Introduction to Python Programming

Using LEGO® Education SPIKE™ Prime Set

Introduction to Python Programming Course 2

Using LEGO® Education SPIKE™ Prime Set

OVERVIEW

In this course, students will extend 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 use functions, compound conditionals, data and math functions, and lists (arrays) to control the flow of your programs. 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 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

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. Course 2 continues the coursework from Course 1 and contains the following units:

- Unit 6: Troubleshooting & Debugging
- Unit 7: Functions
- Unit 8: Compound Conditionals and Logic Operators
- Unit 9: Data & Math Functions
- Unit 10: Lists
- Project Unit: Environmental Impact or Transportation Project

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 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.

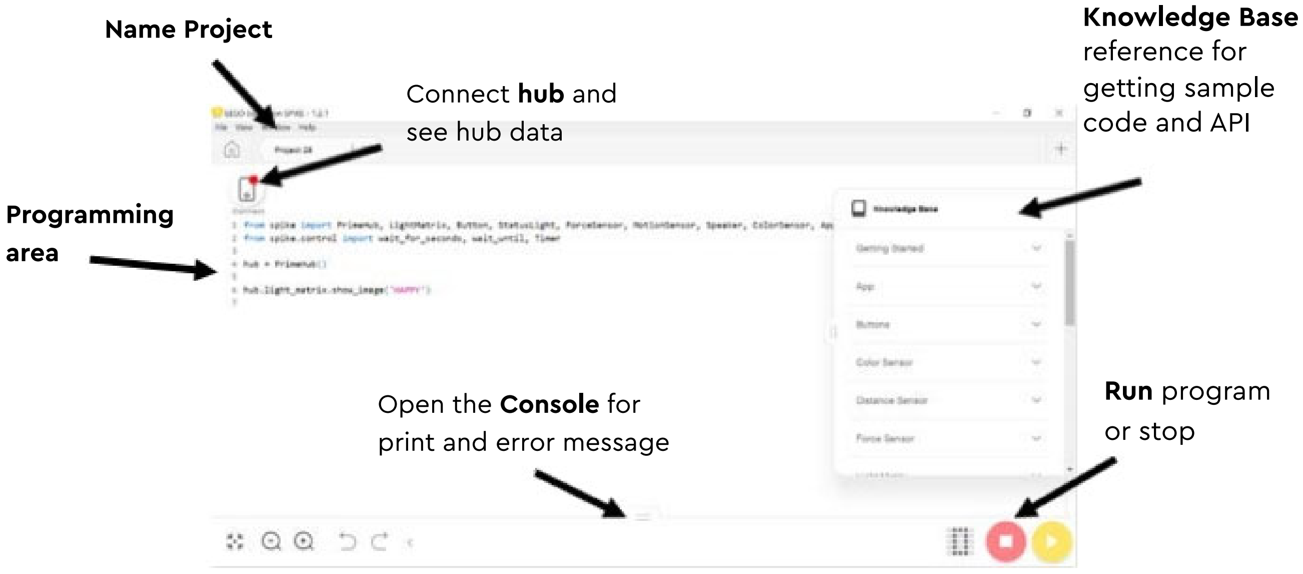
The following sections are reference for teachers who may not have taught Introduction to Python Programming Course 1 recently.

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 key features as shown in the image below.



Name Project
Connect **hub** and see hub data
Knowledge Base reference for getting sample code and API
Programming area
Open the **Console** for print and error message
Run program or stop
Project 28
Knowledge Base
Getting Started
App
Buttons
Color Sensor
Distance Sensor
Force Sensor

```
from spike import PrimeHub, LightMatrix, Button, StatusLight, ForceSensor, MotionSensor, Speaker, ColorSensor, Ap
from spike.control import wait_for_seconds, wait_until, Timer

hub = PrimeHub()

hub.light_matrix.show_image('HAPPY')
```

- 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 to provide students adequate feedback to continue grow their knowledge and skills. The three types of assessment include:

- Teacher observations which encourage teachers to discuss outcomes with students through posing questions and listening to how students 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	• Learn why a Pythonprogram must havelibraries 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	• 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	• 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	• 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	• 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	• 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	• 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	• 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	• 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	• 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	• 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	• 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	• 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	• 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	• Explore the force senor • 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	• 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	• 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
		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.
Grasshopper Troubles 45 minutes	• Make appropriate hardware decisions • Re-design a model to add a sensor	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.
Ideas to Help Your Grasshopper 30-45 minutes	• 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 4 Loops and Variables		
Lesson	Objectives	CSTA Standards
Warm Up Loop with Leo 45 minutes	• 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	• 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	• 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	• 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	• 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	• 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	• 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	• 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	• 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	• 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	• 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
	• 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	• 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 indention.	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	• 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	• 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	• 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	• 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	• 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	• 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	• 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	• 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	• 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	• 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	• 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	• 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	• 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.

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	• 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	• 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	• 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	• 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	• 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	• 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	• 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	• 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	• 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	• 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	• 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	• 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	• 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	• 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	• 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	• 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	• 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	• 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.

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	• 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	• 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	• 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.

Testing for Bugs: Troubleshooting Hardware and Software

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 identifying and fixing bugs. Students will investigate how to identify problems when designing, building, and programming robots and develop strategies for troubleshooting these issues. Students will identify the importance of testing and how to recognize if their bugs are in the hardware (design) or software (program). The lessons are designed in an order that allows students to progress in their skills and knowledge in the following areas:

- Create and debug programs
- Problem solve if an issue is in the design or the program and fix the problem
- Investigate troubleshooting design issues
- Utilize code comment features to documents parts of a program
- Define and decompose a problem

Unit Learning Promise

In this unit, students will explore designing models and programs and the troubleshooting needs that can arise from each. Students will experience how problems can happen in hardware and programming and develop skills for identifying where the issues exist and how to troubleshoot them. Students will utilize pseudocode to support creating algorithms and code comments to document their programs, supporting good debugging habits.

Investigation Questions:
How do software engineers identify and fix bugs in a program? How do engineers identify and repair hardware and software problems within a design?

Unit Lessons

Lesson 1	Lesson 2	Lesson 3	Lesson 4	Lesson 5	Lesson 6
Testing Prototypes Time: 45 min	**Break Dancer Break Down** Time: 45 min	**Dance to the Beat?** Time: 45 min	**Testing for Trouble** Time: 45 min	**Debug-inator** Time: 45 min	**Ideas to Help with the Debug-inator** 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-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.
1B-IC-20 Seek diverse perspectives for the purpose of improving computational artifacts.

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.		
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)		
6th Grade	**7th Grade**	**8th 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'	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'	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	ideas and expressing their own clearly	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	RST.6-8.3 Follow precisely a multistep procedure when carrying out experiments, taking measurements, or performing technical tasks.
L.6.6 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	L.7.6 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	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

Testing Prototypes

Grade 6-8 | 45 minutes | Beginner

Testing Prototypes

Students will explore the engineering design process.

Questions to investigate

- How do engineers take an idea and make it into a product?

