students should realize that the motors will continue to run forever and eventually run into something.

```
import motor_pair
import runloop
from hub import port

async def main():
    # Pair motors on port A and B
    motor_pair.pair(motor_pair.PAIR_1, port.A, port.B)

    # Move straight at a specific velocity
    motor_pair.move(motor_pair.PAIR_1, 0, velocity=500)

runloop.run(main())
```

Prompt students to think about ways to keep the cart from bumping into an object. Students may think about including a stop in the program or adding a force sensor to the cart based on previous experiences. Tell students they will be

using a different sensor.

Ask students to locate the distance sensor they built at the bottom of their cart. Provide students with this sample code to move the cart using the distance sensor. Students will need to type this program into the programming canvas.

Ask students to go through each line of code with their partners. Determine what the program is intended to do.

Ask students to run the program.

```
import runloop
import motor_pair
import distance_sensor
from hub import port

def distance_sensor_greater_than():
    # return if the distance sensor is greater than 150 mm
    distance = distance_sensor.distance(port.B)
    return distance == -1 or distance > 150

def distance_sensor_less_than():
    #return if the distance sensor is less than 100 mm
    distance = distance_sensor.distance(port.B)
    return distance < 100 and distance > -1

async def main():
    # pair motors
    motor_pair.pair(motor_pair.PAIR_1, port.E, port.A)

    # wait until the distance sensor reads more than 150 mm
    await runloop.until(distance_sensor_greater_than) # turn on motor pair
    motor_pair.move(motor_pair.PAIR_1, 0, velocity=500)
```

```
# wait until the distance sensor reads less than 50 mm
```

```
await runloop.until(distance_sensor_less_than)
```

```
# stop motor
```

```
motor_pair.stop(motor_pair.PAIR_1)
```

```
runloop.run(main())
```

If students struggle to have the program work, give them a hint. Students will need to hold an object such as a book or a hand in front of the distance sensor when they start the program. Once the program is running take the object away, the cart should start moving. Place the object back in front of the sensor and it should stop. Make sure the object is farther away than 100 mm. Ask students to try the program multiple times with the delivery cart at different distances away from the object.

Allow the students to explore the program varying the starting distance from the object and re-running the program several times.

Note: The distance to start the motors does **not** have to be equal to the distance to stop the motors.

3. Explain

Review the program together as a group.

Ask students additional questions like:

- What are the lines `await runloop.until(distance_sensor_greater_than)` and `await runloop.until(distance_sensor_less_than)`
- telling the cart to do?
- Why was an object needed to make the program work?
- What happens when you start the cart further back from the object?
- How does the distance sensor work?

Explain to students that the distance sensor on the cart is an ultrasonic sensor. Have students look at the top left area of the programming canvas near where they connect their hub. Here they should see live data coming in from the cart. All hardware attached to the cart, motors, and sensors, will provide live data. This live data is in cm, but students should know that the sensor outputs in mm.

Ask students to look at the distance sensor and then move their hands close to it, then further away. The number should change, increasing as the hand moves

further away. Describe how the sensor works by sending a sonic pulse from one circle or "eye" which will bounce off any objects in front of the sensor and return to the other circle or "eye". The sensor uses the time it takes that pulse to return to "measure" the distance to the object. The sensor uses a mathematical formula to change the time into a distance measurement.

4. Elaborate

Students may have noticed that the rear wheels on the cart can keep the cart from moving in a straight line. When running the program above, students may have seen the cart turn to one side unless the rear wheels are straight. Consider demonstrating for students.

Ask students why a turning cart could be an issue for the distance sensor when it is detecting the distance to an object using an ultrasonic pulse. Discuss ideas with students about how to keep the cart moving straight.

Students should identify that because the rear wheels are attached to a motor, a piece of code can be added to help the cart stay straight. Students may also think to change the build to fix the motor in place. Consider discussing how that could limit the movement of the cart and re-focus students on the code instead.

Provide students with this sample code to set the back motor and wheels straight. Students should purposefully move the large motor with rear wheels to ensure it is not straight. Ask students to run the program.

```
import motor_pair
import runloop
import distance_sensor
import motor
from hub import port

def distance_sensor_closer_than():
    #return if the distance sensor is less than 150 mm
    distance = distance_sensor.distance(port.B)
    return distance < 150 and distance > -1
```

```

async def main():
    # Pair motors on port A and B
    motor_pair.pair(motor_pair.PAIR_1, port.E, port.A)

    # Move the rear motor to the 0 degree position to make the cart move straight
    await motor.run_to_absolute_position(port.C, 0, 250)

    # start motor pair moving
    motor_pair.move(motor_pair.PAIR_1, 0, velocity=200)

    # wait until the distance sensor is closer than 150 mm
    await runloop.until(distance_sensor_closer_than)

    # The cart will stop when the distance is less than 150 mm.
    motor_pair.stop(motor_pair.PAIR_1)

runloop.run(main())

```

Discuss the program with students and how ensuring the rear wheels being straight allows the sensor to work more effectively. When the sensor is perpendicular to an object, the sound waves bounce back more directly giving a good reading.

Debugging

Ask students to review the following program to consider how to address the error message received.

Debug activity #1

```

import motor_pair
import runloop
import motor
from hub import port

```

```

def distance_sensor_closer_than():
    # return if the distance sensor is less than 150 mm
    distance = distance_sensor.distance(port.B)
    return distance < 150 and distance > -1

async def main():
    # Pair motors on port A and B
    motor_pair.pair(motor_pair.PAIR_1, port.E, port.A)

    # Move the rear motor to the 0 degree position to make the cart move straight
    await motor.run_to_absolute_position(port.C, 0, 250)

    # start motor pair moving
    motor_pair.move(motor_pair.PAIR_1, 0, velocity=200)

    # wait until the distance sensor is closer than 150 mm
    await runloop.until(distance_sensor_closer_than)

    # The cart will stop when the distance is less than 150 mm.
    motor_pair.stop(motor_pair.PAIR_1)

runloop.run(main())

```

Traceback (most recent call last):

File "delivery cart new", line 25, in <module>

File "delivery cart new", line 20, in main

File "delivery cart new", line 6, in distance_sensor_closer_than

NameError: name 'distance_sensor' isn't defined

Discuss the error message with students. Students should recognize that the error refers to line 6. However, the error is in line 1. The issue is that a distance sensor is used in the program, but the distance sensor is not imported in line 1.

Debug activity #2

```
import motor_pair
import runloop
import distance_sensor
import motor
from hub import port

def distance_sensor_closer_than():
    #return if the distance sensor is less than 150 mm
    distance = distance_sensor.distance(port.B)
    return distance < 150 and distance > -1

async def main():
    # Pair motors on port A and B
    motor_pair.pair(motor_pair.PAIR_1, port.B, port.A)

    # Move the rear motor to the 0 degree position to make the cart move straight
    await motor.run_to_absolute_position(port.C, 0, 250)

    # start motor pair moving
    motor_pair.move(motor_pair.PAIR_1, 0, velocity=200)

    # wait until the distance sensor is closer than 150 mm
    await runloop.until(distance_sensor_closer_than)

    # The cart will stop when the distance is less than 150 mm.
    motor_pair.stop(motor_pair.PAIR_1)

runloop.run(main())
```

```
Traceback (most recent call last):
File "delivery cart new", line 25, in <module>
File "delivery cart new", line 11, in main
OSError: [Errno 19] ENODEV
```

Discuss the error message with students. Students should recognize that the error is pointing to line 11. The issue is that the MotorPair variable is not defined as a port that has a motor plugged in. Students should check the ports that the motors are plugged into and change the variable to set the correct ports.

5. Evaluate

Teacher Observation

Discuss the program with students. Ask students questions like:

- How did the distance sensor work to control your cart?
- How does the design of the rear wheel affect how the cart moves and potential the effectiveness of the sensor?
- What are some ways you could use the distance sensor?

Self-Assessment

Have students answer the following in their journals:

- What did you learn today about using the distance sensor to keep from bumping into objects?
- What characteristics of a good teammate did I display today?
- Ask students to rate themselves on a scale of 1-3, on their time management today.
- Ask students to rate themselves on a scale of 1-3, on their materials (parts) management today.

Safe Delivery

Grade 6-8

45 minutes

Beginner

Safe Delivery

Students will explore how motors and sensors work together.

Questions to investigate

- How can sensors be used to provide safety?

Materials needed

- SPIKE Prime sets ready for student use. Prior to the first lesson, please visit the following website for help with set up, kit organization and SPIKE App <https://education.lego.com/en-us/start/spike-prime/intro>
- Devices with the SPIKE App installed
- Delivery Cart model, which was used in the Cart Control lesson
- Student journals
- Ruler

Prepare

Ensure SPIKE Prime hubs are charged, especially if connecting through Bluetooth.

1. Engage

Kate and Kyle are working on adding a new treehouse to the playground. The delivery cart is coming with the needed materials, but Kyle is concerned the cart might damage the other items on the playground.

Ignite a discussion about how to help ensure the cart does not bump into any other materials on the playground. Consider showing an image of a playground as an example to support the discussion and later planning.

KEY OBJECTIVES

Students will:

- Program model to move safely using sensors.
- Investigate effects of motor power when using sensors.

STANDARDS

CSTA

2-CS-02 Design projects that combine hardware and software components to collect and exchange data.
2-AP-10 Use flowcharts and/or pseudocode to address complex problems as algorithms
2-AP-13 Decompose problems and subproblems into parts to facilitate the design, implementation, and review of programs.
2-AP-16 Incorporate existing code, media, and libraries into original programs, and give attribution.
2-AP-17 Systematically test and refine programs using a range of test cases.
2-AP-19 Document programs in order to make them easier to follow, test, and debug.

VOCABULARY

Float

Integer

2. Explore

Challenge students to investigate how the Cart can move more safely.

Discuss with students how the cart can move more safely through the playground. Ask students to create a small obstacle course with 2 or 3 items to mimic the playground using objects that are positioned high enough for the distance sensor to "see" them.

Ask students to write a program to allow the Cart to drive forward and stop before hitting an object, thus allowing the cart to move safely through the playground and stop in front of an object. Students should create the program to allow the cart to turn the motor on, until the distance sensor is closer than 15 cm to an object and then stop the motor.

Sample Code:

```
import motor_pair

import runloop

import distance_sensor

import motor

from hub import port

def distance_sensor_closer_than():
    #return if the distance sensor is less than 150 mm
    distance = distance_sensor.distance(port.B)
    return distance < 150 and distance > -1

async def main():
    # Pair motors on port A and B
    motor_pair.pair(motor_pair.PAIR_1, port.E, port.A)

    # Move the rear motor to the 0 degree position to make the cart move straight
    await motor.run_to_absolute_position(port.C, 0, 250)

    # start motor pair moving
    motor_pair.move(motor_pair.PAIR_1, 0, velocity=200)
```

```
# wait until the distance sensor is closer than 150 mm
await runloop.until(distance_sensor_closer_than)

# The cart will stop when the distance is less than 150 mm.
motor_pair.stop(motor_pair.PAIR_1)
```

```
runloop.run(main())
```

Have students create a chart to document each run like the one below.

	Velocity = 200	Velocity = 400	Velocity = 600	Velocity = 800	Velocity = 1000
Distance from object (mm)					

Students should run the program 5 times changing the speed for the motor each time. Students should run the program, then read the distance shown on the SPIKE App under the Distance Sensor. Allow students time to complete each trial and fill in the chart.

3. Explain

Discuss with students what they found when running the program at different velocity levels.

Discuss the difference between speed and power with students. Explain that when students are setting the velocity in the program what they are actually setting is the degrees per second the motor will rotate. This then determines how fast or at what speed the robot will travel.

Ask students questions like:

- How did the speed of the cart change when the power was increased?
- How did increasing the speed affect the way the cart is able to stop?
- Which power level or speed allows the cart to stop with the most accuracy?
- How should we consider power level or speed in creating the cart to move more safely?

4. Elaborate

Challenge students to move through the playground without hitting objects no matter where the cart enters.

Discuss with students how the cart needs to move through the playground to deliver the materials. The cart might enter from different spots. Students should try to run their program from different angles to make sure it can safely to move through the playground without hitting an object by using the distance sensor.

Note: The delivery cart cannot make sharp turns (like a 90-degree turn) but rather makes arc turns

Students should consider the angles the cart might need to move and change the position of the rear wheels as needed (i.e. not set to the 0 degrees position).

Allow students time to explore how to move safely through the playground. Discuss how the students were able to use the cart in the playground area.

Asking students questions like:

- How were you able to use the sensor to avoid hitting objects?
- How can sensors be used to add safety to moving objects?
- How did you use the additional motor in the back to help move the cart?
- What was difficult about this challenge?

Discuss different programs with students to prompt them to think about where they are relying on programming the motors to move and where they are using the sensors to decide.

5. Evaluate

Teacher Observation

Discuss the program with students. Ask students questions like:

- How did you program the sensor to allow the cart to move more safely?
- Why does the power of the motor or speed affect the way the model reacts when stopping with the distance sensor?

Self-Assessment

Have students answer the following in their journals:

- What did you learn today about setting the power level or speed of the motor when using the sensor to ensure safety and accuracy?
- What characteristics of a good teammate did I display today?
- Ask students to rate themselves on a scale of 1-3, on their time management today.
- Ask students to rate themselves on a scale of 1-3, on their materials (parts) management today.

Grasshopper Troubles

Grade 6-8

45 minutes

Beginner

Grasshopper Troubles

Students will investigate how to choose an appropriate sensor for a given task.

Questions to investigate

- When are sensors appropriate to use?
- How do you determine which sensor is best to use for a given task?

Materials needed

- SPIKE Prime sets ready for student use. Prior to the first lesson, please visit the following website for help with set up, kit organization and SPIKE App <https://education.lego.com/en-us/start/spike-prime/intro>
- Devices with the SPIKE App installed.
- Student journals

Prepare

Ensure SPIKE Prime hubs are charged, especially if connecting through Bluetooth.

1. Engage

Ignite a discussion with students about how to determine the best tool for a job. Provide students with several examples, which could be images, of tasks to complete. Tasks could range from digging a hole, to sharpening a pencil, to coding a robot. For each task, discuss the best tool for the job. For example, you would not code a robot with a shovel to sharpen a pencil. Try to give examples that might have more than one tool that can be used such as writing your name which could be done with a pencil, pen, marker, or crayon, but would be very difficult to do with a shovel, hammer, or watering can.

Discuss with students why it is important to make sure you have the right tool for a task.

KEY OBJECTIVES

Students will:

- Make appropriate hardware decisions
- Re-design a model to add a sensor

STANDARDS

CSTA

2-CS-02 Design projects that combine hardware and software components to collect and exchange data.

2-AP-13 Decompose problems and subproblems into parts to facilitate the design, implementation, and review of programs.

2-AP-17 Systematically test and refine programs using a range of test cases.

2-AP-19 Document programs in order to make them easier to follow, test, and debug.

2. Explore

Students will build a Hopper model to investigate different ways to move using a sensor.

Direct students to build the grasshopper model. The building instructions are available at <https://education.lego.com/en-us/support/spike-prime/building-instructions>.

Direct students to open a new project in the python programming canvas. Ask students to erase any code that is already in the programming area. Students should connect their hub.

Students should create a program that allows the Grasshopper Model to move forward at 50% power, which is a velocity of 500.

Sample code:

```
from hub import port
import runloop
import motor_pair

async def main():
    # Pair motors on port A and B
    motor_pair.pair(motor_pair.PAIR_1, port.E, port.F)

    # Move straight at 500 velocity
    motor_pair.move(motor_pair.PAIR_1, 0, velocity=500)

runloop.run(main())
```

Discuss with students how the Grasshopper model moves.

Ignite a discussion with students about the danger our Grasshopper faces when moving through the park. The Grasshopper does not want to run into a foot that might step on it. Have students think about which sensor, force or distance, would be best to keep the Grasshopper from running into an object. Allow students to discuss different ideas in their group to decide which sensor to use. Encourage students to carefully look at how the model currently functions.

Students need to attach their chosen sensor onto their grasshopper model **without** changing the basic design of the model. Allow students time to attach the sensor to their Grasshopper model.

Ask students to change their program to include the newly added sensor. Students should test their model to decide if the chosen sensor works. Remind students to test their program several times and change it as needed in order to ensure the model moves as expected. Ask students to add code comments using # to explain the steps of the program.

3. Explain

Allow students to share their final Grasshopper model designs and programs. Discuss with groups which sensor was chosen and why.

Ask students questions like:

- Which sensor did you choose and why?
- How well did the sensor perform? Did the Grasshopper model run into any objects?
- What issues did you have to overcome when adding the sensor?
- What were some issues you had to debug?
- What was difficult about this challenge?