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
- Unopened water bottles

Prepare

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

1. Engage

Engage students in thinking about the design engineering process.

Engineers use a process called the design engineering process when they are trying to find a solution to a problem. Introduce students to the design engineering process. You may use your own preferred model or the one provided below:

- Define a problem
- Research the problem
- Brainstorm possible solutions
- Select the most promising solution
- Construct a prototype

KEY OBJECTIVES

Students will:

- Brainstorm ideas and develop solutions to a problem.
- Program a model.

STANDARDS

CSTA

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.

VOCABULARY

Engineering Design Process
Constraint
Prototype

- Test and evaluate the prototype
- Communicate
- Redesign

Ignite a discussion with students on how engineers use this process to design solutions to problems. Consider sharing examples.

2. Explore

Students will use the design engineering process to build a bridge.

Present students with a problem to solve. Explain to students that they have been hired to construct a model of a bridge. The roadbed of the model, which is the large yellow technic plate, needs to raise above the table by 3 inches.

Ask students to record the steps of the design engineering process in their notebooks as they complete them. Together, complete step 1 by defining the problem. Students should identify that they need to build a bridge that has the roadbed 3 inches off the table. Identify other needs – like what should a bridge do – carry weight, not move side to side, and so forth.

Allow students time to complete step 2, to research the problem, as needed.

Students should record all ideas as they move to the next step, brainstorming possible solutions. Provide students with these constraints for the challenge:

- Students may only use the following elements from their set
- Large yellow technic plates
- Technic beams
- Connectors or axles
- Students may **not** use frames.
- The roadbed must hold 2 unopened water bottles.

Have students select their best option and build their prototype.

Ask students to verify that their original model does not move side-to-side and can hold some weight. Start with 2 unopened plastic water bottles.

Have students take a picture of their first bridge prototype.

3. Explain

Have students share their model and explain how they arrived at their final design.

Ask students questions like:

- What type of bridge could you build given the materials you can use?
- Why is planning prior to building important?

- What happened as you built? Did you have to change anything from the design?
- Why were you asked to verify that the model would hold the weight of the water bottles?
- What step were you completing when you added the water bottles?

Discuss with students that they were completing the test and iterate phase of their design when they were adding the water bottles. Students may indicate that their bridge did not hold the weight of the water bottles.

Discuss what is learned when this happens and what students can do next. Students should recognize that testing determines if a design works as intended or not. It allows the designer to investigate what needs to change or be fixed.

Prompt students to think about how to make the design better by iterating or making small changes based on the testing and then retesting to identify additional iterations needed.

Ask students questions like:

- Why is testing the model important?
- What can we learn from testing our model?
- What is a next step to take after testing the model and learning from that test?
- When do you know that the testing and iteration phase is complete?
- Did you see ideas from other teams that you might like to incorporate in your own design?

4. Elaborate

Allow for further iteration of the design based on adding new materials.

Explain to students that a new material design has been fabricated and is available for use – the frame and any other pieces in the set. Tell students that they can use frames, but do not have to do so. Additionally, the customer has asked that the prototype hold 10 pounds.

Tell students to experiment with adding new technology to their design. Prompt students to consider if they have to start the entire process over or just begin in the middle. Discuss the process to follow as a group.

Allow students time to redesign their bridge with the new pieces. Remind students to test and iterate on their design to ensure it meets the expectation of holding 10 pounds.

5. Evaluate

Teacher Observation:

Discuss the program with students. Ask students questions like:

- How well did your first prototype work?
- Why was testing your design so important?
- How does an engineer move from idea to prototype?

Self-Assessment:

Have students answer the following in their journals:

- What did you learn today about design engineering and the importance of testing?
- 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.

Break Dancer Break Down

Grade 6-8 | 45 minutes | Intermediate

KEY OBJECTIVES
Students will:

- Identify a problem and debug the program.

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-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
Bug
Debugging
Syntax Error
Logic Error
Runtime Error

Break Dancer Break Down

Students will investigate strategies for debugging programs.

Questions to Investigate

How do software engineers identify and fix bugs in a program?

Materials needed

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

Prepare

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

1. Engage

Discuss with students how to identify if a problem is with the hardware (model) or the program.

In the Testing Prototypes lesson, students identified ways to test and iterate on their model in order to find a final solution. However, there was no programming element involved in their bridge building. Now students will need to find ways to identify if problems are in the model design or the program.

Watch the video of the break dancer model dancing to get an idea of how the model should move.
https://education.lego.com/en-us/lessons/prime-life-hacks/break-dance#ignite-a-discussion

Discuss with students the different ways the dancer can move. Ask students questions like:

- Which parts of the model were moving?
- How can you replicate that movement with the model?
- If something goes wrong, how will you know if it is the model or the program?

2. Explore

Students will build a break dancer model to investigate identifying bugs and fixing them.

Direct students to the **BUILD** section in the SPIKE App. Here students can access the building instructions for the **Break Dancer** 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.

Test the Model
Students will identify that there are bugs in the program when trying the sample code.

Ask students to examine their model closely to identify how it should move. Students should try moving the motors and other parts of the model to see if it will move like the video based on the discussion in the engage section.

Allow students time to investigate how the model moves without creating a program. Ask students questions like:

- Did the model move in similar ways to the video?
- Does anything on the model seem to not be working properly?

Test the Program
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():

    # Move legs motor to 0 degrees
    await motor.run_to_absolute_position(port.F, 0, 600)

    # Move arms motor to 0 degrees

    await motor.run_to_absolute_position(port.D, 0, 400)

    # Repeat arm and leg movement 10 times

    for x in range(10)

        await motor.run(port.F, 600)

        await motor.run(port.D, 400)

runloop.run(main())
```

Allow students to run the program. Notice an error message is received.

Discuss with students that the error message shows there is a **bug** in the program. Students will need to check the lines of code carefully to find and fix the bug. This process is known as **debugging** a program.

Students should identify that the first way to identify if something is wrong with a program that an error message will be received. Look at the error message together:

```
Traceback (most recent call last):

File "Break Dancer", line 13

SyntaxError: invalid syntax
```

The error points students to the line to check for the message. Allow students time to try to troubleshoot this error. Note: a colon**:** is needed at the end of the line to read the code associated with the for loop.

3. Explain

Discuss the program with students and why it does not seem to be working correctly.
Ask students questions like:

- What does it mean to debug a program?
- How is the error message helpful?
- Why was it important to test the model before running the program?
- What does "syntax error" mean?

Discuss the types of errors with students:

- **Syntax errors** are grammar errors in our code. The code is written in a way that the machine can understand it. Typos and misuse of punctuation or parenthesis are good examples of reasons you will get a syntax error.
- **Logic errors** are found when running a program and are the result of not receiving the expected output. The code will run, but not produce the expected result or run as expected.
- **Runtime errors** occur while the program is running. The program often will run fine for some time until it encounters the problem and then end in an error.

4. Elaborate

Allow students additional time to explore the types of errors.

Explain to students that normally they would try hard not to have bugs in their program. However, the next challenge will be for them to try to find different bugs that can occur. Challenge students to change their program from the explore phase to see what types of errors they might find. See if they can replicate all three types.

Example Errors:

Example 1

Change the line await motor.run_to_absolute_position(port.F, 0, 600)

To read await motor.run_to_absolute_position(port.A, 0, 600)

Students will receive a runtime error message. The program does not notice there is a problem until it starts running. Then it realizes that there is not a motor plugged into port A.

```
Traceback (most recent call last):
File "Break Dancer", line 16, in <module>
File "Break Dancer", line 8, in main
OSError: [Errno 19] ENODEV
```

Example 2:

Change the line for x in range(10):

To read for x in range():

Students will receive a Type Error message, which is a type of syntax error, that indicates there is a missing function. The for loop requires a number to indicate how many times it should loop.

```
Traceback (most recent call last):
File "Break Dancer", line 16, in <module>
File "Break Dancer", line 12, in main
TypeError: function missing 1 required positional arguments
```

Note: There are ways to change the numbers to get an unexpected result in this program. However, the console will not show it as a logic error because the program has not specified an intended outcome against which to measure it.

5. Evaluate

Discuss the program with students. Ask students questions like:

- How does testing help identify bugs in your program?
- How can you tell the need to troubleshoot hardware and not the program?
- What are the types of errors you can receive? How can you use them to fix your program?

Self-Assessment

Have students answer the following in their journals:

- What did you learn today about identifying and fixing bugs?
- 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 to the Beat?

Grade 6-8 | 45 minutes | Intermediate

Dance to the Beat?

Students will investigate strategies for debugging programs.

Questions to Investigate

How do software engineers identify problems within 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 Break Dancer model built, which was used in the Break Dancer Break Down lesson.

1. Engage

Create a plan for testing programs to identify and fix bugs.

Spark a discussion with students on how to fix problems. Ask students to think of a time when they had an issue or problem or a time that they had an experience that did not turn out as expected.

Discuss what could have helped change the situation. Ask questions like:

- What steps could you have taken upfront to have a better outcome?
- What could you have done when you realized there was a problem to help change the outcome?

KEY OBJECTIVES

Students will:

- Identify a problem and debug the program.

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-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

Bug
Debugging
Syntax Error
Logic Error
Runtime Error

Discuss students' ideas for troubleshooting their experience and list ideas for how to troubleshoot. Share out the steps for identifying bugs or post in the classroom for all students to reference.

Steps to help identify bugs:

- Plan the program by creating a pseudocode or flowchart
- Document the program by using the # for code comments within the program
- Test your program with all types of data that are relevant or expected to be used
- Test your program with types of data that are not expected to be used or outside the range expected

2. Explore

Students will use a break dancer model to investigate, identifying bugs and fixing them.

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. If students completed the Break Dancer Break Down lesson, they could start with that program and modify it. Students should connect their hub.

Adding to a Working Program
Students will identify strategies to find bugs when adding to a program.

Ask students to run their existing program from the Break Dancer Break Down lesson or provide students with that program. Running the existing program will allow students to confirm there are no current errors in the program.

Ask students to modify their program so the dancer only moves when the color sensor senses a color according to the following assignment. You can use this chart or work together to create your own.

Blue	Move legs only
Green	Move arms only
Yellow	Move both arms and legs slowly
Red	Move both arms and legs quickly

Allow students time to create their new program. Students should test their program to make sure it works. Remind students to utilize the process discussed in the engage section to test the program for bugs.

Ask students to record all debugging activities in their journal. If an error message is received in the console, students can copy this. Students should copy any errors discovered while testing in their journal also.

```
import runloop

import motor

import color_sensor

import color

from hub import port, light_matrix


async def main():

    color_value = color_sensor.color(port.B)


    while True:

        await runloop.until(lambda: color_sensor.color(port.B) not in (-1, color_value))


        color_value = color_sensor.color(port.B)


        #if senses blue the arms will move

        if color_value == color.BLUE:

            light_matrix.show_image(light_matrix.IMAGE_HAPPY)

            await motor.run_for_degrees(port.D, 360, 400)


        #if senses green the legs will move

        elif color_value == color.GREEN:

            light_matrix.show_image(3)

            await motor.run_for_degrees(port.F, 360, 600)


        #if senses yellow both arms and legs will move together slowly

        elif color_value == color.YELLOW:

            light_matrix.show_image(light_matrix.IMAGE_HAPPY)

            motor.run_for_degrees(port.D, 360, 200)
            motor.run_for_degrees(port.F, 360, 400)

        #if senses red both arms and legs will move together quickly

        elif color_value == color.RED:

            light_matrix.show_image(light_matrix.IMAGE_HAPPY)

            motor.run_for_degrees(port.D, 360, 600)

            motor.run_for_degrees(port.F, 360, 800)

        #if senses any other color or no color

        else:

            light_matrix.show_image(light_matrix.IMAGE_SAD)

runloop.run(main())
```

3. Explain

Discuss the program with students and where they found bugs.
Ask students questions like:

- What bugs did you find in your program?
- Did you receive any error messages in the console? If so, how was the error message helpful in locating and debugging the program?
- How did you test the program for the expected values? And the unexpected ones?
- Could any of the problems been with the model? Why or why not?

4. Elaborate

Challenge students to create a program that has their dancer moving to a beat using their color sensor to control movements.

Ask students to utilize their program to create a new dance that has their dancer model moving to a beat. Students can play a song or just hum a beat. The dancer should move fast and slow to the beat which can be controlled when the students hold each color brick up to the color sensor.
Allow students time to investigate the program to see if any changes are needed. Then students should create their movements to match the beat.

Ask students to share their final movements with the class. Consider having a dance party for all the models at the same time.

5. Evaluate

Discuss the program with students. Ask students questions like:

- How does testing help identify bugs in your program?
- Why test your program for expected values and unexpected ones?
- What are the types of errors you can receive? How can you use them to fix your program?

Self-Assessment

Have students answer the following in their journals:

- What did you learn today about identifying and fixing bugs?
- 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.

LEGO® Education SPIKE Prime

Testing for Trouble

Grade 6-8 | 90 minutes | Intermediate

Testing for Trouble

Students will investigate strategies for troubleshooting hardware issues.

Questions to Investigate

How do engineers identify and repair hardware and software problems within a design?

Materials needed

- SPIKE Prime Set
- Device with SPIKE App installed
- Student journal
- Pen or marker (one per team)
- Printed worksheet for Broken Lesson (one per team; link to document)

Prepare

- Check to make sure SPIKE Prime hubs are charged, especially if connecting through Bluetooth.
- Reference for troubleshooting the model is available at https://education.lego.com/en-us/lessons/prime-invention-squad/broken#Planitem2

KEY OBJECTIVES

Students will:

- Identify and repair a hardware problem in a design.

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-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

Debugging
Pseudocode
CNC

1. Engage

Discuss with students what type of machines can make an exact cut repeatedly, so all pieces are the same. A CNC [computer numerical control] machine is programmed to make the same repetitive cuts. For example, a laser cutter cuts plastic, wood, or cardboard into exact shapes.

- Allow students to research what CNC machines are and how they are used.
- Ask students to name objects that they believe are made with a CNC machine.

Watch the video to get an idea of what the CNC machine they will build should do. https://education.lego.com/en-us/lessons/prime-invention-squad/broken#building-tips

Discuss with students what is wrong with the machine in this video. Ask students questions like:

- How do you know when something is not working right?
- What is your first reaction when something breaks?
- How can we figure out where the problem lies?

2. Explore

Students will build a CNC machine to investigate troubleshooting design issues or hardware issues.

Direct students to the **BUILD** section in the SPIKE App. Here students can access the building instructions for the **CNC Machine** 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 listed as Broken.

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.

Test the Model

Students will identify that there are hardware issues when trying the sample code.

Review this sample code to run the CNC machine together as a group. Discuss the code as a group and identify any potential errors in the code. Students should note that there don't appear to be any errors in the program.

Ask students to type this program into the programming canvas. Ask students to run the program to test it for errors.