Consider using a chart to track the sensors chosen. Was there a clear favorite?

4. Elaborate

Challenge students to change their programs to have the Grasshopper reverse direction or turn around instead of turning the motors off when it reaches the object. Students should consider what part of the code stays the same and what part of the code needs to be changed.

Ask students to plan how they want the Grasshopper to move. Have students write pseudocode before they change the code. Students should include the following in their new programs:

- Add an image on the hub when the grasshopper stops at the object
- Reverse direction or turn around to return to the starting position
- Play a new sound
- Include a second use of the sensor in the program somewhere

Optional: Remind students that they can also print a message in the console.

For this challenge, tell students to include a code comment using the # in their code for each step to explain the movement of the model (i.e. move straight, stop, move backwards).

Encourage students to complete one step at a time. Testing and iterating on the

program will be important during this challenge. Remind students to watch their console for error messages and to reference the **Knowledge Base** as needed for help.

Allow students to share and discuss their final programs.

5. Evaluate

Teacher Observation

Discuss the program with students.

- Ask students questions like:
 - How did you decide which sensor to use for the challenge?
 - How did you create your program to use the sensor in the most effective way?
 - How were you able to add to your program to allow the Grasshopper model to move away from the object safely?

Self-Assessment

Have students answer the following in their journals:

- What did you learn today about choosing an appropriate sensor for a task?
- What characteristics of a good teammate did I display today?
- Ask students to rate themselves on a scale of 1-3, on their time management today.
- Ask students to rate themselves on a scale of 1-3, on their materials (parts) management today.

Ideas to Help Your Grasshopper

Grade 6-8

30-45 min.

Intermediate

Ideas to Help Your Grasshopper

Practice giving and using feedback from others.

Questions to investigate

- How can input from others help me make a better design and program?

Materials needed

- SPIKE Prime sets ready for student use. Prior to the first lesson, please visit the following website for help with set up, kit organization and SPIKE App <https://education.lego.com/en-us/start/spike-prime/intro>
- Devices with the SPIKE App installed
- Student journals
- Models from the Grasshopper Trouble lesson

Prepare

- Ensure SPIKE Prime hubs are charged, especially if connecting through Bluetooth.
- Ensure students have their built model from the Grasshopper Trouble lesson.

1. Engage

Review the model for providing feedback with students.

Explain to students the following guidelines for giving feedback. Consider posting the guidelines for student reference.

- Feedback is not doing something for someone else.
- You should not rebuild a model for someone else.
- You should not type into someone's program.
- You should ask questions of each other.
- You should share your ideas and show your own programming, explaining why and how you did something.
- You should be encouraging and helpful to others and not provide negative or mean comments.

KEY OBJECTIVES

Students will:

- Give specific feedback on a peer's project.
- Explore how to use feedback to improve a project.

STANDARDS

1B-IC-20 Seek diverse perspectives for the purpose of improving computational artifacts.

VOCABULARY

Feedback, Specific, Positive, Negative

2. Explore

Have students work together to provide feedback to each other about the Grasshopper Trouble models.

Have two teams work together to provide feedback to each other. Teachers should model the process and what specific feedback looks and sounds like.

Review the procedure with students. Then have students take turns providing feedback.

- Team B will show their working model.
- Team A provides feedback while Team B takes notes in their journal.
- Then teams can switch roles. Team A will show their working model and take notes while Team B provides feedback.

Feedback should include:

1. Tell something they really like. This could be the model, program, or design.
2. Tell something that worked well.
3. Share something the group could try differently.
4. Share anything that is confusing, did not work or that could be improved,
 - Remind students to be kind and clear in explaining why it is not clear or could be improved.
 - Let the team receiving the feedback ask questions as needed for more clarity.
 - The team giving feedback can also share ideas for improvement.

Teacher tip – Model providing feedback for the class frequently to help them learn to use positive language instead of negative language when providing feedback. Also practice taking feedback and thinking about how to use it rather than becoming defensive.

3. Explain

Have students discuss what they learned from their feedback session.

Ask students questions like:

- What did you notice in models that worked well?
- What ideas did you get from others?
- What is something you can do with your feedback?

4. Elaborate

Students should incorporate the feedback they were given.

Give students time to modify their designs and program based on the feedback they received. Have students document their changes in their journal.

Allow students to share their updated models and programs. Ask students to share what changes they incorporated and how they were able to make the changes.

5. Evaluate

Teacher Observation:

Discuss the program with students.

Ask students questions like:

- How did you use the feedback given?
- How did it feel to give feedback to others? And to receive it?
- How did you work to provide good feedback today?

Self-Assessment:

Have students answer the following in their journals:

- What did you learn today about providing good feedback?
- What did you learn today about how feedback can help in your work?
- What characteristics of a good teammate did I display today?
- Ask students to rate themselves on a scale of 1-3, on their time management today.
- Ask students to rate themselves on a scale of 1-3, on their materials (parts) management today.

Doing Reps with Loops

A LEGO® Education Unit

Unit Introduction

This unit allows students to explore essential computer science principles and programming concepts of the text-based coding language, Python, through investigating ways to get moving and stretching. Students will investigate how to program various types of loops to allow their models to move in repeated patterns mimicking movements of stretching and exercising. The lessons are designed in an order that allows students to progress in their skills and knowledge in the following areas:

- Program for loops, while loops and infinite loops
- Explore setting variables and conditions in loops
- Investigate how a robot moves when using loops
- Use and modify existing code to explore new ideas
- Create pseudocode to support creating algorithms
- Utilize code comment features to documents parts of a program
- Define and decompose a problem

Unit Learning Promise

In this unit, students will explore how to program loops and develop an understanding for when to use each type of loop to get their models moving and stretching. Students will experience how to effectively utilize loops which includes thinking about setting variables and conditions in their loops. Students will utilize pseudocode to support creating algorithms and code comments to document their programs.

Investigation Questions:

How can repeated patterns be used in programming?

What are different ways to use loops in creating a program?

Unit Lessons

Lesson 1	Lesson 2	Lesson 3	Lesson 4	Lesson 5	Lesson 6	Lesson 7
Warm Ups with Leo	Counting Reps with Leo	Dance Loop with Coach	Setting Conditions for Yoga	Infinite Moves	Leading the Team with Loops	Ideas to Help: Feedback for Leading the Team with Loops
Time: 45 min.	Time: 45 min.	Time: 45 min.	Time: 45 min.	Time: 45 min.	Time: 90 min.	Time: 45 min.

Assessment: We recommend assessing students on various skills throughout the unit.

- Use the progression of lessons as an opportunity to provide on-going feedback to prepare students for success for the open-ended project at the end of the unit.
- Each lesson includes a recommendation for teacher observations, student self-assessment, evaluation of success.

Unit Standards

CSTA
2-CS-02 Design projects that combine hardware and software components to collect and exchange data.
2-AP-10 Use flowcharts and/or pseudocode to address complex problems as algorithms.
2-AP-13 Decompose problems and subproblems into parts to facilitate the design, implementation, and review of programs.
2-AP-16 Incorporate existing code, media, and libraries into original programs, and give attribution.
2-AP-17 Systematically test and refine programs using a range of test cases.
2-AP-19 Document programs in order to make them easier to follow, test, and debug.
1B-IC-20 Seek diverse perspectives for the purpose of improving computational artifacts.

Integrated Standards

NGSS		
MS-ETS1-4 Develop a model to generate data for iterative testing and modification of a proposed object, tool or process such that an optimal design can be achieved.		
Common Core English Language Arts (ELA)		
6 th Grade	7 th Grade	8 th Grade
SL.6.1 Engage effectively in a range of collaborative discussions (one-on-one, in groups, and teacher-led) with diverse partners on grade 6 topics, texts, and issues, building on others' ideas and expressing their own clearly	SL.7.1 Engage effectively in a range of collaborative discussions (one-on-one, in groups, and teacher-led) with diverse partners on grade 7 topics, texts, and issues, building on others' ideas and expressing their own clearly	SL.8.1 Engage effectively in a range of collaborative discussions (one-on-one, in groups, and teacher-led) with diverse partners on grade 8 topics, texts, and issues, building on others' ideas and expressing their own clearly
SL.6.2 Interpret information presented in diverse media and formats (e.g., visually, quantitatively, orally) and explain how it contributes to a topic, text, or issue under study	SL.7.2 Analyze the main ideas and supporting details presented in diverse media and formats (e.g., visually, quantitatively, orally) and explain how the ideas clarify a topic, text, or issue under study	SL.8.2 Analyze the purpose of information presented in diverse media and formats (e.g., visually, quantitatively, orally) and evaluate the motives (e.g., social, commercial, political) behind its presentation
SL.6.4 Present claims and findings, sequencing ideas logically and using pertinent descriptions, facts, and details to accentuate main ideas or themes; use appropriate eye contact, adequate volume, and clear pronunciation	SL.7.4 Present claims and findings, emphasizing salient points in a focused, coherent manner with pertinent descriptions, facts, details, and examples; use appropriate eye contact, adequate volume, and clear pronunciation	SL.8.4 Present claims and findings, emphasizing salient points in a focused, coherent manner with relevant evidence, sound valid reasoning, and well-chosen details; use appropriate eye contact, adequate volume, and clear pronunciation
RST.6-8.3 Follow precisely a multistep procedure when carrying out experiments, taking measurements, or performing technical tasks	RST.6-8.3 Follow precisely a multistep procedure when carrying out experiments, taking measurements, or performing technical tasks	<u>RST.6-8.3</u> Follow precisely a multistep procedure when carrying out experiments, taking measurements, or performing technical tasks.
<u>L.6.6</u> Acquire and use accurately grade-appropriate general academic and domain-specific words and phrases; gather vocabulary knowledge when	L.7.6 Acquire and use accurately grade-appropriate general academic and domain-specific words and phrases; gather vocabulary knowledge when	L.8.6 Acquire and use accurately grade-appropriate general academic and domain-specific words and phrases; gather vocabulary knowledge when

considering a word or phrase important to comprehension or expression

considering a word or phrase important to comprehension or expression

considering a word or phrase important to comprehension or expression

Warm Up Loop with Leo

Grade 6-8

45 minutes

Beginner

Warm Up Loop with Leo

Students will learn about loops and program with loops.

Questions to investigate

- How can lines of code be repeated while creating efficient programs?

Materials needed

- SPIKE Prime sets ready for student use. Prior to the first lesson, please visit the following website for help with set up, kit organization and SPIKE App <https://education.lego.com/en-us/start/spike-prime/intro>
- Devices with the SPIKE App installed.
- Student journals

Prepare

Ensure SPIKE Prime hubs are charged, especially if connecting through Bluetooth.

1. Engage

Have students stand and walk in a square around their chair, desk, or the room. As they walk, ask students to describe their individual movements (step forward, step forward, turn to the right/left, etc.).

Ignite a discussion with students around creating their movement into a program. Ask student to think about the lines of code needed to create this step-by-step program. Write the steps out together as a pseudocode. Pseudocode is writing in words what you want the program to do.

Ask students if they are excited to type out so many lines of code. Discuss alternatives to make the program more efficient (i.e. a shorter program that uses less lines of code to do an action or task).

Prompt students as needed to recognize the pattern that repeats within the steps.

KEY OBJECTIVES

Students will:

- Program with loops.
- Build and program a sit-up machine.

STANDARDS

CSTA

2-CS-02 Design projects that combine hardware and software components to collect and exchange data.

2-AP-10 Use flowcharts and/or pseudocode to address complex problems as algorithms

2-AP-13 Decompose problems and subproblems into parts to facilitate the design, implementation, and review of programs.

2-AP-16 Incorporate existing code, media, and libraries into original programs, and give attribution.

2-AP-17 Systematically test and refine programs using a range of test cases.

2-AP-19 Document programs in order to make them easier to follow, test, and debug.

Introduce the loop to students. Loops allow you to repeat lines of code, which shortens the program. This saves the programmer time and makes the code easier for the computer to run the program.

2. Explore

Students will explore working with for loops using the Leo, the trainer model.

Direct students to the **BUILD** section in the SPIKE App. Here students can access the building instruction for **Leo, the Trainer**. Ask students build the model.

You can also find the building instructions at <https://education.lego.com/en-us/support/spike-prime/building-instructions>.

Direct students to open a new project in the Python programming canvas. Ask students to erase any code that is already in the programming area. Students should connect their hub.

Start Moving

Challenge students to program Leo to complete one sit-up.

Ask students to program Leo to complete 1 sit-up. Encourage students to create the program on their own using notes and the Knowledge Base as a reference. Provide the sample code as needed if students struggle.

Sample Code:

```
from hub import port
import runloop
import motor_pair
async def main():

    # Pair motors on port B and F
    motor_pair.pair(motor_pair.PAIR_1, port.B, port.F)

    # Run motors one direction and then the other
    await motor_pair.move_for_time(motor_pair.PAIR_1, 500 ,0, velocity = 500)
    await motor_pair.move_for_time(motor_pair.PAIR_1, 500 ,0, velocity = -500)

runloop.run(main())
```

Discuss the program with students.

Ask students if they think one sit-up is enough of a workout for Leo. Prompt students to challenge Leo to complete at least 5 sit-ups. Discuss ways to change the program to repeat the same action several times.

Students might suggest copying and pasting the code five times. This is one way to do it, but not very efficient. Discuss efficiency as needed with students.

Prompt students to think about how we could use only one additional line of code to do the action. Share this code with students and see if students can identify the one new line of code. Ask students to add this line of code and run the program.

```
from hub import port
import motor_pair
import runloop

async def main():
    # Pair motors on port A and B
    motor_pair.pair(motor_pair.PAIR_1, port.B, port.F)

    for index in range(5):
        await motor_pair.move_for_time(motor_pair.PAIR_1, 500, 0, velocity = 500)
        await runloop.sleep_ms(500)
        await motor_pair.move_for_time(motor_pair.PAIR_1, 500, 0, velocity = -500)
        await runloop.sleep_ms(500)

runloop.run(main())
```

Discuss the program with students.

3. Explain

Discuss with students how the program worked.

Ask students questions like:

- How did the new program work?
- What does the "for index in range" line to tell the program?
- What is efficiency? How is using a for loop instead of copying and pasting the code more efficient?

Explain to students that this is an example of a "for loop" or repetitive statement, which allows us to repeat a portion of the code a set number of times. In this case, we repeated the code 5 times as indicated in the parentheses. Notice the lines of code following the for loop is indented. All lines of code indented after the for loop will be looped.

4. Elaborate

Challenge students to add to their program to include something for Leo to do after his sit-ups (i.e. after the loop).

Ask students to add to their code to include either an image or a sound and to print something in the console. Leo should still complete at least 5 sit-ups, but then do something new. Remind students that anything that do not want repeated in the loop will not be indented.

Note: students may change the power level of the motors. Changing the power level will change the way Leo moves. As time allows, consider investigating this together.

Allow students to share their final programs and discuss how they were able to add to their code. Specifically draw attention to where students use indentation and where they do not.

5. Evaluate

Teacher Observation:

Discuss the program with students.

Ask students questions like:

- How can you use a for loop to make a program more efficient? (shorter to code)
- How do you show in the program that the loop has ended?

Self-Assessment:

Have students answer the following in their journals:

- What did you learn today about using for loops in your program?
- What characteristics of a good teammate did I display today?
- Ask students to rate themselves on a scale of 1-3, on their time management today.
- Ask students to rate themselves on a scale of 1-3, on their materials (parts) management today.

Counting Reps with Leo

Grade 6-8

45 minutes

Beginner

Counting Reps with Leo

Students will apply for loops and learn to use the count variable.

Questions to investigate

- How can a robot count the number of times it performs a function?

Materials needed

- SPIKE Prime sets ready for student use. Prior to the first lesson, please visit the following website for help with set up, kit organization and SPIKE App <https://education.lego.com/en-us/start/spike-prime/intro>
- Devices with the SPIKE App installed.
- Student journals

Prepare

- Ensure SPIKE Prime hubs are charged, especially if connecting through Bluetooth.
- Ensure students have built the Leo, the Trainer model, which was used in the Warm Up Loop with Leo lesson.

1. Engage

Have students do a light exercise for one minute. This could be jumping jacks, toe touches, or anything that allows them to repeat an action several times within the minute. Do not ask them to count how many they do.

When the minute is finished, ask students to report how many of the exercise they were able to complete. Most students will not know because you did not tell them to count.

KEY OBJECTIVES

Students will:

- Program a sit-up machine to count the reps and to complete a count down.

STANDARDS

CSTA

2-CS-02 Design projects that combine hardware and software components to collect and exchange data.

2-AP-10 Use flowcharts and/or pseudocode to address complex problems as algorithms

2-AP-13 Decompose problems and subproblems into parts to facilitate the design, implementation, and review of programs.

2-AP-16 Incorporate existing code, media, and libraries into original programs, and give attribution.

2-AP-17 Systematically test and refine programs using a range of test cases.

2-AP-19 Document programs in order to make them easier to follow, test, and debug.

VOCABULARY

Counter variable

Have a short discussion about what we kind of information we could share if we knew how many each person did of the exercise. Examples might include a total number of the reps, create a comparison like boys to girls, or create a graph or a range of highest to lowest.

2. Explore

Students will explore working with a count variable with for loops.

Discuss with students how we can help Leo count his reps when doing his warm up exercises. Show students the sample code from the Warm Up Loop with Leo lesson to support your discussion.

Sample Code from Warm Up Loop with Leo:

```
from hub import port
import runloop
import motor_pair

async def main():
    # Pair motors on port A and B
    motor_pair.pair(motor_pair.PAIR_1, port.B, port.F)

    for index in range(5):
        await motor_pair.move_for_time(motor_pair.PAIR_1, 500 ,0, velocity = 500)
        await runloop.sleep_ms(500)

        await motor_pair.move_for_time(motor_pair.PAIR_1, 500 ,0, velocity = -500)
        await runloop.sleep_ms(500)

runloop.run(main())
```

Students may identify that the 5 included in the for index in range line of code tells us how many total sit-ups Leo will complete. Prompt students to think about how we can keep count and where the count could show up.

Explain to students that the word "index" in the "for count in range" line of code

is a **variable**. Students can assign any word or even just a letter to be the variable. This variable "counts" the number of times the code loops. The first loop is 0, then 1 and so on. The loop stops when it reaches the number in the parenthesis. In this case it is 5. So, the index counts 0, 1, 2, 3, 4. That makes 5 loops. The same name will need to be used consistently.

Show students this new program that includes a print function for the variable named index. By adding this new line of code, students can follow the "count" with Leo to know how many sit-ups are completed.

```
from hub import port
import runloop

import motor_pair

async def main():

    # Pair motors on port A and B
    motor_pair.pair(motor_pair.PAIR_1, port.B, port.F)

    for index in range(5):

        await motor_pair.move_for_time(motor_pair.PAIR_1, 500 ,0, velocity = 500)

        await runloop.sleep_ms(500)

        await motor_pair.move_for_time(motor_pair.PAIR_1, 500 ,0, velocity = -500)

        await runloop.sleep_ms(500)

    # The +1 is added since index starts at 0
    print(index + 1)

    print('Whew! That was tough!')

runloop.run(main())
```

Ask students to add these new lines of code into their program if they have the program saved from the Warm Up Loop with Leo Lesson. Otherwise, students will need to type this code in. Ensure students have their console open to watch the count.

3. Explain

Discuss the program with students. Identify how the print function is working to show a count of the sit-ups that Leo completes.

Ask students questions like:

- How did the program work?
- What is the counter variable and how does it work?
- What is the purpose of the line `print(count+1)`?
- Why is the last line "Whew! That was tough" not printed five times in the console?

Ensure students are comfortable with the counter variable, understanding that any amount can be assigned to the variable. The counter variable can be used in a variety of ways in the code including to assign the number of times the program loops as well as being used in the print function to show the value.

4. Elaborate

Challenge students to create a countdown for Leo.

Explain to students that sometime when exercising we count down to one instead of counting up. Ask students to change their code to count from five to one and then play a fun sound at the end to show they are finished.

Sample Code:

```
from hub import port
import runloop
import motor_pair

async def main():
    # Pair motors on port A and B
    motor_pair.pair(motor_pair.PAIR_1, port.B, port.F)

    for index in range(5):
        await motor_pair.move_for_time(motor_pair.PAIR_1, 500, 0, velocity = 500)
        await runloop.sleep_ms(500)

    await motor_pair.move_for_time(motor_pair.PAIR_1, 500, 0, velocity = -500)
```

```
await runloop.sleep_ms(500)

# The +1 is added since index starts at 0

print(5 - index)

print('Whew! That was tough!')

runloop.run(main())
```

Allow students additional time to explore how to use the counter variable in their program. Prompt students with ideas like how to add something motivational after each rep or count by 5's.

Allow students to share their final programs. Discuss different ways that students used their counter variable.

5. Evaluate

Teacher Observation:

Discuss the program with students. Ask students questions like:

- What are different ways that you could use the counter variable?
- How can a loop be embedded or included in a program?
- After the loop is completed, how can the program continue?
- What are different ways you could use a for loop in programs?

Self-Assessment:

Have students answer the following in their journals:

- What did you learn today about using a counter variable?
- What characteristics of a good teammate did I display today?
- Ask students to rate themselves on a scale of 1-3, on their time management today.
- Ask students to rate themselves on a scale of 1-3, on their materials (parts) management today.

Dance Loop with Coach

Grade 6-8

45 minutes

Beginner

Dance Loop with Coach

Students will practice programming for loops.

Questions to investigate

- What are ways to make programs more efficient?

Materials needed

- SPIKE Prime sets ready for student use. Prior to the first lesson, please visit the following website for help with set up, kit organization and SPIKE App <https://education.lego.com/en-us/start/spike-prime/intro>
- Devices with the SPIKE App installed.
- Student journals

Prepare

Ensure SPIKE Prime hubs are charged, especially if connecting through Bluetooth.

1. Engage

Have students stand with feet together. Ask them to all pick up their right foot and move their bodies slightly right, putting their right foot down. Next, pick up their left foot and move their bodies slightly left, putting their left foot down. Have the class work together and get everyone to move together 5 times in each direction.

Ask students to write a pseudocode for the actions just taken, using a for loop.

2. Explore

Students will explore programming the Coach model to move using a for loop.

Direct students to build the Coach model using the building instructions provided at <https://education.lego.com/en-us/support/spike-prime/building-instructions>.

KEY OBJECTIVES

Students will:

- Program a model using for loops.
- Debug four programs to learn tips and tricks.

STANDARDS

CSTA

2-CS-02 Design projects that combine hardware and software components to collect and exchange data.
2-AP-10 Use flowcharts and/or pseudocode to address complex problems as algorithms
2-AP-13 Decompose problems and subproblems into parts to facilitate the design, implementation, and review of programs.
2-AP-16 Incorporate existing code, media, and libraries into original programs, and give attribution.
2-AP-17 Systematically test and refine programs using a range of test cases.
2-AP-19 Document programs in order to make them easier to follow, test, and debug.

Direct students to open a new project in the python programming canvas. Ask students to erase any code that is already in the programming area. Students should connect their hub.

Start Moving

Ask students to examine the way the Coach model is built and brainstorm how the model can be programmed to lead a workout or dance (move back and forth) for us.

Provide students with this sample program to discuss making the Coach model move to lead us in an exercise. Ask students to explain the program and make suggestions for making the program more efficient.

Sample program:

```
from hub import port
import runloop
import motor

async def main():
    # Run at 250 velocity for 1/4 second
    await motor.run_for_time(port.B, 250, 300)
    await motor.run_for_time(port.F, 250, 300)

    await motor.run_for_time(port.B, 250, -300)
    await motor.run_for_time(port.F, 250, -300)

    await motor.run_for_time(port.B, 250, 300)
    await motor.run_for_time(port.F, 250, 300)

    await motor.run_for_time(port.B, 250, -300)
    await motor.run_for_time(port.F, 250, -300)

runloop.run(main())
```

Prompt students as needed to change the code from repeating several lines to including a loop. Students will need to decide how many times they want to repeat the loop.

Point out to students that in this workout, the coach wants us to move one foot (motor) and then the other motor. Remind students they need to be thoughtful to name each motor something specific in order to distinguish between the two. Naming variables up front is an important step to easily reuse that information throughout the program. Discuss ideas for naming the motor variables such as using the port each motor is plugged into as seen in the sample program.

Allow time for students to modify and test their program to allow their coach to move. If students need help, they should utilize the Knowledge Base and their notes. Remind them also to watch the console for any error messages they might receive.

If needed, provide this sample code to students.

```
from hub import port

import runloop

import motor

async def main():

    for index in range(10):

        # Run at 250 velocity for 1/4 second

        await motor.run_for_time(port.B, 250, 300)

        await motor.run_for_time(port.F, 250, 300)

        await motor.run_for_time(port.B, 250, -300)

        await motor.run_for_time(port.F, 250, -300)

runloop.run(main())
```

Allow students to explore the program, changing different values, to see how the coach model moves.

3. Explain

Discuss with students how the program worked.

Ask students questions like:

- How did you make your program more efficient?
- What does the "in range" part of the code tell the model to do?
- What happens when the program finishes the count?

4. Elaborate

Challenge students to recognize bugs in loop programs.

Show students each program and error message. Have students discuss what needs to change in each line of code to fix the bug. Consider making the changes as a class and ensuring the program runs correctly after each change.

Debug Activity 1:

Recognize a bug in the loop code.

```
from hub import port
import runloop
import motor

async def main():
    for index in range():
        # Run at 250 velocity for 1/4 second
        await motor.run_for_time(port.B, 250, 300)
        await motor.run_for_time(port.F, 250, 300)

        await motor.run_for_time(port.B, 250, -300)
        await motor.run_for_time(port.F, 250, -300)

runloop.run(main())
```

Traceback (most recent call last):

File "dance loop with coach", line 14, in <module>

File "dance loop with coach", line 6, in main

TypeError: function missing 1 required positional arguments

Students should recognize that the error is in line 6. The loop function is missing a required piece, the positional argument, which is number of times you want

the loop to run. Ask students to identify the number that should be included. While any number will work, students can use the comment as a hint.

Debug Activity 2:

Recognize a bug in the motor variable.

```
from hub import port
import runloop
import motor

async def main():
    for index in range(10):
        # Run at 250 velocity for 1/4 second
        await motor.run_for_time(250, 300)
        await motor.run_for_time(port.F, 250, 300)

        await motor.run_for_time(port.B, 250, -300)
        await motor.run_for_time(port.F, 250, -300)

runloop.run(main())
```

Traceback (most recent call last):

File "dance loop with coach", line 14, in <module>

File "dance loop with coach", line 8, in main

TypeError: function missing 1 required positional arguments

Students should recognize that the error is pointing to line 8 and saying that there is 1 missing argument. The motor port is not in the parenthesis.

Debug Activity 3:

Recognize a bug in the structure of the program.

```
from hub import port
import runloop
import motor

async def main():
    for index in range(10):
        # Run at 250 velocity for 1/4 second
        await motor.run_for_time(port.B, 250, 300)
        await motor.run_for_time(port.F, 250, 300)

        await motor.run_for_time(port.B, 250, -300)
        await motor.run_for_time(port.F, 250, -300)

runloop.run(main())
```

Traceback (most recent call last):

File "Project 2", line 7

SyntaxError: invalid syntax

Students should recognize that there is an error in line 8. The invalid syntax means that the program cannot recognize the command given to convert it. Prompt students to identify that we gave a for loop line of code, but then did not tell the loop what to do because we did not indent the next line of code. Every line that you want to include in the loop needs to be indented.

Students should also recognize that line 9 will need to be indented as well. Ask students to think about why no error message was given for line 9. The error is only given for the first line of code that creates a problem. Explain to students that this can mean that you miss one error while you are fixing another.

Debug Activity 4:

Another issue that can happen is creating an error that the program does not recognize as an error.

```
from spike import Motor

from hub import port

import runloop

import motor

async def main():

    for index in range(10):

        # Run at 250 velocity for 1/4 second

        await motor.run_for_time(port.B, 250, 300)

        await motor.run_for_time(port.F, 250, 300)

        await motor.run_for_time(port.B, 250, -300)

        await motor.run_for_time(port.F, 250, -300)

runloop.run(main())
```

Show students the code from Activity 3 with line 8 indented, i.e. the fix to the previous error. However, what happens if we do not indent line 9. Ask students to predict the new error that will occur.

Traceback (most recent call last):

File "dance loop with coach", line 11

IndentationError: unexpected indent

Show students the error message. While students might have expected an error in line 11 for not indenting, the actual error occurred in line 9. Ask students to think about what the program was doing. Explain that the program read the loop in line 6 which connected to the action to loop in line 8 (the connection indicated by the indentation). When line 9 was not indented, the program read that the

loop ended. However, when seeing an indentation in line 11 the program was confused. There was nothing requiring an indentation, so the program read an error.

Discuss the various error messages and ways to troubleshoot them together as a class.

5. Evaluate

Teacher Observation:

Discuss the program with students.

Ask students questions like:

- What are methods to make programs more efficient?
- How can loops be used?
- What actions should you take when you receive an error message?
- Why would engineers make a robot move repetitively?

Self-Assessment:

Have students answer the following in their journals:

- What did you learn today about creating programs with loops that are error free?
- What debugging tips did you learn today?
- What characteristics of a good teammate did I display today?
- Ask students to rate themselves on a scale of 1-3, on their time management today.
- Ask students to rate themselves on a scale of 1-3, on their materials (parts) management today.

Setting Conditions for Yoga

Grade 6-8

45-90 minutes

Intermediate

Setting Conditions for Yoga

Students will investigate using while loops.

Questions to investigate

- How can a loop be used only when certain conditions are met?

Materials needed

- SPIKE Prime sets ready for student use. Prior to the first lesson, please visit the following website for help with set up, kit organization and SPIKE App <https://education.lego.com/en-us/start/spike-prime/intro>
- Devices with the SPIKE App installed.
- Student journals

Prepare

- Ensure SPIKE Prime hubs are charged, especially if connecting through Bluetooth.
- Ensure students have the Coach model built, which was used in the Dance Loop with Coach lesson.

1. Engage

Engage students in thinking about how conditional statements work.

Play a game of *Teacher Says* with students.

Rules of the game:

- Students have to follow the actions given by the teacher, but only when you say *Teacher Says* first.
- For example, when you say "*Teacher Says jump*", students would jump.
- If you just say "*jump*", then students would not jump because you did not say *Teacher Says* first.

Play a few rounds of *Teacher Says* with students.

KEY OBJECTIVES

Students will:

- Investigate while statements.
- Program a model using while loops.

STANDARDS

CSTA

2-CS-02 Design projects that combine hardware and software components to collect and exchange data.
2-AP-10 Use flowcharts and/or pseudocode to address complex problems as algorithms
2-AP-13 Decompose problems and subproblems into parts to facilitate the design, implementation, and review of programs.
2-AP-16 Incorporate existing code, media, and libraries into original programs, and give attribution.
2-AP-17 Systematically test and refine programs using a range of test cases.
2-AP-19 Document programs in order to make them easier to follow, test, and debug.

Discuss with students the conditions set that indicate when to do the action. In the game, the condition for doing the action was *Teacher Says*. Students were only to do the act while the condition was true. For example, to add a bit of complexity using conditionals, you may try *Teacher Says if you have shoes on, jump. Teacher Says if you are wearing a t-shirt, raise your arms.*

Brainstorm other examples of when you do something based on a condition that is set (examples might be the weather deciding on your clothing, parents saying you can only have dessert after dinner, etc.). Discuss how you only do that action while the condition is true. For example, you generally only use an umbrella when it is raining. When it stops raining, usually you put the umbrella away. You stop the action when the condition is no longer true.

2. Explore

Students will explore programming the Coach model to move using a while loop.

Direct students to open a new project in the Python programming canvas. Ask students to erase any code that is already in the programming area. Students should connect their hub.

Following Directions

Discuss with students that the Coach wants to ensure we follow directions to only move when he/she tells us. The condition Coach has set for our yoga workout is any number less than 5. Explain to students that the model should only move when that condition is true.

Provide students with the sample code to move the motor and create a yoga pose. Ask students to type this program into the programming canvas. Ask students to run the program.

```
from hub import port
import runloop
import motor

async def main():
    count = 2
    #run motors when count is less than 5
    while count < 5:
        # Run at 300 velocity for 1/4 second
        await motor.run_for_time(port.B, 250, 300)
        await motor.run_for_time(port.F, 250, 300)
```

```
await motor.run_for_time(port.B, 250, -300)
await motor.run_for_time(port.F, 250, -300)
```

```
runloop.run(main())
```

Allow students time to run and explore the program. In the sample program, we included the count as 2 just as an example. The purpose of the count variable is to assign a number therefore any number can be included. Prompt students to put new numbers or values in for the count variable (count=10, count=2, etc.). Point out to the students that count is a variable for which they can set any value. We are naming this variable count here because we want to count our exercise. However, students could name the variable anything if they use the same name to reference it later.

3. Explain

Discuss with students how the program worked.

Ask students questions like:

- What was new in this program?
- What happens when you change the value assigned to the count variable? Higher than 5? Lower than 5?
- What is the "while" doing in the program?
- When does the program stop running?
- How is this an example of a loop?