```
from hub import button, port
import runloop
import motor

def right_button_pressed():
    return button.pressed(button.RIGHT)

def left_button_pressed():
    return button.pressed(button.LEFT)

async def main():
    while True:
        #feed the paper
        await runloop.until(right_button_pressed)
        await motor.run_for_degrees(port.C, 1000, 500)

        #cut a square and a rectangle
        await runloop.until(left_button_pressed)
        await motor.run_for_time(port.A, 1500, -500)

        # These 4 blocks should 'cut' a square.
        await motor.run_for_degrees(port.A, 400 , 500)
        await motor.run_for_degrees(port.C, 575, 500)
        await motor.run_for_degrees(port.A, -400 , 500)
        await motor.run_for_degrees(port.C, -575, 500)

        # These 4 blocks should 'cut' a rectangle.
        await motor.run_for_degrees(port.A, -60, 500)
        await motor.run_for_degrees(port.C, -800, 500)
        await motor.run_for_degrees(port.A, 400, 500)
        await motor.run_for_degrees(port.C, 1600, 500)
        await motor.run_for_degrees(port.A, -400, 500)
        await motor.run_for_degrees(port.C, -800, 500)

runloop.run(main())
```

Allow students time to run the program. Notice the machine is not working properly. The machine is supposed to help you "cut" (draw) parts.

Discuss together as a group if the problem is with the hardware/model or the program. Ask students questions like:

- What is not working?
- How does the model move? Does it seem to be moving correctly?
- Did you receive any errors in the console or see any indication that the program was not working?

Students should recognize that there is an issue with the model not the program.

Identify and troubleshoot design problems.

Work together to identify the problems. Look at the machines and see how it behaves versus what you think it will do. Have students write the issues found in their journals or on chart paper. Encourage students to use these steps:

- Identify the problem.
- Take a close look.
- Observe how the model behaves verses what you think it should do.
- Brainstorm solutions.
- Make one change and test the model (run the program). Did it help?

- You may ask other teams for help as needed.

Prompt students as needed to start identifying the problems. Ask students questions like:

- Do you think the paper is moving the way it should?
- Is the pencil able to make the marks needed to represent the cuts?
- Does the model seem to be stable?

Note: there are four issues to identify and fix.

1. A paper feeder wheel is missing, causing the Y axis to not work properly.
2. The top of the CNC machine isn't correctly attached to the bottom.
3. The paper feeder gears are inverted, causing the paper to enter the CNC too quickly.
4. The pencil carriage isn't affixed, causing the X axis to work improperly.

Note: The Broken Lesson shows hints for what is wrong with the model. https://education.lego.com/en-us/lessons/prime-invention-squad/broken#Planitem0

Select one issue to investigate together as a group. Consider starting with the paper moving too fast. Brainstorm ideas for fixing the problem together. How can you slow the movement down without changing the program? Prompt students to look at how the gears are working and consider how to change them to slow the paper movement.

3. Explain

Discuss the program with students and why it does not seem to be working correctly.
Ask students questions like:

- What do you notice about the CNC machine when you run the program?
- How do you know it is a hardware problem?
- Why is testing your design important?
- How did you determine the issue was with the model and not the program?
- What parts of the machine do not seem to be working?
- As engineers, what is the first step we should take in fixing the machine? (Identifying problems; if more than one problem has been discovered, then deciding which problem to repair first.)

4. Elaborate

Allow students additional time to finish troubleshooting the model.

Ask the students to brainstorm solutions to the remaining issues. Allow students to retest the model running their program as needed to help troubleshoot the hardware.

After students have identified different issues, ask different groups to work on fixing the problems. Each group should share their fixes with other groups to ensure the CNC machine will work as needed. Students should retest their models to ensure they are working properly.

Share all solutions together as a group after the models are working correctly.

5. Evaluate

Discuss the program with students. Ask students questions like:

- How does testing help identify issues that need troubleshooting?
- How can you tell the need to troubleshoot hardware and not software?
- What are ways you can troubleshoot hardware issues?

Self-Assessment

Have students answer the following in their journals:

- What did you learn today about troubleshooting hardware issues?
- 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.

LEGO® Education SPIKE™ Prime

Debug-inator

Grade 6-8 | 45 minutes | Intermediate

Debug-inator

Practice troubleshooting hardware and program issues by designing and programming a new model.

Questions to Investigate

What debugging techniques can be used when designing a new model?

Materials needed

- SPIKE PRIME Set
- Device with SPIKE App installed
- Student journal

Prepare

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

KEY OBJECTIVES

Students will:

- Debug a software problem.

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-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

Debugging
Pseudocode
CNC

1. Engage

Spark a discussion about what a debug-inator is through brainstorming.

Ask students to come up with at least 3 things that their debug-inator will need to do. During this brainstorming session, students should gather as many ideas as possible and record them in their journals.

Prompt students as needed with questions like:

- Will your model need to sense anything?
- How will your model need to move?
- Will you need to utilize the console in any way?

Allow students to share their ideas from their brainstorming. Students should then decide on the final three main criteria that need to be included in their model.

2. Explore

Challenge students to design, build, program, test, and troubleshoot a new model that is a debug-inator meeting the criteria that they set.

Students should create their prototype being careful to include the three main criteria determined in the engage section. Students should practice their troubleshooting strategies while designing and building their model by testing the model's ability to move as intended.

Students will need to program their model. When creating their program, students should

- write a pseudocode program first to show the intended outcome of their program
- document their program using code comments with the #
- test the program, watching the console for error messages
- test the program using expected and unexpected outcomes or data

Allow students time to design, build, and program their models. Students should document any problems they encounter and how they fix or troubleshoot these issues.

3. Explain

Allow students to share their work. Discuss students' models and programs together.

Ask students questions like:

- What does your debug-inator do?
- What 3 expectations did you have for your model? How did you create something to meet these expectations?
- How did you program your model? Ask students to share the program using the code comments to explain it.
- What trouble did you have? Where did you find bugs? How did you fix them?

4. Elaborate

Have students finish their models and programs.

Allow students additional time to finalize their model and programs. Encourage collaboration between teams and sharing of ideas.

5. Evaluate

Discuss the program with students. Ask students questions like:

- What problems did you run into while creating your debug-inator?
- How did you test your model and program for errors? How did you troubleshoot the errors found?
- How did you determine if problems encountered were from the model or the program?

Self-Assessment

Have students answer the following in their journals:

- Ask students what challenges they encountered in creating their debug-inator.
- 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.
- What characteristics of a good teammate did you display today?

LEGO® Education SPIKE™ Prime

Ideas to Help with the Debug-inator

Grade 6-8 | 30-45 min. | Intermediate

Ideas to Help with the Debug-inator

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 Debug-inator lesson

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

Prepare

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

2. Explore

Have students work together to provide feedback to each other about the Debug-inator 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.

Impacting the Environment with Functions

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 our impact on the environment. Students will investigate how to create and use functions in their programs and the benefits of using functions. The lessons are designed in an order that allows students to progress in their skills and knowledge in the following areas:

- Create and call functions in a program
- Explore setting parameters in functions
- Investigate when it is best to use functions in a program
- Utilize code comment features to documents parts of a program
- Define and decompose a problem

Unit Learning Promise

In this unit, students will explore creating and calling functions within programs to create a block of code that can be reused multiple times in a program. Students will experience using functions, including setting parameters, while supporting a healthy environment for living things. Students will utilize pseudocode to support creating algorithms and code comments to document their programs.

Investigation Questions:
How can you create more efficient programs using functions?
How can functions allow for variability when called?

Unit Lessons

Lesson 1	Lesson 2	Lesson 3	Lesson 4	Lesson 5	Lesson 6
Turtle Trouble Time: 45 min	**Clean Up with Multiple Functions** Time: 45 min	**Clean Indicator** Time: 45 min	**Automate the Clean Up** Time: 90 min	**Taking Care of My Environment** Time: 90 min	**Ideas to Help with My Environment** 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-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.
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)		
6th Grade	**7th Grade**	**8th 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	RST.6-8.3 Follow precisely a multistep procedure when carrying out experiments, taking measurements, or performing technical tasks.
L.6.6 Acquire and use accurately grade-appropriate general academic and domain-specific words and phrases; gather vocabulary	L.7.6 Acquire and use accurately grade-appropriate general academic and domain-specific words and phrases; gather vocabulary	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	knowledge when considering a word or phrase important to comprehension or expression	knowledge when considering a word or phrase important to comprehension or expression

LEGO® Education SPIKE™ Prime

Turtle Trouble

Grade 6-8 | 45 minutes | Beginner

Turtle Trouble

Students will investigate how to create and use functions in a program.

Questions to investigate

- How can lines of code be used over and over again?

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 we impact the world around us.

Guide the discussion by showing a video of a sea turtle trapped in plastic bags or other trash items found in the ocean. Continue the discussion on other ways humans have an impact on the environment.

2. Explore

Students will investigate how an animal reacts to pollution to their habitat.

KEY OBJECTIVES

Students will:

- 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

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-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

Function

Direct students to the **BUILD** section in the SPIKE App. Here students can access the building instructions for the **Hopper** 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. In this lesson, the hopper will represent a sea turtle.

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
Prompt students to think about how they can program the sea turtle to move its flippers to swim or walk on land. Remember, turtles move slowly.

Ask students to write a program that will make the turtle's flippers move.

Sample Program:

```
from hub import port

import runloop

import motor_pair


async def main():

    # pair motors E and F

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

    # move straight for 1440 degrees at a velocity of 500

    await motor_pair.move_for_degrees(motor_pair.PAIR_1, 1440, 0, velocity=500)


runloop.run(main())
```

Allow students time to program their turtles.

Get Moving through Trash
Discuss how turtles move. Turtles do not swim or walk at one speed. When a turtle is frightened, it will swim faster. When it is struggling to get out of a plastic bag or other trash found in the ocean, the turtle might move its flippers back and forth.

Ask students to modify their program to allow different reactions or lines of code to run based on the situation. One way to do this is through using a function. A **function** is a block of reusable code that can be called at any time in the program.

Explain to students to set up, or define, a function start with a def line of code followed by a name for the function, parenthesis, and end with a colon. (Remind students to name the function something that makes sense to the action in the function.)

For example:

```
async def frightened():

    await motor_pair.move_for_degrees(motor_pair.PAIR_1, 1440, 0, velocity=1000)
```

Explain the sample program to students to see how the function is defined and then called in the program to be used.

Sample program:

```
from hub import port

import runloop

import motor_pair


# a function to move turtle quickly

async def frightened():

  # move straight for 1440 degrees at a velocity of 1000

  await motor_pair.move_for_degrees(motor_pair.PAIR_1, 1440, 0, velocity=1000)


# the main function

async def main():

  # pair motors E and F

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


  await frightened()
```

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

3. Explain

Discuss with students how the program worked.

Ask students questions like:

- Explain your function and how functions work?
- What is the benefit of using a function in your program?
- What is the difference between defining a function and calling a function?

Functions are used when a block of code needs to be repeated multiple times in a program. In our program, we created a function and called it one time. We can call this function as many times as we wish throughout the program. By creating a function, we shorten the program by using lines of code over and over again.

4. Elaborate

Challenge students to add two more functions to program the turtle to swim/walk at a normal speed and to struggle when getting trapped in a plastic bag.

Ask students to call each function when needed in the program.

Sample code:

```
from hub import port

import runloop

import motor_pair


async def frightened():

    # move straight for 1440 degrees at a velocity of 1000

    await motor_pair.move_for_degrees(motor_pair.PAIR_1, 1440, 0, velocity=1000)


async def normal():

    # move straight for 1440 degrees at a velocity of 250

    await motor_pair.move_for_degrees(motor_pair.PAIR_1, 1440, 0, velocity=250)


async def struggle():
    # move straight forward for 1440 degrees at a velocity of 1000 and then repeat backwards
    await motor_pair.move_for_degrees(motor_pair.PAIR_1, 1440, 0, velocity=1000)
    await motor_pair.move_for_degrees(motor_pair.PAIR_1, -1440, 0, velocity=1000)