Explain to students that the condition set in this program is for the motor to move only when the count variable is assigned a number less than 5. If the count is changed to 5 or higher than the model is not moving because we did not tell it to do anything else. The "while" is creating the conditional statement in the program to tell it only to work if the condition set is true.

To answer the last question, explain to students that while this is a good example of a conditional statement, it is not actually a loop right now. Nothing in the program is telling the action to repeat itself right now.

4. Elaborate

Challenge students to turn this program into a while loop.

Discuss ideas with students of how to change this into a while loop. Provide students with this sample code. Discuss the additional step added and how this will create a loop.

Ask students to run the program.

```
from hub import port
import runloop
import motor

async def main():
    count = 2
    #run motors when count is less than 5
    while count < 5:
        # Run at 250 velocity for 1 second
        await motor.run_for_time(port.B, 250, 300)
        await motor.run_for_time(port.F, 250, 300)

        await motor.run_for_time(port.B, 250, -300)
        await motor.run_for_time(port.F, 250, -300)

    count += 1

runloop.run(main())
```

Allow students time to explore the program. Students should try different values for the count variable to see how the program works. Consider re-visiting the same questions from the explain section for discussion.

Challenge students to change the program to set different values for the while count, which can also include a greater than and equal to.

5. Evaluate

Teacher Observation:

Discuss the program with students.

Ask students questions like:

- What happened when you added the count+1 at the end of the program?
- How do while loops work?

- When is it helpful to use a while loop in a program?

Self-Assessment:

Have students answer the following in their journals:

- What did you learn today about using while loops?
- What characteristics of a good teammate did I display today?
- Ask students to rate themselves on a scale of 1-3, on their time management today.
- Ask students to rate themselves on a scale of 1-3, on their materials (parts) management today.

Infinite Moves

Grade 6-8

45-90 minutes

Intermediate

Infinite Moves

Students will investigate using while loops.

Questions to investigate

- How can you create loops where the conditions always are true?

Materials needed

- SPIKE Prime sets ready for student use. Prior to the first lesson, please visit the following website for help with set up, kit organization and SPIKE App <https://education.lego.com/en-us/start/spike-prime/intro>
- Devices with the SPIKE App installed.
- Student journals

Prepare

- Ensure SPIKE Prime hubs are charged, especially if connecting through Bluetooth.
- Ensure students have built Coach model, which was used in the Dance Loop with Coach lesson.

1. Engage

Engage students in a conversation about setting conditions.

Prompt students to brainstorm examples of when they need to meet certain conditions. Easy examples can include deciding to eat only when hungry or cleaning your room only when someone tells you to. Have students name the conditions in each case.

Next, challenge students to find an example of a time when a condition is always true. Students should brainstorm ideas of a condition that could be set that would not stopping being true (no false for the condition). An example might include the sun rising each day. Prompt students to write a statement that sets sun rise as a condition. Example: If the sun rises, then I will see light outside. This is true every day because the sun rises every day.

KEY OBJECTIVES

Students will:

- Program infinite loops.
- Create a model that includes a force sensor that will provide a condition for the robot to move.

STANDARDS

CSTA

2-CS-02 Design projects that combine hardware and software components to collect and exchange data.
2-AP-10 Use flowcharts and/or pseudocode to address complex problems as algorithms
2-AP-13 Decompose problems and subproblems into parts to facilitate the design, implementation, and review of programs.
2-AP-16 Incorporate existing code, media, and libraries into original programs, and give attribution.
2-AP-17 Systematically test and refine programs using a range of test cases.
2-AP-19 Document programs in order to make them easier to follow, test, and debug.

Explain to students that a loop with a condition that is always true is an infinite loop. Infinite loops are while loops that can never become false.

2. Explore

Students will explore programming the Coach model to move using an infinite loop.

Direct students to open a new project in the Python programming canvas. Ask students to erase any code that is already in the programming area. Students should connect their hub.

Explain to students that Coach wants to take us on a long hike now that we are all warmed up. Discuss with students how we can include a loop for moving the Coach model using a while loop that sets a condition for the movement, but the condition is created in a way that it is always true (cannot be false).

Ask students to review their program from the Setting Condition for Yoga lesson. Ask students to brainstorm ways to set the conditions so that the while loop is always true and therefore does not end.

Sample Program from Setting Condition for Yoga:

```
from hub import port
import runloop
import motor

async def main():
    count = 2
    #run motors when count is less than 5
    while count < 5:
        # Run at 250 velocity for 1 second
        await motor.run_for_time(port.B, 250, 300)
        await motor.run_for_time(port.F, 250, 300)

        await motor.run_for_time(port.B, 250, -300)
        await motor.run_for_time(port.F, 250, -300)

    count += 1
```

```
runloop.run(main())
```

Allow students time to try several different ideas to ensure the condition is always true. Sample solutions include setting the count at 1 and the while count at greater than 0. This program will continue to run continuously because the condition will continue to be true.

Remind students to watch the console for errors as they are investigating.

While True

Introduce a new line of code to students that will allow them to accomplish the goal to create a condition for the loop that is always true (cannot become false).

Share this line of code with students and ask them where they think it belongs in the program they just created (what it should replace).

Ask students to change their program to include the while True line of code which will allow the program to loop infinitely.

Sample Program:

```
from hub import port
import runloop
import motor

async def main():
    while True:
        # Run at 250 velocity for 1 second
        await motor.run_for_time(port.B, 250, 300)
        await motor.run_for_time(port.F, 250, 300)

        await motor.run_for_time(port.B, 250, -300)
        await motor.run_for_time(port.F, 250, -300)
```

```
runloop.run(main())
```

3. Explain

Discuss with students how the program worked.

Ask students questions like:

- What was difficult about this challenge?
- How is this loop different than a while loop?
- When might you want to use an infinite loop?

4. Elaborate

Challenge students to add a force sensor to the Coach model.

Ask students to add a force sensor to their model by plugging the sensor into port D. Students do not need to attach the force sensor to the model.

With the sensor attached, ask students to update their programs to include using a sensor in the loop. The loop should continue to always be true, but also include using a force sensor.

Sample Program:

```
from hub import port
import runloop
import motor
import force_sensor

def is_pressed():
    return force_sensor.pressed(port.D)

async def main():
    while True:
        # when the force sensor is pressed, start coach
        await runloop.until(is_pressed)

        # Run at 300 velocity for 1/4 second
        await motor.run_for_time(port.B, 250, 300)
        await motor.run_for_time(port.F, 250, 300)
        await motor.run_for_time(port.B, 250, -300)
        await motor.run_for_time(port.F, 250, -300)

runloop.run(main())
```

Allow students time to test and modify their program. Remind students

to watch the console for error messages.

Allows students to share their programs. Discuss the different approaches as a group.

5. Evaluate

Teacher Observation:

Discuss the program with students.

Ask students questions like:

- What turns a while loop into an infinite loop?
- What is the most important thing to remember when creating an infinite loop?
- How can sensors be included in using loops?

Self-Assessment:

Have students answer the following in their journals:

- What did you learn today about using infinite loops?
- What characteristics of a good teammate did I display today?
- Ask students to rate themselves on a scale of 1-3, on their time management today.
- Ask students to rate themselves on a scale of 1-3, on their materials (parts) management today.

Leading the Team with Loops

Grade 6-8

90-120 minutes

Advanced

Leading the Team with Loops

Students will apply their knowledge of loops to create their own model and program it.

Questions to investigate

- How can a challenge determine the best type of loop to use in a program?

Materials needed

- SPIKE Prime sets
- Device with SPIKE App installed
- Student journal

Prepare

Check to make sure SPIKE Prime hubs are charged, especially if connecting through Bluetooth.

1. Engage

Leo the Trainer and the Coach are ready to hand over the team to you for leading the final exercise. Brainstorm different types of exercises, stretches, and yoga poses that would be a good cool down for the team.

Create a chart as a class to organize the different ideas by the types of movements. Consider a chart like this:

Body Part	Movement Examples
Arms	Push ups
Legs	Squats Warrior Pose
Core/Middle	Sit ups Plank

KEY OBJECTIVES

Students will:

- Design a model for repetition.
- Program a model to move using loops.

STANDARDS

CSTA

2-CS-02 Design projects that combine hardware and software components to collect and exchange data.

2-AP-13 Decompose problems and subproblems into parts to facilitate the design, implementation, and review of programs.

2-AP-16 Incorporate existing code, media, and libraries into original programs, and give attribution.

VOCABULARY

Infinite loop

2. Explore

Students will design and build a new trainer to lead their exercise. Prompt students to use the chart above to think about what they need to design and build to complete the exercise move.

Challenge students to think about the movements that will be needed to complete the exercise. Allow students time to investigate how to move properly to think about their use of motors in their design. Students should sketch their design.

Remind students to test their design several times in order to ensure the movement works. Students should consider tradeoffs in their design such as the need to design an entire person if only moving legs in order to spend time on the most important part of the design. Consider pointing students to the **BUILD** section of the SPIKE App for inspiration if needed.

Requirements for this challenge:

- The model must be able to complete a simulation of an exercise.
- The model should include at least one motor for movement.
- The program must include a type of loop.
- Prior to writing code, students must show a sketch of their design and write an explanation of what they want to the code.
- Students should use the comment feature in their Python code to explain what the lines of code are meant to do.

Allow students time to design and program their new trainer. Remind students to test their program several times and change it as needed in order to ensure the model moves as expected.

3. Explain

Students should share their design and explain how it works. Conduct an initial sharing session with students.

Ask students questions like:

- How did you program your model to complete your chosen exercise?
Ask students to share their program comments to explain.
- What decisions did you have to make while creating your design?
- What type of loop did you choose? Why this type and not others?
- What were areas that you had to debug or troubleshoot?
- What was difficult about this challenge?

4. Elaborate

Allow students additional time to complete their program after the initial sharing session.

Students should finalize their design and program. Encourage students to incorporate any new ideas they got from the sharing session.

Leave the models together if you are completing the lesson on feedback next.

5. Evaluate

Teacher Observation:

Discuss the program with students. Ask students questions like:

- What was difficult about this challenge?
- What was your approach to solving this challenge?
- What type of loops did you include and why?

Self-Assessment:

Have students answer the following in their journals:

- What did you learn today about designing a model to do a specific task?
- What characteristics of a good teammate did I display today?
- Ask students to rate themselves on a scale of 1-3, on their time management today.
- Ask students to rate themselves on a scale of 1-3, on their materials (parts) management today.

Ideas to Help: Feedback for Leading the Team with Loops

Grade 6-8

30-45 min.

Intermediate

Ideas to Help: Feedback for Leading the Team with Loops

Practice giving and using feedback from others.

Questions to investigate

- How can input from others help me make a better design and program?

Materials needed

- SPIKE Prime sets ready for student use. Prior to the first lesson, please visit the following website for help with set up, kit organization and SPIKE App <https://education.lego.com/en-us/start/spike-prime/intro>
- Devices with the SPIKE App installed
- Student journals
- Models from the Leading the Team with Loops lesson

Prepare

- Ensure SPIKE Prime hubs are charged, especially if connecting through Bluetooth.
- Ensure students have their built model from the Leading the Team with Loops lesson.

1. Engage

Review the model for providing feedback with students.

Explain to students the following guidelines for giving feedback. Consider posting the guidelines for student reference.

- Feedback is not doing something for someone else.
- You should not rebuild a model for someone else.
- You should not type into someone's program.
- You should ask questions of each other.

KEY OBJECTIVES

Students will:

- Give specific feedback on a peer's project.
- Explore how to use feedback to improve a project.

STANDARDS

1B-IC-20 Seek diverse perspectives for the purpose of improving computational artifacts.

VOCABULARY

Feedback, Specific, Positive, Negative

- You should share your ideas and show your own programming, explaining why and how you did something.
- You should be encouraging and helpful to others and not provide negative or mean comments.

2. Explore

Have students work together to provide feedback to each other about the Leading the Team with Loops models.

Have two teams work together to provide feedback to each other. Teachers should model the process and what specific feedback looks and sounds like.

Review the procedure with students. Then have students take turns providing feedback.

- Team B will show their working model.
- Team A provides feedback while Team B takes notes in their journal.
- Then teams can switch roles. Team A will show their working model and take notes while Team B provides feedback.

Feedback should include:

1. Tell something they really like. This could be the model, program, or design.
2. Tell something that worked well.
3. Share something the group could try differently.
4. Share anything that is confusing, did not work or that could be improved,
 - Remind students to be kind and clear in explaining why it is not clear or could be improved.
 - Let the team receiving the feedback ask questions as needed for more clarity.
 - The team giving feedback can also share ideas for improvement.

Teacher tip – Model providing feedback for the class frequently to help them learn to use positive language instead of negative language when providing feedback. Also practice taking feedback and thinking about how to use it rather than becoming defensive.

3. Explain

Have students discuss what they learned from their feedback session.

Ask students questions like:

- What did you notice in models that worked well?
- What ideas did you get from others?
- What is something you can do with your feedback?

4. Elaborate

Students should incorporate the feedback they were given.

Give students time to modify their designs and program based on the feedback they received. Have students document their changes in their journal.

Allow students to share their updated models and programs. Ask students to share what changes they incorporated and how they were able to make the changes.

5. Evaluate

Teacher Observation:

Discuss the program with students.

Ask students questions like:

- How did you use the feedback given?
- How did it feel to give feedback to others? And to receive it?
- How did you work to provide good feedback today?

Self-Assessment:

Have students answer the following in their journals:

- What did you learn today about providing good feedback?
- What did you learn today about how feedback can help in your work?
- What characteristics of a good teammate did I display today?
- Ask students to rate themselves on a scale of 1-3, on their time management today.
- Ask students to rate themselves on a scale of 1-3, on their materials (parts) management today.

Playing Games with Simple Conditions

A LEGO® Education Unit

Unit Introduction

This unit allows students to explore essential computer science principles and programming concepts of the text-based coding language, Python, in the context of playing different games. Students will investigate how to plan for and program conditional statements when creating their games to set how models move given certain conditions are met. The lessons are designed in an order that allows students to progress in their skills and knowledge in the following areas:

- Create programs using conditional statements
- Explore writing programs using if/else, if/elif/else, and while true statements
- Investigate how sensors can be used with conditional statements
- Use and modify existing code to explore new ideas
- Create flow charts to support creating conditional statements
- Utilize code comment features to document parts of a program
- Define and decompose a problem

Unit Learning Promise

In this unit, students will explore how to program loops and develop an understanding for when to use each type of loop as they create different games. Students will experience how to effectively utilize loops which includes thinking about setting variables and conditions in their loops. Students will utilize pseudocode to support creating algorithms and code comments to document their programs.

Investigation Questions:

How can you set conditions for how a sensor will respond?

How can you set multiple conditions in your program?

Unit Lessons

Lesson 1	Lesson 2	Lesson 3	Lesson 4	Lesson 5	Lesson 6	Lesson 7	Lesson 8
Controlling Motion with Tilt	Claw Machine	Charting Game Decisions	Guess Which Color	Guessing Game	Score!	Game Time	Ideas to Help with Game Time
Time: 45 min.	Time: 45 min.	Time: 45 min.	Time: 45 min.	Time: 45 min.	Time: 45 min.	Time: 45 min.	Time: 45 min.

Assessment: We recommend assessing students on various skills throughout the unit.

- Use the progression of lessons as an opportunity to provide on-going feedback to prepare students for success for the open-ended project at the end of the unit.
- Each lesson includes a recommendation for teacher observations, student self-assessment, evaluation of success.

Unit Standards

CSTA
2-CS-02 Design projects that combine hardware and software components to collect and exchange data.
2-AP-10 Use flowcharts and/or pseudocode to address complex problems as algorithms.
2-AP-13 Decompose problems and subproblems into parts to facilitate the design, implementation, and review of programs.
2-AP-16 Incorporate existing code, media, and libraries into original programs, and give attribution.
2-AP-17 Systematically test and refine programs using a range of test cases.
2-AP-19 Document programs in order to make them easier to follow, test, and debug.
1B-IC-20 Seek diverse perspectives for the purpose of improving computational artifacts.