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

    await frightened()

    await struggle()

    await normal()

runloop.run(main())
```

5. Evaluate

Teacher Observation:

Discuss the program with students.

Ask students questions like:

- What happened when you called the functions?
- Does it matter which order the functions are defined? Why or why not?
- What are the benefits of using functions?

Self-Assessment:

Have students answer the following in their journals:

- What did you learn today about using functions?
- 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.

LEGO® Education SPIKE™ Prime

Clean Up with Multiple Functions

Grade 6-8 | 45 minutes | Beginner

Clean Up with Multiple Functions

Students will investigate using a grabber to pick up items.

Questions to investigate

- How can grabber be programmed to pick up trash using multiple functions?

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
- Various items to pick up with the grabber (items that can simulate trash are ideal)

Prepare

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

KEY OBJECTIVES

Students will:

- Build and program a grabber to pick up items
- Modify the program to include multiple functions

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.

VOCABULARY

Function

1. Engage

Engage students in a discussion about trash and how trash affects our environment.

Consider showing images and videos of trash in different types of environments like the ocean, rivers, parks, etc. Have students think about how this trash affects the things that live in these environments.

Ask students to consider ways to clean up trash.

2. Explore

Students will investigate how to use more than one function in a program.

Direct students to the **BUILD** section in the SPIKE App. Here students can access the building instructions for the **Super Cleanup** 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.

Prompt students to think about how they program a tool to help them pick up trash.

Sample Program:

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

def is_pressed():
    return force_sensor.pressed(port.E)

def is_released():
    return not force_sensor.pressed(port.E)

async def main():
    while True:
        await runloop.until(is_pressed)
        #motor will close grabber
        await motor.run_to_absolute_position(port.A, 345, 500,
direction=motor.COUNTERCLOCKWISE )
        await runloop.until(is_released)
        #motor will release grabber
        await motor.run_to_absolute_position(port.A, 25, 500,
direction=motor.SHORTEST_PATH)

runloop.run(main())
```

Allow students to test the program by experimenting picking up a variety of materials with both grabbers. Discuss as a group which materials each grabber is best suited for picking up.

Using Multiple Functions
Prompt students to modify the program to include a function for a stronger grip and a second function for the lighter grip.

Discuss with students how to include functions that can be called based on which grabber attachment is being used. Students will want to make sure that the grabber opens and closes as needed depending on which attachment is used. Therefore, students can create a separate function for each.

Sample Program:

```
from hub import port

import runloop

import motor

import force_sensor


def is_pressed():

    return force_sensor.pressed(port.E)


def is_released():

    return not force_sensor.pressed(port.E)



async def grab_and_release(grip):

    await runloop.until(is_pressed)

    # motor will close grabber

    await motor.run_to_absolute_position(port.A, 345, grip,
direction=motor.COUNTERCLOCKWISE )

    await runloop.until(is_released)

    # motor will release grabber

    await motor.run_to_absolute_position(port.A, 25, grip)


async def main():

    while True:

        await grab_and_release(1000)
```

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

Note: Students will have to change grabber arms depending on which is being used at the time.

Allow students time to test their program using both grabbers to see if the functions are working properly. Remind students of the importance of how they call their functions to be used in their program to ensure they use the right function at the right time.

3. Explain

Discuss with students how each of the programs worked.
Ask students questions like:

- How do the two functions compare?
- Why might someone want to add a function to a program?
- What do you need to consider when using multiple functions in your program?

4. Elaborate

Challenge students to modify their programs to allow for picking up objects differently with each grabber.

Students should modify their program to include a function specific to each grabber. This will allow students to call the function depending on which grabber arm is being used rather than using the same program for each grabber arm.

5. Evaluate

Teacher Observation:
Discuss the program with students.
Ask students questions like:

- Why would you need a separate function for each grabber arm?
- What did you change about the function for each grabber arm to make it more specific to the way that grabber arm worked?
- Why is it helpful to define multiple functions in a program that can be called later as needed?

Self-Assessment:

Have students answer the following in their journals:

- What did you learn today about using multiple functions 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.

Clean Indicator

Grade 6-8 | **45 minutes** | **Intermediate**

Clean Indicator

Students will investigate the role of parameters in functions.

Questions to investigate

- How can you use parameters within your functions?

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 creating graphs and ways to visualize information or data.

Ask students to stand up and be ready to vote by moving their feet. Pose a question to students that has several options that students can chose from. Students will indicate their answer by moving to the spot that is for their choice. This could be different corners in the room or any area you designate. Once students have made their vote, have them move together in a line to create a human bar graph.

Some example questions might be:

- What is your favorite ice cream flavor? (choose 4 flavors to list)

KEY OBJECTIVES

Students will:

- Create functions that use parameters
- Investigate debugging functions and functions that use parameters

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

function
parameter

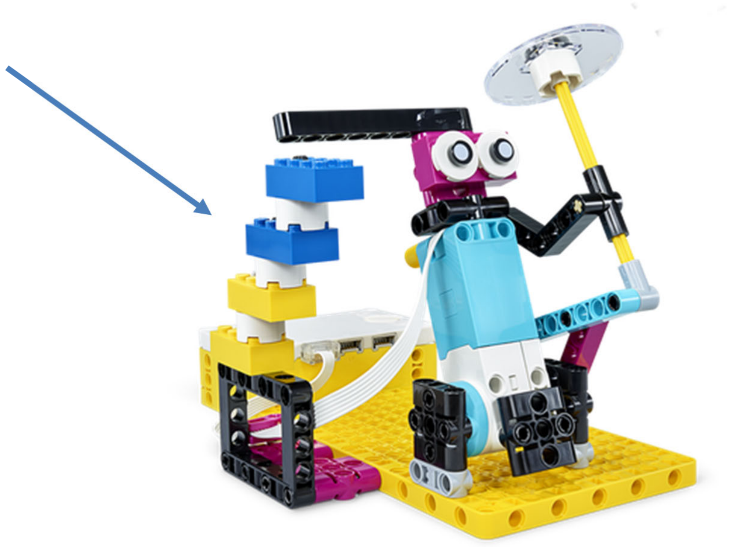
- What is your favorite book genre? (choose 3-4 to list)
- Which character are you most like? (choose a book or movie)

2. Explore

Students will investigate using parameters in their functions.

Direct students to the **BUILD** section in the SPIKE App. Here students can access the building instructions for the **Wind Indicator** 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.

Ask students to make a change to the model design. Ask students to change the stack of two yellow and two blue bricks to the side should be replaced with red on the bottom, then yellow, then blue, and green at the top. Students also need to add the color sensor in a place that will not obstruct the movement of the wind indicator.



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.

Prompt students to think about how they can use a function for the indicator to show if the environment is clean (at the green level) or not (at the red level). The indicator should move to the appropriate color brick in the stack to show the level.

Ask students to write a pseudocode program for what the function could look like in the program. Pseudocode is writing in words what you want the program to do.
Sample Pseudocode:

- Define function
- Color sensor detect level of clean
- Move motor to the position based on color reading
- Print the color in the console

Discuss Pseudocode together.

Adding Parameters
Challenge students to add parameters to their program and "scan" their environment to indicate how clean it is.

Students will need a red, violet, yellow, and green 2x4 bricks from their set that will be scattered on the table as their environment to scan. Additionally, ask students to add the color sensor to their model by connecting it to the hub, but not attaching it to the model.

Ask students to modify their program to set parameters in their function for "scanning" the environment (extra bricks) using their color sensor. A parameter is like a variable but can only be used in a function. The parameter is data that is inputted from the user. It acts as a place holder for the new information that can be added when the function is called.

Sample Program:

```
from hub import port, button

import runloop

import motor

import color_sensor

import color


# define function for reporting the clean level

def report(height, color):

    runloop.run(motor.run_to_absolute_position(port.A, height, 500))

    print('Clean Level', color)
```

```
async def main():
    while True:
        await runloop.until(lambda: button.pressed(button.LEFT))
        #when sensing a color, the height is set
        if color_sensor.color(port.E) is color.RED:
            report(70, 'Red')
        elif color_sensor.color(port.E) is color.YELLOW:
            report(60, 'Yellow')
        elif color_sensor.color(port.E) is color.BLUE:
            report(50, 'Blue')
        elif color_sensor.color(port.E) is color.GREEN:
            report(25, 'Green')
        #if it doesn't sense a color, it moves to the 0 position
        else:
            await motor.run_to_absolute_position(port.A, 0, 1000)
            print("No Reading")

runloop.run(main())
```

Allow students time to explore their program and take multiple readings of the environment.

3. Explain

Discuss with students how the program worked.
Ask students questions like:

- Why did we add parameters to our function?
- What is the different between a parameter and a variable?
- How did using a function in your program make the program more efficient?

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, button
import runloop
import motor
import color_sensor
import color

# define function for reporting the clean level
def report(    ):
    runloop.run(motor.run_to_absolute_position(port.A, height, 500))
    print('Clean Level', color)

async def main():
    while True:
        await runloop.until(lambda: button.pressed(button.LEFT))
        #when sensing a color, the height is set
        if color_sensor.color(port.E) is color.RED:
            report(70, 'Red')
        elif color_sensor.color(port.E) is color.YELLOW:
            report(60, 'Yellow')
        elif color_sensor.color(port.E) is color.BLUE:
            report(50, 'Blue')
        elif color_sensor.color(port.E) is color.GREEN:
            report(25, 'Green')
        #if it doesn't sense a color, it moves to the 0 position
        else:
            await motor.run_to_absolute_position(port.A, 0, 1000)
            print("No Reading")

runloop.run(main())
```

```
Traceback (most recent call last):

File "Project 6", line 26, in <module>

File "Project 6", line 20, in main

TypeError: function takes 0 positional arguments but 2 were given
```

The error should draw attention to line 20, which does not seem to have an error.

The actual error is occurring in line 5 where the function is defined. When defining the function, the parameters must be set so that when the function is called the value for the parameter can be provided. The value for the parameter provided now is not able to reference a parameter provided.

Debug activity 2:

```
from hub import port, button

import runloop

import motor

import color_sensor

import color


# define function for reporting the clean level

def report(height, color):

    runloop.run(motor.run_to_absolute_position(port.A, height, 500))

    print('Clean Level', color)


async def main():

    while True:

        await runloop.until(lambda: button.pressed(button.LEFT))

        #when sensing a color, the height is set

        if color_sensor.color(port.E) is color.RED:

            report(70)

        elif color_sensor.color(port.E) is color.YELLOW:

            report(60, 'Yellow')

        elif color_sensor.color(port.E) is color.BLUE:

            report(50, 'Blue')

        elif color_sensor.color(port.E) is color.GREEN:

            report(25, 'Green')

        #if it doesn't sense a color, it moves to the 0 position
        else:

            await motor.run_to_absolute_position(port.A, 0, 1000)

            print("No Reading")

runloop.run(main())
```

```
Traceback (most recent call last):

File "Project 6", line 26, in <module>

File "Project 6", line 14, in main

TypeError: function takes 2 positional arguments but 1 were given
```

Note: the line reference in this error message will depend on which color you scan.

This error should indicate to students to check line 14. Students should recognize that the function is being called with a reference to the parameter. However, there should be two arguments but only 1 is given, which is causing the error. The 70 is the first, but it also needs a color.

Ask students to consider other comment errors that might occur in this type of program and ideas for debugging when they do get errors.

5. Evaluate

Teacher Observation:

Discuss the program with students.

Ask students questions like:

- What type of errors should you watch for when programming with functions and parameters?
- What are ways that you can use parameters within your functions?

Self-Assessment:

Have students answer the following in their journals:

- What did you learn today about using parameters in functions?
- 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.

LEGO® Education SPIKE™ Prime

Automate the Clean Up

Grade 6-8 | 90 minutes | Intermediate

Automate the Clean Up

Students will investigate how to create a sorting robot and program it to use functions for an automated clean-up process.

Questions to investigate

- How can functions be used in programming for efficiency?

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.

KEY OBJECTIVES

Students will:

- 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

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-14 Create procedures with parameters to organize code and make it easier to reuse.
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.

1. Engage

Engage students in a discussion about how to automate the clean-up effort.

Discuss with students the ways that they need to work to keep the environment clean. Think of ways that this can be automated using robotics to help keep things clean. Discuss that some materials are trash, but many can be recycled. Have students identify various materials that would be considered trash and items that could be recycled.

Prompt students to think about ways to sort the materials into trash and recyclable materials.

2. Explore

Students will investigate how to automate clean up and sorting materials into trash and recycling.

Direct students to build the **Quality Check Robot** 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.

Prompt students to think about how to program their sorting robot to identify if a material is trash or recyclable. Students should create a function for checking the color of materials then sorting the recyclable materials into the blue area in front
of the robotic sorter while throwing the trash away.
Note: Students will need to assign a color to trash and a color to recycle. The colors used in the building instructions are yellow and violet. Consider letting students change these to other colors if they want.

Sample Program:

```
from hub import port
import runloop
import color_sensor
import motor
import color
from app import sound

async def main():
    await motor.run_to_absolute_position(port.A, 0, 500)
    await motor.run_to_absolute_position(port.F, 240, 500)
    await motor.run_to_absolute_position(port.A, 90, 500)
    await motor.run_to_absolute_position(port.F, 25, 500)

    await check_color()

    await motor.run_to_absolute_position(port.A, 0, 500)
    await motor.run_to_absolute_position(port.F, 240, 500)
    await motor.run_to_absolute_position(port.A, 270, 500)
    await motor.run_to_absolute_position(port.F, 25, 500)

    await check_color()

    await motor.run_to_absolute_position(port.A, 0, 500)
    await motor.run_to_absolute_position(port.F, 240, 500)

async def check_color():
    await motor.run_to_absolute_position(port.F, 235, 500)

    if color_sensor.color(port.D) is color.MAGENTA:
        await motor.run_to_absolute_position(port.A, 0, 500)
        await motor.run_to_absolute_position(port.F, 25, 500)
        await sound.play('Triumph')
        await motor.run_to_absolute_position(port.F, 240, 500)
    else:
        await sound.play('Oops')
        await motor.run_to_absolute_position(port.F, 25, 500)
        for x in range(3):
            await motor.run_for_degrees(port.F, -100, 1000)
            await motor.run_for_degrees(port.F, 100, 1000)