Integrated Standards

NGSS		
MS-ETS1-4 Develop a model to generate data for iterative testing and modification of a proposed object, tool or process such that an optimal design can be achieved.		
Common Core English Language Arts (ELA)		
6 th Grade	7 th Grade	8 th Grade
SL.6.1 Engage effectively in a range of collaborative discussions (one-on-one, in groups, and teacher-led) with diverse partners on grade 6 topics, texts, and issues, building on others' ideas and expressing their own clearly	SL.7.1 Engage effectively in a range of collaborative discussions (one-on-one, in groups, and teacher-led) with diverse partners on grade 7 topics, texts, and issues, building on others' ideas and expressing their own clearly	SL.8.1 Engage effectively in a range of collaborative discussions (one-on-one, in groups, and teacher-led) with diverse partners on grade 8 topics, texts, and issues, building on others' ideas and expressing their own clearly
SL.6.2 Interpret information presented in diverse media and formats (e.g., visually, quantitatively, orally) and explain how it contributes to a topic, text, or issue under study	SL.7.2 Analyze the main ideas and supporting details presented in diverse media and formats (e.g., visually, quantitatively, orally) and explain how the ideas clarify a topic, text, or issue under study	SL.8.2 Analyze the purpose of information presented in diverse media and formats (e.g., visually, quantitatively, orally) and evaluate the motives (e.g., social, commercial, political) behind its presentation
SL.6.4 Present claims and findings, sequencing ideas logically and using pertinent descriptions, facts, and details to accentuate main ideas or themes; use appropriate eye contact, adequate volume, and clear pronunciation	SL.7.4 Present claims and findings, emphasizing salient points in a focused, coherent manner with pertinent descriptions, facts, details, and examples; use appropriate eye contact, adequate volume, and clear pronunciation	SL.8.4 Present claims and findings, emphasizing salient points in a focused, coherent manner with relevant evidence, sound valid reasoning, and well-chosen details; use appropriate eye contact, adequate volume, and clear pronunciation
RST.6-8.3 Follow precisely a multistep procedure when carrying out experiments, taking measurements, or performing technical tasks	RST.6-8.3 Follow precisely a multistep procedure when carrying out experiments, taking measurements, or performing technical tasks	<u>RST.6-8.3</u> Follow precisely a multistep procedure when carrying out experiments, taking measurements, or performing technical tasks.
<u>L.6.6</u> Acquire and use accurately grade-appropriate general academic and domain-specific words and phrases; gather vocabulary knowledge when	L.7.6 Acquire and use accurately grade-appropriate general academic and domain-specific words and phrases; gather vocabulary knowledge when	L.8.6 Acquire and use accurately grade-appropriate general academic and domain-specific words and phrases; gather vocabulary knowledge when

considering a word or phrase important to comprehension or expression

considering a word or phrase important to comprehension or expression

considering a word or phrase important to comprehension or expression

Controlling Motion with Tilt

Grade 6-8

45 minutes

Beginner



Controlling Motion with Tilt

Students will explore writing conditional statements in Python using the motion sensor.

Questions to investigate

- How can you set conditions for how a sensor will respond?
- How does a motion sensor work?

Materials needed

- SPIKE Prime sets ready for student use. Prior to the first lesson, please visit the following website for help with set up, kit organization and SPIKE App <https://education.lego.com/en-us/start/spike-prime/intro>
- Devices with the SPIKE App installed
- Student journals

Prepare

Ensure SPIKE Prime hubs are charged, especially if connecting through Bluetooth.

1. Engage

Engage students in an experience with conditional statements.

Students need the following elements from their set:

- Magenta frame (pictured here)
- 3 different yellow elements
- 3 different red elements



Ask students to place the magenta frame in the middle of a sheet of paper with the other elements set outside the frame.

Students will identify which elements meet the given condition by placing the elements that are true (meet condition) inside the frame and false (do not meet

KEY OBJECTIVES

Students will:

- Program the motion sensor.
- Create conditional statements.

STANDARDS

CSTA

2-CS-02 Design projects that combine hardware and software components to collect and exchange data.
2-AP-10 Use flowcharts and/or pseudocode to address complex problems as algorithms
2-AP-16 Incorporate existing code, media, and libraries into original programs, and give attribution.
2-AP-17 Systematically test and refine programs using a range of test cases.
2-AP-19 Document programs in order to make them easier to follow, test, and debug.

VOCABULARY

Conditional statement

Orientation

the condition) outside the frame.

Practice with students. Tell students the condition is yellow. Discuss with students which elements meet the conditions and ask students to place those elements inside the frame. All other pieces will remain outside the frame as false.

Empty the frame between rounds.

Round 1: The condition is red.

Round 2: The condition is larger than a 2x2 brick (students may need to compare).

Round 3: The condition is any element that is not a traditional brick shape (such as an axle or a technic piece).

Discuss the experience together as a whole group. Ask students what a conditional statement is. Explain that conditional statements are pieces of code that only run when the stated condition is met or true. The Boolean values, true and false, are used in conditional statements.

2. Explore

Students will explore creating simple conditional statements while also exploring how to use the motion sensor in the hub.

Direct students to get the hub from their set and open a new project in the Python programming canvas. Ask students to erase any code that is already in the programming area. Students should connect their hub.

Tap for Action

Sensors can be used in a variety of ways. The motion sensor can be programmed to take different actions based on the way you move it. Brainstorm with students how they think the hub can be moved when held in a hand. Ask students how they think the motion sensor should be programmed. What information will the software need to run the hardware correctly?

Write a pseudocode program for making the motion sensor work to show an image on the light matrix when the hub is tapped. Pseudocode is writing in words what you want the program to do.

Sample Pseudocode:

- Import motion_sensor
- If hub is tapped
- Show image

Note: The code you write out in steps does not need to match exactly what will be put in the program. This is to help student think through what is needed.

Provide students with the sample code to use the motion sensor to show different images depending on the gesture or action taken on the hub. Students need to type this program into the programming canvas.

Ask students to run the program.

```
from hub import light_matrix, motion_sensor

import runloop

def if_tapped():

    # If the hub is tapped, return the TAPPED gesture

    return motion_sensor.gesture() == motion_sensor.TAPPED

async def main():

    #Wait until the hub is tapped to show a sad face

    await runloop.until(if_tapped)

    light_matrix.show_image(IMAGE_SAD)

    print('Stop Please')

runloop.run(main())
```

Discuss the program with students. Ask students to point out things about this program that are new. Students might notice the use of "if" and double equal signs (==).

Explain to students that this program is a conditional statement. It is telling the hub to take one action if the condition is met. Discuss the condition – tapping the hub. The condition is set by the "if" statement. Only if the condition is met does the sad face appear on the hub. If the condition is not met, nothing appears on the hub.

Also point out the double equal signs to the students. A single equal sign cannot be used in our conditional statement because it is used to set the value of a variable or an object. Python uses a double equal sign to designate a comparison.

A conditional statement is comparing the input to whatever condition is set.

Allow students time to investigate setting a condition in their program by changing the different inputs. The options are `motion_sensor.TAPPED`, `motion_sensor.DOUBLE_TAPPED`, `motion_sensor.SHAKEN`, `motion_sensor.FALLING`, and `motion_sensor.UNKNOWN`. The gesture options can also be referenced in the Knowledge Base under API Modules, Hub, Motion Sensor.

Tilt for Action

Students will explore orienting the hub different ways for action.

In addition to gestures, the motion sensor can be used to set conditions for how you orient or tilt it. Provide students with the sample code to use the motion sensor to show different images depending on the orientation or tilt of the hub. Students need to type this program into the programming canvas.

Ask students to run the program.

Discuss the condition met in this example – orienting or tilting the hub up. Only if the condition is met does the arrow appear on the hub. If the condition is not met, nothing appears on the hub.

Challenge students to add to their program. If the condition is met, print a saying in the console and play a sound. Allow students time to write and run their program.

Note: If the hub is already in the correct orientation specified by the program, the arrow will automatically appear and the sound play.

Sample Code:

```
from hub import light_matrix, motion_sensor

from app import sound

import runloop

def face_up():

    # Return if the hub is oriented USB port up

    return motion_sensor.up_face() == motion_sensor.BACK

async def main():

    await runloop.until(face_up)
```



```
light_matrix.show_image(IMAGE_ARROW_N)
```

```
print('I am awake')
```

```
await sound.play('Applause 1')
```

```
runloop.run(main())
```

Allow students time to investigate setting a condition in their program by changing the different inputs. The orientation refers to when a particular side of the hub is facing up. The orientation options are:

The speaker side is `motion_sensor.FRONT`

The USB side is `motion_sensor.BACK`

The A, C, E port side is `motion_sensor.LEFT`

The B, D, F port side is `motion_sensor.RIGHT`

The light matrix side is `motion_sensor.TOP`

The battery side is `motion_sensor.BOTTOM`

The orientation options can also be referenced in the Knowledge Base Hub section under Motion Sensor.

Remind students to reference the Knowledge Base as needed to add additional lines of code. Also students should watch the console for error messages.

3. Explain

Allow students to share their programs and discuss how the programs worked.

Ask students questions like:

- What is a conditional statement in a program?
- What does the "if" do to set up the condition?
- Why are the lines after the "if" indented?
- How does the motion sensor work?
- What are the conditions can you set for the motion sensor?

4. Elaborate

Challenge students to create a game that includes using the motion sensor to interact with a story.

Ask students to create a short story that has a blank in it. The blank should be

filled by an action from the motion sensor that can then show an image and play a sound. Students can use the `print()` function to include the words of the story in the console.

A simple example written as a pseudocode program is:

- `Print('One day Jane was walking through the park. She looked up.')`
- Tilt motion sensor up.
- Ghost Image appears.
- Sound plays.
- `Print('Then she woke up and realized it was a dream')`.

Allow students time to create their story games. Students should write their story out in pseudocode first. Ask students to also include code comments in their program to explain the parts of the program.

Student groups should share their stories with other groups. Ask students to reflect on different ways the motion sensor can be used.

5. Evaluate

Teacher Observation:

Discuss the program with students.

Ask students questions like:

- How do conditional statements work?
- How does the motion sensor work?
- What are different ways you can use the motion sensor?

Self-Assessment:

Have students answer the following in their journals:

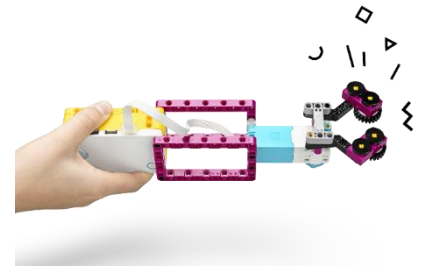
- What did you learn today about how to use conditional statements?
- What characteristics of a good teammate did I display today?
- Ask students to rate themselves on a scale of 1-3, on their time management today.
- Ask students to rate themselves on a scale of 1-3, on their materials (parts) management today.

Claw Machine

Grade 6-8

45 minutes

Beginner



Claw Machine

Students will create a robotic hand to move bricks using conditional statements.

Questions to investigate

- How can we create conditional statement without using an if statement?

Materials needed

- SPIKE Prime sets ready for student use. Prior to the first lesson, please visit the following website for help with set up, kit organization and SPIKE App <https://education.lego.com/en-us/start/spike-prime/intro>
- Devices with the SPIKE App installed.
- Student journals

Prepare

Ensure SPIKE Prime hubs are charged, especially if connecting through Bluetooth.

1. Engage

Engage students in thinking about how robotic arms move. Share several images and videos of different types of robotics arms. Examples may include welding, manufacturing, shipping, picking, sorting.

Ignite a discussion with students on how the robotic arms move and work. Have students point out how robotic arms are designed.

Shift the discussion to ask students if they have ever played the arcade game that uses a claw to grab a prize. Have students compare how this is similar and different to the robotic arms they studied. Discuss their ideas.

2. Explore

Students will explore working with conditional statements to program a robotic

KEY OBJECTIVES

Students will:

- Create a basic loop.
- Program a grabber model based on set conditions.

STANDARDS

CSTA

2-CS-02 Design projects that combine hardware and software components to collect and exchange data.

2-AP-10 Use flowcharts and/or pseudocode to address complex problems as algorithms

2-AP-13 Decompose problems and subproblems into parts to facilitate the design, implementation, and review of programs.

2-AP-16 Incorporate existing code, media, and libraries into original programs, and give attribution.

2-AP-17 Systematically test and refine programs using a range of test cases.

2-AP-19 Document programs in order to make them easier to follow, test, and debug.

hand to grab objects.

Direct students to the **BUILD** section in the SPIKE App. Here students can access the building instructions for **Robotic Hand**. Ask students to build the model. The building instructions are also available at <https://education.lego.com/en-us/support/spike-prime/building-instructions> listed as Pass the Brick.

Direct students to open a new project in the Python programming canvas. Ask students to erase any code that is already in the programming area. Students should connect their hub.

Use the Robotic Hand

Discuss ideas for making the Robotic Hand pick up objects like the claw machine. Think about ways to program the hand to open and close to pick up objects. Students should realize that they need to set a condition for what to do for the hand to be open or closed.

Share this sample program with students. Review the program as a group to ensure students understand how it will work. The program sets the condition to close the hand by pressing and holding the left button to close the hand. When released, the hand will open.

Students will need to type this program into the programming canvas. Students should run the program.

```
from hub import port, button

import runloop

import motor

def pressed():

    # When the left button is pressed, it returns to open

    return button.pressed(button.LEFT)

def released():

    # When the left button is released, it returns to close

    return not button.pressed(button.LEFT)

async def main():

    await runloop.until(pressed)
```

```
motor.run(port.F, -750)

await runloop.until(released)

motor.run(port.F, 750)

runloop.run(main())
```

After students see how the program will work, ask students to remove all the 2x4 bricks from their set. Practice using the Robotic Hand as if it is the claw machine to pick up the bricks and move them. Note, the program is set to only work one time.

3. Explain

Discuss with students how the program worked.

Ask students questions like:

- How does the program work?
- How is a conditional statement used in the program? What are the conditions?
- What are the true and false indicating in the program?

Point out to students that this is a conditional statement even though we do not use the if statement. Discuss why this is still considered a conditional statement. The program is using a condition to run the motor based on the button being pushed and released.

4. Elaborate

Challenge students to play a relay game.

Have two groups work together to pass a brick from one Robot Hand to another. Group 1 should grab the brick, then turn to Group 2 and try to pass the brick to them. Then Group 2 can try to pass the brick back. Challenge students to see how many times they can pass the brick back and forth.

Discuss this challenge with students. Ask them if they need to change the program in any way. Students should recognize that they will need to be able to have the hand open and close more than one time. Prompt students to think about including a while loop in their program.

Allow students time to modify their program and run it. Remind students to watch the console for error messages.

Sample Program:

```
from hub import port, button
import runloop
import motor

def pressed():
    # When the left button is pressed, it returns to open
    return button.pressed(button.LEFT)

def released():
    # When the left button is released, it returns to close
    return not button.pressed(button.LEFT)

async def main():
    while True:
        await runloop.until(pressed)
        motor.run(port.F, -750)
        await runloop.until(released)
        motor.run(port.F, 750)

runloop.run(main())
```

Allow students time to practice passing the bricks back and forth. Consider having the whole class try to pass one brick or a series of bricks through all groups as a final challenge.

Discuss the experience as a group. Ask students to define what a conditional statement is and different ways that you can create them. Students should recognize that a conditional statement is about setting condition in the program that can be true or false. In this example, the condition is set by "while True" indicating that the loop will continue as long as the condition is true. If the condition is not true, then the action will not happen.

5. Evaluate

Teacher Observation:

Discuss the program with students.

- Ask students questions like:
 - How did the Robotic Hand react to the conditional statement?
 - What are different ways to create conditional statements?
 - How can you create a program that allows you to manually indicate if the condition is met or not?

Self-Assessment:

Have students answer the following in their journals:

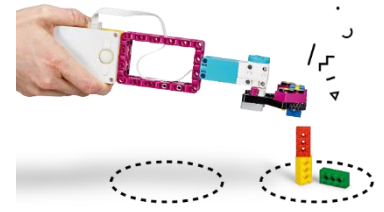
- What did you learn today about conditional statements and how to use them with loops?
- What characteristics of a good teammate did I display today?
- Ask students to rate themselves on a scale of 1-3, on their time management today.
- Ask students to rate themselves on a scale of 1-3, on their materials (parts) management today.

Charting Game Decisions

Grade 6-8

45-90 minutes

Intermediate



Charting Game Decisions

Students will learn to create flowcharts.

Questions to investigate

- How can flowcharts support designing a program?

Materials needed

- SPIKE Prime sets ready for student use. Prior to the first lesson, please visit the following website for help with set up, kit organization and SPIKE App <https://education.lego.com/en-us/start/spike-prime/intro>
- Devices with the SPIKE App installed.
- Student journals

Prepare

- Ensure SPIKE Prime hubs are charged, especially if connecting through Bluetooth.
- Ensure students have the Robotic Hand model built, which was used in the Claw Game lesson.

1. Engage

Ignite a discussion with students about a decision they make each day. Let students list several examples.

Prompt students with an example. One decision might be what to eat for breakfast each morning. Discuss with students what are the factors that help determine what you eat for breakfast. This might include how hungry they are or how much time they have.

Let students choose their own example and write out a list of questions they might ask themselves to help make a decision.

Here is an example:

- I need to eat breakfast before school.

KEY OBJECTIVES

Students will:

- Understand how to use flowcharts in planning.
- Create flowcharts and write programs that follow them.

STANDARDS

CSTA

2-CS-02 Design projects that combine hardware and software components to collect and exchange data.
2-AP-10 Use flowcharts and/or pseudocode to address complex problems as algorithms
2-AP-13 Decompose problems and subproblems into parts to facilitate the design, implementation, and review of programs.
2-AP-16 Incorporate existing code, media, and libraries into original programs, and give attribution.
2-AP-17 Systematically test and refine programs using a range of test cases.
2-AP-19 Document programs in order to make them easier to follow, test, and debug.

VOCABULARY

Flowchart

- Am I very hungry? Yes
- Do I have time to eat? Yes
- Do we have my favorite cereal? Yes
- I decide to eat a bowl of my favorite cereal.

Let several groups share their examples. Ask students to identify areas that might be a conditional statement. What conditions for deciding what to each exist for example? Ask students to also think about how the output or end would change if you had answered any questions differently.




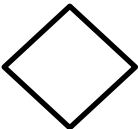
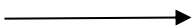
2. Explore

Students will explore creating flowcharts to document their program then create a stacking game program.

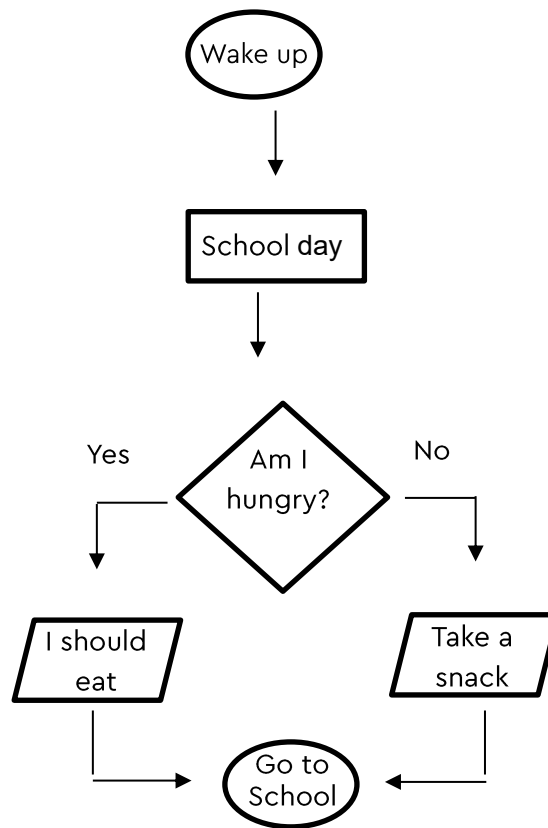
Introduce Flowcharts

Explain to students that the use of flowcharts to students as a tool to organize and plan their program. Flowcharts are used to organize and sequence an algorithm to help break down complex problems. Flowcharts are especially helpful when writing programs with conditional statements because it helps you map out the various paths the program can take if the conditions are met or not.

Introduce students to these common flowchart symbols, which they can use in their work. Consider providing a copy of the table to the students or posting somewhere in the classroom.

Symbol	Meaning
	Start/Finish Use to show the beginning or end of a program
	Input/Output Use to show input operation or output
	Processes Use to show a process
	Decision Use to show a decision to be made or condition to be met
	Flow Use to show the direction or flow of the program

Work together to create a flowchart based on the engage activity. Share this flowchart with students to discuss what possible outcomes could occur.



Discuss the flowchart together as a group. Notice there is more than one outcome. The question we are asking is setting up a condition. Either we are hungry, making the condition true, or not, which makes it false.

Allow students time to complete their own flowchart from the example in the engage, practicing using the symbols provided.

Discuss the flowcharts as a group.

Stacking Game

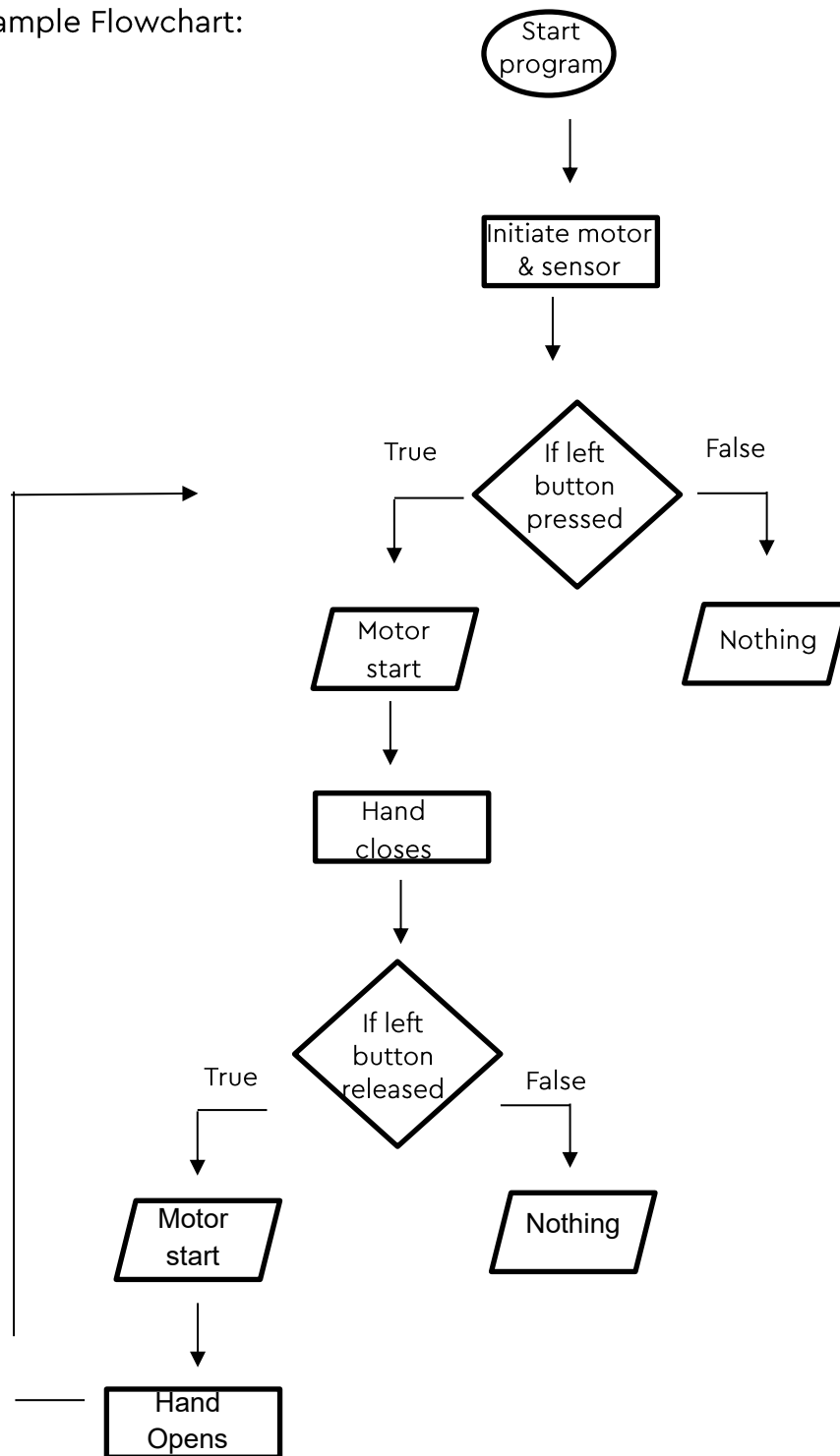
Direct students to open a new project in the python programming canvas. Students should open their program from the Claw Game lesson. Students should connect their hub.

Challenge students to use their Robotic Hand for even more precise movements. Students should try to stack as many of the 2x4 bricks as possible in 3 minutes.

Explain to students that they need to create a flowchart for completing this challenge prior to starting. Students should write down the steps needed to move from the start (all bricks separated) to the end (all bricks stacked).

Students can refer to their Claw Game program to help think about what needs to happen.

Example Flowchart:



Discuss the flowchart example with students. Point out where the conditional statement is and the fact that a while loop is included. Show the example flowchart if needed.

Have each student in the group take turns seeing how many bricks can be stacked on top of each other. Students can re-use the program from the Claw Game or create a new program.

Allow students time to complete the challenge.

3. Explain

Discuss the challenge with students.

Ask students questions like:

- How would the flowchart help you create a program?
- What is difficult about creating flowcharts?
- How were you able to stack the bricks?
- Which part of the program is the conditional statement?
- What was difficult about this challenge?

4. Elaborate

Challenge students to race.

Prompt students to time themselves playing the stacking game one more time. This time students stack and then unstack all their bricks. Allow groups of students to race against each other.

Allow students time to complete the challenge. Discuss the challenge together.

5. Evaluate

Teacher Observation:

Discuss the program with students.

Ask students questions like:

- What is the purpose of a flowchart?
- Why are the shapes for each section of the flowchart different?
- What happened when you tried to stack the bricks? What this an easy task?

Self-Assessment:

Have students answer the following in their journals:

- What did you learn today about using flowcharts?
- What characteristics of a good teammate did I display today?

- Ask students to rate themselves on a scale of 1-3, on their time management today.
- Ask students to rate themselves on a scale of 1-3, on their materials (parts) management today.

Guess Which Color

Grade 6-8

45 minutes

Beginner

Guess Which Color

Students will learn to program a color sensor to create a game.

Questions to investigate

- How can a color sensor be used to provide information to cause reactions?

Materials needed

- SPIKE Prime sets ready for student use. Prior to the first lesson, please visit the following website for help with set up, kit organization and SPIKE App <https://education.lego.com/en-us/start/spike-prime/intro>
- Devices with the SPIKE App installed.
- Student journals

Prepare

Ensure SPIKE Prime hubs are charged, especially if connecting through Bluetooth.

1. Engage

Engage students in thinking about how a color sensor works.

Using 4 different color bricks, challenge students to compete in a game. Tell students you will hold up a color brick and they should do the action assigned to that color. Consider writing the actions listed below somewhere that students can view.

Blue = Clap

Yellow = Jump

Green = Stomp

Red = Spin around

KEY OBJECTIVES

Students will:

- Program the color sensor using conditional code.
- Create a game.

STANDARDS

CSTA

2-CS-02 Design projects that combine hardware and software components to collect and exchange data.
2-AP-10 Use flowcharts and/or pseudocode to address complex problems as algorithms
2-AP-13 Decompose problems and subproblems into parts to facilitate the design, implementation, and review of programs.
2-AP-16 Incorporate existing code, media, and libraries into original programs, and give attribution.
2-AP-17 Systematically test and refine programs using a range of test cases.
2-AP-19 Document programs in order to make them easier to follow, test, and debug.

Any other color = do nothing

Ask students to stand up. Play several rounds by holding up different colors and letting the students react. Consider starting slow and then speed up to have the students moving very quickly in the last round.

Discuss this game as a class. Ask students what told them to do a certain action. Discuss how this is similar to a color sensor. The sensor takes in information (input) based on the color it "sees" and will then take the action assigned by the program (or do nothing if that color is not in the code).

2. Explore

Students will explore working with a color sensor and conditional statement.

Direct students to the BUILD section in the SPIKE App. Here students can access the building instruction for Kiki, the Dog. Students should build the model. You can also find the building instructions at <https://education.lego.com/en-us/support/spike-prime/building-instructions> listed as Help!

Direct students to open a new project in the Python programming canvas. Ask students to erase any code that is already in the programming area. Students should connect their hub.

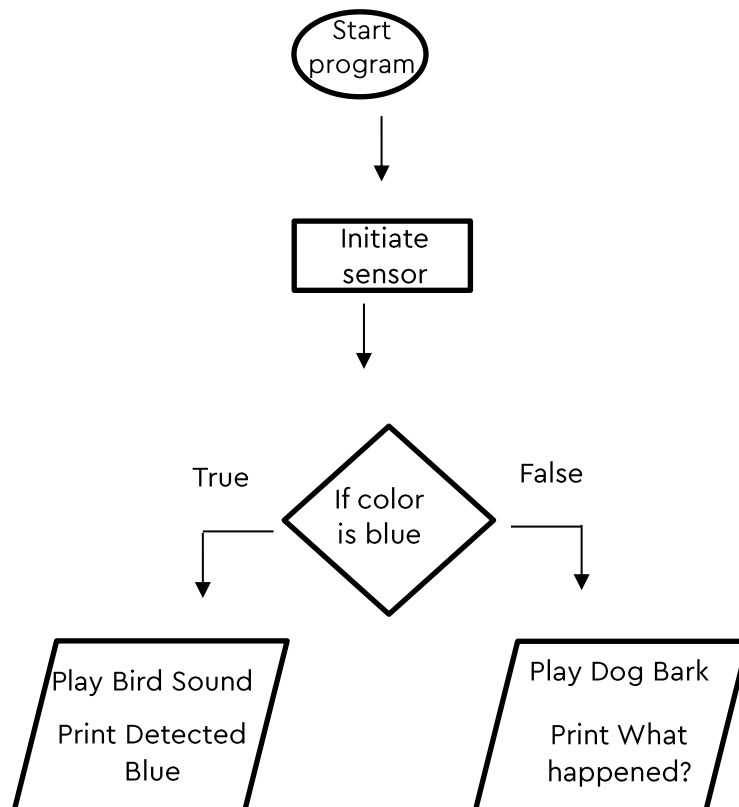
If/Else Statements

Students will investigate how to use the color sensor and create an if/else statement.

Introduce students to the Kiki model who has keen sight for different colors using his color sensor "eye". Prompt students to think about ways to use the color sensor to provide information. Ask students to locate the color sensor they built in the middle of the model. Discuss with students how to create a conditional statement that allows Kiki to play one sound if the blue color is detected and a different sound if the statement is false.

Create a flowchart for this program.

Sample Flowchart:



Provide students with this sample code to use the color sensor. Students will need to type this program into the programming canvas. Ask students to run the program.

```

from hub import port
import runloop
import color_sensor
import color
from app import sound
  
```

```

async def main():
    #if the color sensor senses blue, it will play bird sound
    if color_sensor.color(port.A) == color.BLUE:
        await sound.play('Bird')
        print('Blue Detected')
    #if the color sensor does not sense blue, it will play dog bark 1 sound
    else:
        await sound.play('Dog Bark 1')
        print('What happened?')
  
```

```
runloop.run(main())
```

Allow students time to investigate the program including changing the detected color. Discuss the program together as a group.

3. Explain

Discuss with students how the program worked.

Ask students questions like:

- How does this program work?
- How does the if/else statement work (using true/false language)?
- Why does the model not react to all three colors differently in the program?
- How does the else statement work?

4. Elaborate

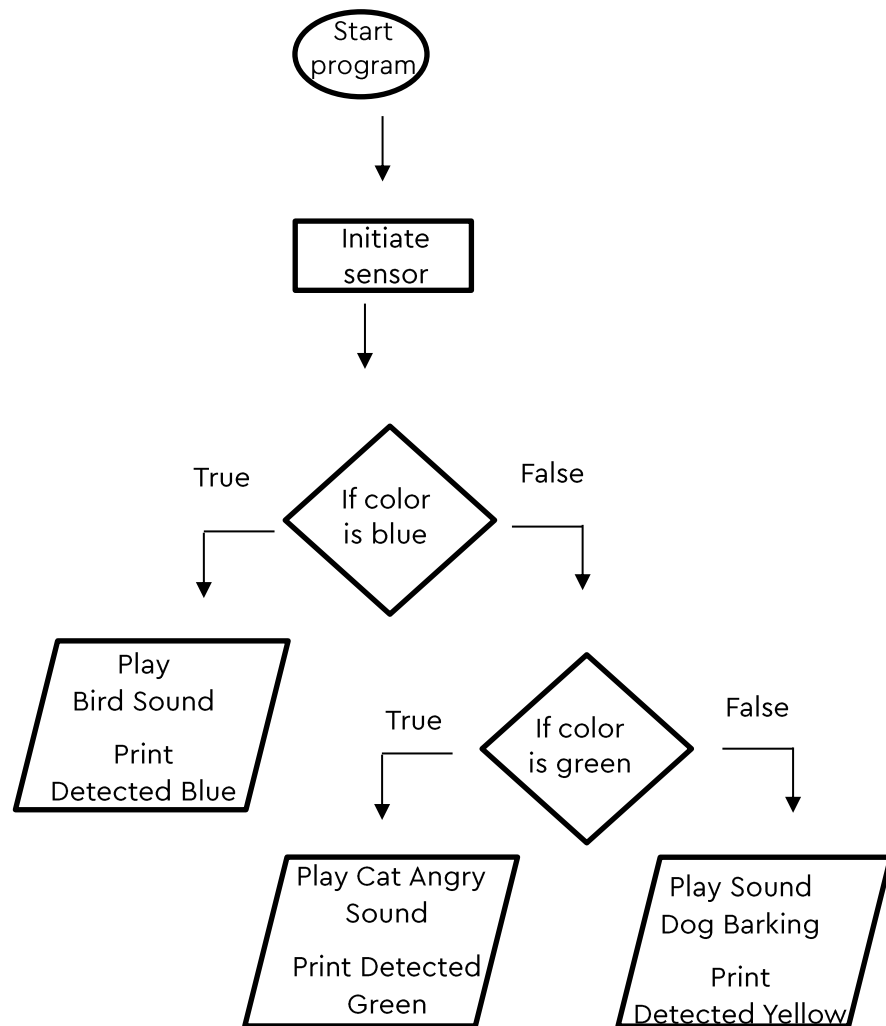
Students will investigate how to combine conditional statements into a single statement using the Elif code.

Prompt students to think about how to add to their program so that we can play different sounds for each color. Ask students to play one sound if blue is detected, a different sound if green is detected, and a third sound if neither of those are detected.

Introduce the elif statement to students. Elif is a combination of else and if and can only be included following an if statement. An elif statement allows the program to set a new condition if the first condition or if statement is read as false, providing a next step in the program.

Share the sample flowchart with students to discuss the path of a program that includes an elif statement.

Sample Flowchart:



Ask students to modify their program to include an elif statement. Students should recognize that the elif should be between the if of the first condition and be followed by the else of the condition set by the elif.

Sample Program:

```

from hub import port

import runloop

import color_sensor

import color

from app import sound

async def main():

    #if the color sensor senses blue, it will play bird sound

    if color_sensor.color(port.A) == color.BLUE:
  
```

```

await sound.play('Bird')

print('Blue Detected')

#if the color sensor senses green, it will play cat angry sound
elif color_sensor.color(port.A) == color.GREEN:

    await sound.play('Cat Angry')

    print('Green Detected')

#if the color sensor does not sense blue, it will play dog bark 1 sound
else:

    await sound.play('Dog Bark 1')

    print('What happened?')

runloop.run(main())

```

Allow students time to explore the program and change sounds. Ask students to share their programs.

5. Evaluate

Teacher Observation:

Discuss the program with students.

Ask students questions like:

- How did the if/elif/else program work?
- What happened when you added a loop to your conditional statement?
- Why would you want to include if/else and elif statements in programs?

Self-Assessment:

Have students answer the following in their journals:

- What did you learn today about using conditional statements and loops together?
- What characteristics of a good teammate did I display today?
- Ask students to rate themselves on a scale of 1-3, on their time management today.
- Ask students to rate themselves on a scale of 1-3, on their materials (parts) management today.

Guessing Game

Grade 6-8

45 minutes

Intermediate

Guessing Game

Students will investigate using conditional statements with loops.

Questions to investigate

- Why would a programmer combine an if/elif/else conditional statement with a loop?

Materials needed

- SPIKE Prime sets ready for student use. Prior to the first lesson, please visit the following website for help with set up, kit organization and SPIKE App <https://education.lego.com/en-us/start/spike-prime/intro>
- Devices with the SPIKE App installed
- Student journals

Prepare

- Ensure SPIKE Prime hubs are charged, especially if connecting through Bluetooth.
- Ensure students have built the Kiki, the Dog model, which was used in the Guess Which Color lesson.

1. Engage

Engage students in thinking about choosing from multiple options.

Prompt students to brainstorm how they make a choice when presented with several options. Ask students to imagine they are at a breakfast buffet with as many breakfast options as they can think of. How do they decide which things to eat? Ask students to select three items that they would choose to eat.

KEY OBJECTIVES

Students will:

- Write code that uses multiple condition statements using if/elif/else programming.
- Add a loop to code.
- Debug coding that has incorrect/missing syntax, missing code, or incorrect indentation.

STANDARDS

CSTA

2-CS-02 Design projects that combine hardware and software components to collect and exchange data.
2-AP-10 Use flowcharts and/or pseudocode to address complex problems as algorithms
2-AP-13 Decompose problems and subproblems into parts to facilitate the design, implementation, and review of programs.
2-AP-16 Incorporate existing code, media, and libraries into original programs, and give attribution.
2-AP-17 Systematically test and refine programs using a range of test cases.
2-AP-19 Document programs in order to make them easier to follow, test, and debug.

VOCABULARY

While loop

Let students share their choices. What influences help make their choices (mood, time, favorites, etc.)? Discuss how making a plate of food at the buffet is similar to creating a code that allows you to choose some items (add to the plate) while not choosing others (skip them in line).

2. Explore

Students will explore programming the Kiki model to react to colors according to conditions set in a loop.

Direct students to open their saved program for the Guess Which Color lesson. Students should connect their hub. Consider having students run the program one time to remember how it works.

Using a loop with an if/elif/else statement

Ask students to think about how they could modify the program used in the Guess Which Color lesson to repeat the condition allowing the program to identify more than one color. Students should recognize that a while loop will allow the color sensor to continue to react to the color read according to the conditions set.

Ask students to update their flowchart from the Guess Which Color lesson or create a new flowchart to show how the program should work.

Allow students time to modify their program and try it. Remind students to watch the console for error messages. Suggest to students to save this as a new program so as not to lose the original program.

Sample program:

```
from hub import port
import runloop
import color_sensor
import color
from app import sound

async def main():
    #if the color sensor senses blue, it will play bird sound
    if color_sensor.color(port.A) == color.BLUE:
        await sound.play('Bird')
        print('Blue Detected')
```

```

#if the color sensor senses green, it will play cat angry sound
elif color_sensor.color(port.A) == color.GREEN:
    await sound.play('Cat Angry')
    print('Green Detected')

#if the color sensor does not sense blue, it will play dog bark 1 sound
else:
    await sound.play('Dog Bark 1')
    print('What happened?')

runloop.run(main())

```

Review the program with students. Discuss what changed as a result of having a loop in the program. Students should see that the loop allows the program to continue running after the first color is detected, which means the program will continue to react when a new color is presented.

Play a Guessing Game

Have students remove the other two colors of 2x4 bricks from their set. Have students rearrange their model to allow all five colors to fit. Students should rework their program to include a new condition for reading each color. Prompt students that more than one elif statement can be included. Working together have students program each brick to make a different sound.

Sample Program:

```

from hub import port
import runloop
import color_sensor
import color
from app import sound

```

```

async def main():

    #if the color sensor senses blue, it will play bird sound
    if color_sensor.color(port.A) == color.BLUE:
        await sound.play('Bird')
        print('Blue Detected')

    #if the color sensor senses green, it will play cat angry sound
    elif color_sensor.color(port.A) == color.GREEN:
        await sound.play('Cat Angry')
        print('Green Detected')

    #if the color sensor senses red, it will play snoring sound
    elif color_sensor.color(port.A) == color.RED:
        await sound.play('Snoring')
        print('Red Detected')

    #if the color sensor senses magenta, it will play moo sound
    elif color_sensor.color(port.A) == color.MAGENTA:
        await sound.play('Moo')
        print('Magenta Detected')

    #if the color sensor senses yellow, it will play wobble sound
    elif color_sensor.color(port.A) == color.YELLOW:
        await sound.play('Wobble')
        print('Yellow Detected')

    #if the color sensor does not sense any of these colors, it will play dog bark 1 sound
    else:
        await sound.play('Dog Bark 1')
        print('What happened?')

```

```
runloop.run(main())
```

Each partner will take a turn being the game master and the other partner will play the guessing game. Without Partner 2 looking at the model, Partner 1 should play the program allowing the model to scan each color of the 2x4 bricks chosen. Partner 1 will get to decide which colors to scan. Partner 2 can listen to the sounds played in order to guess the color combination. Partner 2 can either write the color order down or use other pieces from the set to indicate the color order.

Have students check the guess at the end of the game and then switch partners.

3. Explain

Discuss with students how the program worked.

Ask students questions like:

- How can a loop and conditional statement work together in one program?
- How is this program using more than one conditional statement?

Remind students that the while loop actually sets up a conditional statement as well. While that section of code is true, it will continue to repeat.

- How many conditions can you set in one program?
- When will the program end?

4. Elaborate

Challenge students to try a new game where they are not detecting colors, but checking for bugs.

Show students each program and error message. Have students discuss what needs to change in each code to fix the bug or add the missing code. Consider making the changes as a class and ensuring the program runs correctly after each change.

Debug activity 1:

```
from hub import port
```

```
import runloop
```

```
import color_sensor
```

```
import color

async def main():
    #if the color sensor senses blue, it will play bird sound
    if color_sensor.color(port.A) == color.BLUE:
        await sound.play('Bird')
        print('Blue Detected')

    #if the color sensor senses green, it will play dog bark 1 sound
    elif color_sensor.color(port.A) == color.GREEN:
        await sound.play('Dog Bark 1')
        print('Green Detected')

    #if the color sensor does not sense blue, it will play cat angry sound
    else
        await sound.play('Cat Angry')
        print('What happened?')
```

Traceback (most recent call last):

File "Project 2", line 17

SyntaxError: invalid syntax

Students should recognize that the error is in line 17 from the error message. The invalid syntax means that we have not formatted something in the code correctly. Looking at line 17, students should notice that the else statement should have a colon at the end (:) with the statement of what to do as the else written on the next line with an indent.

Debug Activity 2

```
from hub import port
```



```

import runloop
import color_sensor
import color
from app import sound

async def main():
    #if the color sensor senses blue, it will play bird sound
    if color_sensor.color(port.A) == color.BLUE:
        await sound.play('Bird')
        print('Blue Detected')

    #if the color sensor senses green, it will play cat angry sound
    elif color_sensor.color(port.A) == color.GREEN:
        await sound.play('Cat Angry')
        print('Green Detected')

    #if the color sensor does not sense blue, it will play dog bark 1 sound
    else:
        await sound.play('Dog Bark 1')
        print('What happened?')

runloop.run(main())

```

Traceback (most recent call last):

File "Color 2", line 8

SyntaxError: invalid syntax

Students should recognize that the error is in line 8 as indicated in the error message. The conditional statement created by the if is not reading the condition because the lines of code for the condition are not indented. To fix the program, students need to indent all the lines of code needed for the conditional (lines 8 and 9).

Ask students to indent line 8 and only line 8. Run the program again. Students will receive a new error message.

SyntaxError: invalid syntax

Traceback (most recent call last):

File "Color 2", line 12

SyntaxError: invalid syntax

Students should recognize that there is still an error in the code. By indenting

line 8, we completed the conditional statement. The condition reads line 9 and then ends because line 9 is not indented.

Missing Code Activity

```
from hub import port

import runloop

import color_sensor

import color

async def main():

    #if the color sensor senses blue, it will play bird sound
    if color_sensor.color(port.A) == color.BLUE:
        await sound.play('Bird')
        print('Blue Detected')

    #if the color sensor senses green, it will play dog bark 1
    elif

    #if the color sensor does not sense blue, it will play cat angry
    else
```

Discuss this program with students. Ask students what is missing to make this a complete program. Similar to debugging, you can run the program and receive an error message.

Traceback (most recent call last):

File "Project 2", line 12

SyntaxError: invalid syntax

In this case, the program sends an error for line 12 because the elif statement is not defined. Students need to add something that sets the new condition if the

original condition is read as false.

Ask students to add a piece of code into the elif statement and try the program again. Students might have indicated a need to add to the else statement as well. If students run the new program, they will see a new error message to add a line of code to the else statement as well. Tell students that this error message did not show up before because the program will read the first error message and then stop. This means that additional messages are not initially called out.

5. Evaluate

Teacher Observation:

Discuss the program with students.

Ask students questions like:

- How can a loop and conditional statement work together in one program?
- What actions should you take when you receive an error message?
- How many elif statements can you include in a program?

Self-Assessment:

Have students answer the following in their journals:

- What did you learn today about adding multiple conditions in one program?
- What characteristics of a good teammate did I display today?
- Ask students to rate themselves on a scale of 1-3, on their time management today.
- Ask students to rate themselves on a scale of 1-3, on their materials (parts) management today.

Score!

Grade 6-8

45 minutes

Intermediate

Score!

Students will apply using conditional statements to keeping score in a game.

Questions to investigate

- How can programmers use conditions to keep track of actions or conditions?

Materials needed

- SPIKE Prime sets ready for student use. Prior to the first lesson, please visit the following website for help with set up, kit organization and SPIKE App <https://education.lego.com/en-us/start/spike-prime/intro>
- Devices with the SPIKE App installed.
- Student journals

Prepare

Ensure SPIKE Prime hubs are charged, especially if connecting through Bluetooth.

1. Engage

Watch the goal lesson engage video available at <https://education.lego.com/en-us/lessons/prime-extra-resources/goal#lesson-plan>. Ask students to study the way the arms are moving on the player. Students should identify that they move around an entire circle. Discuss with students how they can program the player to score as many goals as possible.

2. Explore

Students will build a Goal and Player model to investigate different ways to move using the motors.

Direct students to the **BUILD** section in the SPIKE App. Here students can access the building instruction for **Table Top Game**. Ask students build the Goal and

KEY OBJECTIVES

Students will:

- Program movement and light matrix.
- Apply knowledge of conditional statements.

STANDARDS

CSTA

2-CS-02 Design projects that combine hardware and software components to collect and exchange data.

2-AP-10 Use flowcharts and/or pseudocode to address complex problems as algorithms

2-AP-13 Decompose problems and subproblems into parts to facilitate the design, implementation, and review of programs.

2-AP-16 Incorporate existing code, media, and libraries into original programs, and give attribution.

2-AP-17 Systematically test and refine programs using a range of test cases.

2-AP-19 Document programs in order to make them easier to follow, test, and debug.

Accessories and Player models. You can also find the building instructions named Goal! 1 of 2 and Goal! 2 of 2 at <https://education.lego.com/en-us/support/spike-prime/building-instructions>.

Direct students to open a new project in the Python programming canvas. Ask students to erase any code that is already in the programming area. Students should connect their hub.

Score a Goal

Students will need to program their player to score a goal. Discuss with students the best way to program the motor to move – using seconds, degrees, or for position. Which would be the more efficient way to write your code? Encourage students to try more than one method before settling on their final idea.

Ask students to create a flowchart to document what the program needs to do. If students need help, they should utilize the Knowledge Base and their notes. Ask students to write and run their program.

Sample Code:

```
import runloop
```

```
import motor
```

```
from hub import port
```

```
async def main():
```

```
    # Run a motor on port A to 180 degrees at 720 degrees per second. This will put the arm up to be ready to swing.
```

```
    await motor.run_to_absolute_position(port.A, 180, 720)
```

```
    # Run a motor on port A for 360 degrees at 720 degrees per second.
```

```
    await motor.run_for_degrees(port.A, 360, 720)
```

```
runloop.run(main())
```

Allow students time to investigate and choose the best program. Remind them also to watch the console for any error messages they might receive.

Keeping Score

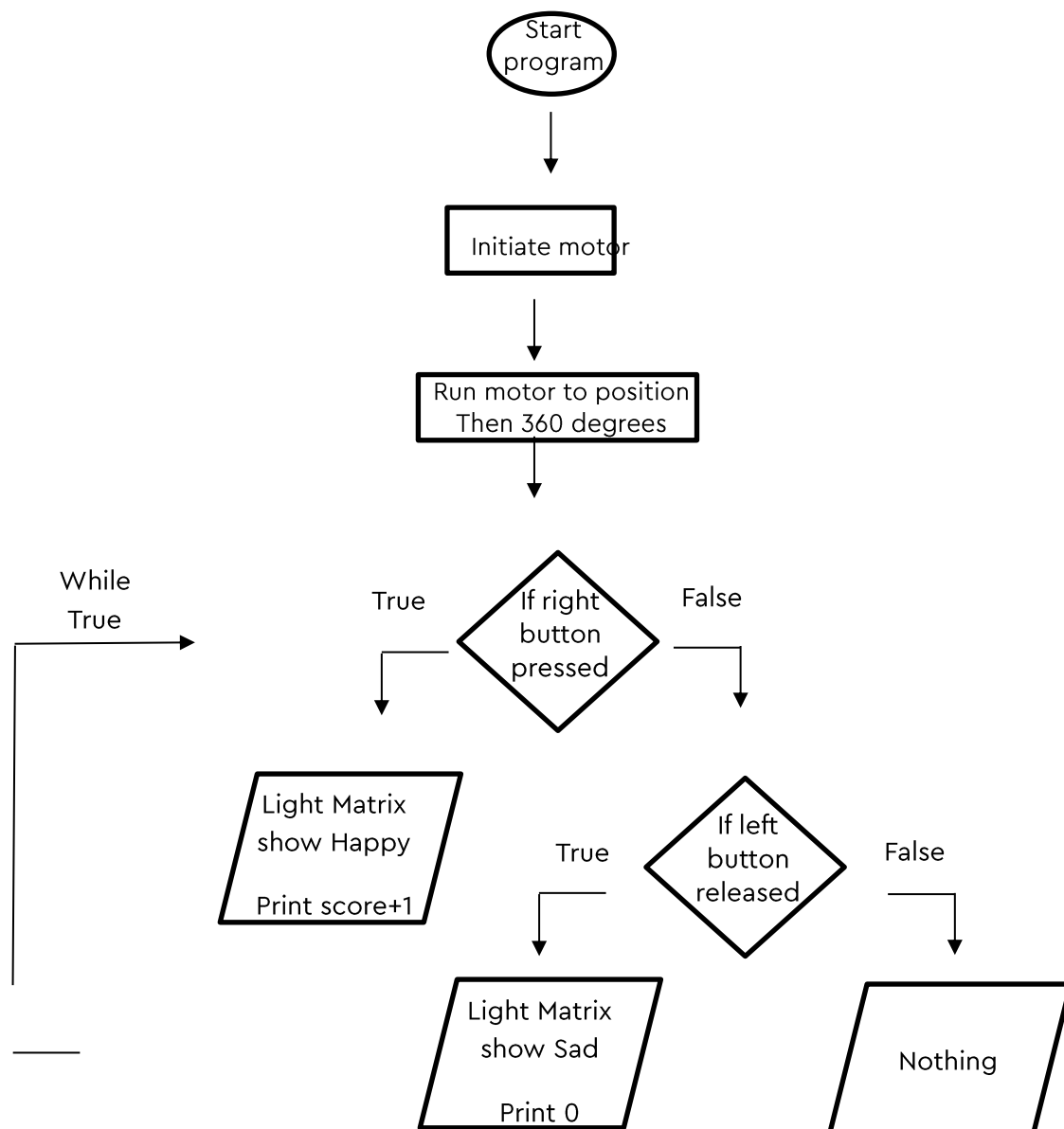
Students will want to show that they score each time their player makes a goal.

Challenge students to add to their previous code to include a conditional statement for keeping score. Students need to:

- Program the hub to indicate if a goal is scored or not by showing an image, writing a message, or playing a sound. Choose one if a goal is score and different image, word, or sound if a goal is not scored
- Program a way to show the score in the console, adding one if the goal is scored and showing 0 if one is not scored

Brainstorm with students the new lines of code needed through their flowchart. Ask students to update their flowchart to include the new steps in the program.

Flowchart example:



Discuss the flowchart together. Students should notice that we are indicating actions to happen when the conditional statement is both true and false.

Ask students to write and run their program.

Sample Program:

```
import runloop
import motor
from hub import port, light_matrix, button

def right_released():
    return not button.pressed(button.RIGHT)

def left_released():
    return not button.pressed(button.LEFT)

async def main():
    # Run a motor on port A to 180 degrees at 720 degrees per second. This will put the arm
    # up to be ready to swing.
    await motor.run_to_absolute_position(port.A, 180, 720)
    # Run a motor on port A for 360 degrees at 720 degrees per second.
    await motor.run_for_degrees(port.A, 360, 720)
    #set the score to 0.
    score = 0

    #Loop conditional statement to indicate score
    while True:
        if button.pressed(button.RIGHT):
            await runloop.until(right_released)
            light_matrix.show_image(light_matrix.IMAGE_HAPPY)
            score += 1
            print(score)
```

```
elif button.pressed(button.LEFT):  
    await runloop.until(left_released)  
    light_matrix.show_image(light_matrix.IMAGE_SAD)  
    print('0')
```

```
runloop.run(main())
```

Note: in this sample program, students need to set a variable for keeping score. The sample uses the name `score` for the variable and assigns a 0 value to it.

Allow students to run the program several times to try and score a goal. Review the code together as a group.

3. Explain

Students should explain the program they create to help the player score a goal, indicated a goal was scored, and how a conditional statement was used.

Ask students questions like:

- How did the program allow for you to indicate when you scored a goal?
- How was the conditional statement used?
- How does the flowchart guide you in creating a program?
- What was difficult about this challenge?
- How does the program decide whether to do the if action or the else action? Can both conditions be used?

Explain to students that the program determines if the conditional statement is true or not. The program will only run that portion of the program. So if it is true, the if will run and the else will be ignored.

4. Elaborate

Challenge the students to change their program to allow the player to keep score by adding up each scored goal rather than just indicating if one goal is scored.

All students time to brainstorm ideas for changing the program to add an additional point each time the player scores a goal. Prompt students to think about how the action will need to be repeated or continued (i.e. leading them to add a while loop).

Ask students to update their flowchart then modify and test their program.

Allow students time to share their program with the class and show how many goals they were able to score in total.

5. Evaluate

Teacher Observation:

Discuss the program with students.

Ask students questions like:

- How do if/else conditional statements work?
- How can you create a conditional statement that allows for two true responses?
- How can one input be substituted for another?

Self-Assessment:

Have students answer the following in their journals:

- What did I learn today about using if/else conditional statements?
- What characteristics of a good teammate did I display today?
- Ask students to rate themselves on a scale of 1-3, on their time management today.
- Ask students to rate themselves on a scale of 1-3, on their materials (parts) management today.

Game Time

Grade 6-8

90 minutes

Advanced

Game Time

Students will create and program their own tabletop game.

Questions to investigate

- How can conditions be set to create a simple table game?

Materials needed

- SPIKE Prime sets ready for student use. Prior to the first lesson, please visit the following website for help with set up, kit organization and SPIKE App <https://education.lego.com/en-us/start/spike-prime/intro>
- Devices with the SPIKE App installed.
- Student journals

Prepare

Ensure SPIKE Prime hubs are charged, especially if connecting through Bluetooth.

1. Engage

Engage students in a conversation about tabletop games.

If you completed the Score! Lesson, then discuss how this game worked as a tabletop game. Otherwise, consider showing images or videos to students of examples of tabletop games. Ask students to think about the positives of having a game like this. They should think about when they would use it and what would be fun.

Brainstorm

Brainstorm different ideas of tabletop games or games that could be made into a tabletop game.

KEY OBJECTIVES

Students will:

- Write code that includes conditions that must be met in a game format
- Create a game that requires a series of events requiring a robot to respond

STANDARDS

CSTA

2-CS-02 Design projects that combine hardware and software components to collect and exchange data.
2-AP-10 Use flowcharts and/or pseudocode to address complex problems as algorithms
2-AP-13 Decompose problems and subproblems into parts to facilitate the design, implementation, and review of programs.
2-AP-16 Incorporate existing code, media, and libraries into original programs, and give attribution.
2-AP-17 Systematically test and refine programs using a range of test cases.
2-AP-19 Document programs in order to make them easier to follow, test, and debug.
2-IC-22 Collaborate with many contributors through strategies such as crowdsourcing or surveys when creating a computational artifact.

As a group, brainstorm several ideas for creating a tabletop game. Consider what makes it a tabletop game and what would be fun to play.

Ask students to then work in their smaller groups to create a short survey to ask other students what they would like in a tabletop game. Students can share examples they are thinking about using and ask for other ideas. Students should include 3 questions about what would be fun, challenging, and ideas of themes. Students want to ensure the user interest in their game and also age-appropriateness of the idea.

Allow students time to create and complete their survey.

2. Explore

Students will design, build, and program a game to use as a tabletop game.

Design and Choose the Right Idea

Students should consider the input from their surveys and start to design their game.

Students will design, build, and program a game. The constraints for the game are:

1. It must use the light matrix
2. It must use at least one motor
3. It must use at least one sensor
4. It must include a timer feature

Students should create a sketch of their building idea and a flowchart of their programming idea.

Test and Iterate

Allow time for students to test and analyze their idea as they go, making improvements where needed. Students should test and evaluate their designs against the design criteria set and their flowcharts as they started making their solutions.

Ensure students use sketches and photos of their models to record their design journey during the creation stage of the lesson.

Allow students to receive feedback on their designs as time allows. This can be from other groups or the teacher.

3. Explain

Students should share their design and explain how it works. Conduct an initial sharing session with students.

Ask students questions like:

- How did you program your model to create a game? Ask students to share their program comments to explain.
- What decisions did you have to make while creating your design?
- What type of conditional statement did you choose?
- What were areas that you had to debug or troubleshoot?
- What was difficult about this challenge?

4. Elaborate

Allow students additional time to complete their program after the initial sharing -session.

Students should finalize their design and program. Encourage students to incorporate any new ideas they got from the sharing session.

Leave the models together if you are completing the lesson on feedback next.

5. Evaluate

Teacher Observation:

Discuss the program with students. Ask students questions like:

- What was difficult about this challenge?
- What was your approach to solving this challenge?
- What type of loops did you include and why?

Self-Assessment:

Have students answer the following in their journals:

- What did you learn today about creating my own design based on our ideas?
- What characteristics of a good teammate did I display today?
- Ask students to rate themselves on a scale of 1-3, on their time management today.
- Ask students to rate themselves on a scale of 1-3, on their materials (parts) management today.

Ideas to Help with Game Time

Grade 6-8

30-45 min.

Intermediate

Ideas to Help with the Game Time

Practice giving and using feedback from others.

Questions to investigate

- How can input from others help me make a better design and program?

Materials needed

- SPIKE Prime sets ready for student use. Prior to the first lesson, please visit the following website for help with set up, kit organization and SPIKE App <https://education.lego.com/en-us/start/spike-prime/intro>
- Devices with the SPIKE App installed
- Student journals
- Models from the Game Time lesson

Prepare

- Ensure SPIKE Prime hubs are charged, especially if connecting through Bluetooth.
- Ensure students have their built model from the Game Time lesson.

1. Engage

Review the model for providing feedback with students.

Explain to students the following guidelines for giving feedback.

Consider posting the guidelines for student reference.

- Feedback is not doing something for someone else.
- You should not rebuild a model for someone else.
- You should not type into someone's program.
- You should ask questions of each other.
- You should share your ideas and show your own programming, explaining why and how you did something.

KEY OBJECTIVES

Students will:

- Give specific feedback on a peer's project.
- Explore how to use feedback to improve a project.

STANDARDS

1B-IC-20 Seek diverse perspectives for the purpose of improving computational artifacts.

VOCABULARY

Feedback, Specific, Positive, Negative

- You should be encouraging and helpful to others and not provide negative or mean comments.

2. Explore

Have students work together to provide feedback to each other about the Game Time models.

Have two teams work together to provide feedback to each other. Teachers should model the process and what specific feedback looks and sounds like.

Review the procedure with students. Then have students take turns providing feedback.

- Team B will show their working model.
- Team A provides feedback while Team B takes notes in their journal.
- Then teams can switch roles. Team A will show their working model and take notes while Team B provides feedback.

Feedback should include:

1. Tell something they really like. This could be the model, program, or design.
2. Tell something that worked well.
3. Share something the group could try differently.
4. Share anything that is confusing, did not work or that could be improved,
 - Remind students to be kind and clear in explaining why it is not clear or could be improved.
 - Let the team receiving the feedback ask questions as needed for more clarity.
 - The team giving feedback can also share ideas for improvement.

Teacher tip – Model providing feedback for the class frequently to help them learn to use positive language instead of negative language when providing feedback. Also practice taking feedback and thinking about how to use it rather than becoming defensive.

3. Explain

Have students discuss what they learned from their feedback session.

Ask students questions like:

- What did you notice in models that worked well?
- What ideas did you get from others?
- What is something you can do with your feedback?

4. Elaborate

Students should incorporate the feedback they were given.

Give students time to modify their designs and program based on the feedback they received. Have students document their changes in their journal.

Allow students to share their updated models and programs. Ask students to share what changes they incorporated and how they were able to make the changes.

5. Evaluate

Teacher Observation:

Discuss the program with students.

Ask students questions like:

- How did you use the feedback given?
- How did it feel to give feedback to others? And to receive it?
- How did you work to provide good feedback today?

Self-Assessment:

Have students answer the following in their journals:

- What did you learn today about providing good feedback?
- What did you learn today about how feedback can help in your work?
- What characteristics of a good teammate did I display today?
- Ask students to rate themselves on a scale of 1-3, on their time management today.
- Ask students to rate themselves on a scale of 1-3, on their materials (parts) management today.

Career Connections – Lesson Extension

Students will extend learning to explore and research careers related to the topic explored

STEM, Computer Science

Grade 6-8

60-90 min.

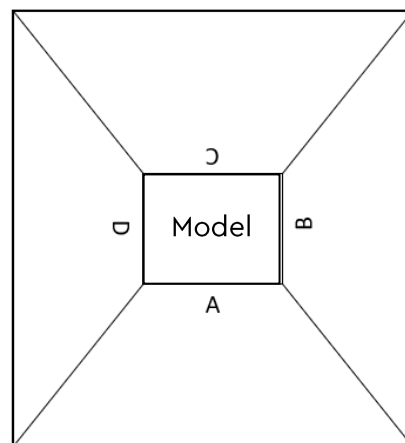
Beginner

Career Connections

Prepare

- LEGO® Education model from previous lesson.
- Have chart paper and markers available for student use. Prepare the charts ahead of time by drawing lines to separate the chart paper into four sections and label each section with a letter: A, B, C, D.

The chart would like this:



You may also want to include the name of the career area in the center.

- Prior to starting the lesson, make sure you have enough devices and access to the Internet for student use during this lesson.

1. Engage

Have students brainstorm possible jobs they see connected to the Game Time lesson. Students should consider all the possible jobs involved in creating an advertisement such as this. Prompt students to think about the model they

KEY OBJECTIVES

Students will:

- Articulate their personal interests and goals.
- Relate their personal interests and goals into possible career pathways.
- Explore various careers in career pathways.

STANDARDS

Career Ready Practice 10- Plan education and career path aligned to personal goals. (CCTC)

created, the programming, and deciding on the message to include.

2. Explore

Working in groups, students will explore one career cluster to identify jobs that relate to the Game Time lesson they completed. Students will use the model and programs created in that lesson to tie into the relevant jobs they find during their investigation.

Review the 16 career clusters if needed. Explain to students that they are going to look at specific jobs within the career clusters that relate to the concepts they have been investigating using the model they still have built.

- Consider assigning career cluster areas or letting students choose.
- Possible career areas to consider are Marketing, Business, IT, and STEM.

Ask students to consider where they can find the most reliable information on these career areas. Provide students will resources/websites as needed. Provide the students with a piece of chart paper with the 4 quadrants marked off. The information students should provide in each quadrant is

A. What career areas are relevant to the model/concepts you have been learning. Students can place the model in the center of the chart paper which will allow them to easily reference it in this section.

B. What skills are needed for this career area or job? Did you use any of these skills when working with the model/concept from the lesson? If so, which ones and how did you use them?

C. What are the types of jobs you might have in this career areas? What specialized training or certifications would you need to get this job?

D. What about this career area or job interests you and why?

3. Explain

Have each group share their findings with the whole class which will allow the class to learn about jobs in each career area. Ask students questions such as:

- How are the concepts we are learning now related to the jobs you investigated?
- What career cluster does this job belong in?
- What kind of qualifications are needed for this job?

- What kind of day-to-day responsibilities are associated with this job?
- What similar jobs are related to this (specific job)?
- Are you interested in this job?

4. Elaborate

Ask students to reflect about these career areas. How do you see these concepts overlapping across career clusters?

Ask students to reflect in a journal or other appropriate ways on all the career areas and jobs discussed to consider which might be most interesting to them and why.

5. Evaluate

Teacher Observation:

Discuss the program with students.

Ask students questions like:

- What are some careers involved with creating Game Time?
- What from the careers discussed interests you?
- What else would you be interested in knowing about these careers?

Self-Assessment:

Have students answer the following in their journals:

- What did I learn today that interested me about different careers?
- What characteristics of a good teammate did I display today?
- Ask students to rate themselves on a scale of 1-3, on their time management today.
- Ask students to rate themselves on a scale of 1-3, on their materials (parts) management today.

Career Exploration Activity

	<p style="text-align: center;">C</p> <p>What are the types of jobs you might have in this career areas? What training or specialized certifications would you need to get this job?</p>	
<p style="text-align: center;">D</p> <p>What about this career area or job interests you and why?</p>	<p>LEGO Education Model</p> <p>Career Cluster</p>	<p style="text-align: center;">B</p> <p>What skills are needed for this career area or job? Did you use any of these skills when working with the model/concept from the lesson? If so, which ones and how did you use them?</p>
	<p style="text-align: center;">A</p> <p>What career areas are relevant to the model/concepts you have been learning. Students can place the model in the center of the chart paper which will allow them to easily reference it in this section.</p>	