runloop.run(main())
```

Allow students time to create and run their program. Remind students to watch the console for error messages during their programming. Students may need to debug their program as they test it.

3. Explain

Discuss with students how the program worked.
Ask students questions like:

- Why did we use a function in this program? How did it make the program more efficient?
- What are other ways that functions could be used in your program?
- If students included parameters, what parameters were used and why?
- Show students the sample program used here and ask them to provide code comments (using the #) that should be included to make the program easier to ready.

4. Elaborate

Challenge students to modify their program to include a second function in order to make their program more efficient.

Discuss the idea of making programs more efficient with students. Taking time to define variables, functions, etc. at the beginning of the program can allow students to make the body of the program shorter and easier to execute.

Discuss ways to make the program more efficient. One way could be to include a second function that will help the robotic checker know what to do with the materials after they are checked. Students should try to reduce the lines of code being used after their function is defined. If there are currently 12 lines of code after the function, could they cut that in half with a new program. Challenge students to see how efficient they can make their program. Remind students to include their code comments.

5. Evaluate

Teacher Observation:

Discuss the program with students.

Ask students questions like:

- What happened when you added an additional function to your program?
- Why is efficiency important to creating large programs?
- What are the different ways to use functions in your programs? What did you learn from how other students approached this challenge?

Self-Assessment:

Have students answer the following in their journals:

- What did you learn today about making programs efficient?
- 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.

Taking Care of My Environment

Grade 6-8 | 90 minutes | Advanced

Taking Care of My Environment

Students will investigate local environmental issues and possible ways to help solve the local environmental issues.

Questions to investigate

- How can robots be used to help the environment?

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.

KEY OBJECTIVES

Students will:

- Design, build, and program an environmental helper to take care of a local area

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-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.

1. Engage

Engage students in a discussion about their local environments.

Ask students to consider local areas like parks, the school grounds, areas near water like a lake or river. Prompt a discussion on these areas and the animals that live there. What are some ways that students could help those living things by creating a robot that can help their environment?

As a group, discuss students' ideas. Then launch into a brainstorming session.

Brainstorm
Brainstorm different ideas for a robotic helper that could benefit the environment and living things in it.

In small groups, students should brainstorm several ideas for creating a new type of robotic helper, drawing from ideas used in previous lessons. Students might consider adding something to indicate the cleanliness level, or to sort the trash and recycling, or pick up items. Students might try to design something new and create a helper that pushes trash into one area while spreading seeds behind it. Each group should come up with unique ideas that support their specific needs.

2. Explore
Students will design, build, and program a robotic helper clean up the environment.

Design and Choose the Right Idea
Students will design, build, and program a robotic helper for their environment. The constraints 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 at least one function in the program.
5. It must include parameters to work with the function.

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 ideas as they go, making improvements where needed. Students should test and evaluate their designs against the design criteria set and their flowcharts as they start making their solutions.

Ensure students use sketches and photos of their models to record in their design journey.

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 an environmental helper using at least one function? Ask students to share their program comments to explain.
- What decisions did you have to make while creating your design?
- How did you incorporate parameters?
- 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 get from the sharing session.

Allow students to share out their final designs and explain how they met the criteria.

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 functions and parameters did you include and why?

Self-Assessment:

Have students answer the following in their journals:

- What did you learn today about creating your own design based on given criteria?
- 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.

LEGO® Education SPIKE™ Prime

Ideas to Help with Taking Care of My Environment

Grade 6-8 | 30-45 min. | Intermediate

Ideas to Help with Taking Care of My Environment

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 Taking Care of My Environment lesson

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

Prepare

- Ensure SPIKE Prime hubs are charged, especially if connecting through Bluetooth.
- Ensure students have their built model from the Taking Care of My Environment 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.

2. Explore

Have students work together to provide feedback to each other about the Taking Care of My Environment 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:

5. Tell something they really like. This could be the model, program, or design.
6. Tell something that worked well.
7. Share something the group could try differently.
8. 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.

Compound Conditionals and Logic Operators

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 aspects of cybersecurity and digital safety. Students will explore how to keep systems safe through designing, building, and programming models with security measures while utilizing logic operators and nested conditional statements. Students will identify the importance of safety by utilizing both physical and digital security measures. The lessons are designed in an order that allows students to progress in their skills and knowledge in the following areas:

- Investigate cyber security through setting passwords
- Explore physical security measures
- Investigate nested conditional statements
- Investigate using logic operators to combine conditions
- Investigate using sensors for security
- Create two-step security programs
- Simulate a security device with both physical and digital safety features

Unit Learning Promise

In this unit, students will explore how to program conditional statements and logic operators. They will develop an understanding for when to use conditional statements and logic operators as they create different prototypes utilizing both physical and digital safety measures. Students will incorporate their new knowledge and skills in a cybersecurity simulation.

Investigation Questions:
How can you use conditional statements and logic operators as safety measures? Why is cybersecurity important for digital wellness?
How can physical and digital safety measures be combined?

Unit Lessons

Lesson 1	Lesson 2	Lesson 3	Lesson 4	Lesson 5	Lesson 6
Password Protection	**Make it Physically Safe**	**Make a Safer Safe**	**Security Operating with Logic**	**Escape Room**	**Ideas to Help with the Escape Room**
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-NI-05 Explain how physical and digital security measures protect electronic information.
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.
2-IC-22 Collaborate with many contributors through strategies such as crowdsourcing or surveys when creating a computational artifact.

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)		
6th Grade	**7th Grade**	**8th 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	RST.6-8.3 Follow precisely a multistep procedure when carrying out experiments, taking measurements, or performing technical tasks.
L.6.6 Acquire and use accurately grade-appropriate general academic and domain-specific words and phrases; gather vocabulary	L.7.6 Acquire and use accurately grade-appropriate general academic and domain-specific words and phrases; gather vocabulary	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	knowledge when considering a word or phrase important to comprehension or expression	knowledge when considering a word or phrase important to comprehension or expression

LEGO® Education SPIKE™ Prime

Password Protection

Grade 6-8 | 45 minutes | Beginner

Password Protection

Students will create a password using bricks to explore security measures.

Questions to investigate

- How can you create secure passwords?

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
- Device with the SPIKE App installed.
- Student journals

Prepare

- Each student will need 5 colors of 2x4 bricks from the SPIKE Prime Set.
- Ensure SPIKE Prime hubs are charged, especially if connecting through Bluetooth.

KEY OBJECTIVES

Students will:

- Investigate cyber security through setting passwords
- Explore physical security measures

STANDARDS

CSTA

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.

1. Engage

Ignite a discussion around locks. Where do we use a lock? Why do we use a lock?

Explain to students that sometimes people use a lock when they want to keep something safe. Ask students the following questions:

- What is a password?
- Where do we use passwords?
- How can a strong password keep our information safe?
- Why do we not share our passwords?

Allow students to share examples of passwords reminding them to not share any actual passwords.

2. Explore

Students will investigate creating more secure passwords.

Ask students to select 5 different colors of 2x4 bricks from their set.

Working in pairs, assign each team a Partner A and a Partner B. Partner A should write 3 numbers in his/her journal without Partner B seeing what they are writing. Partner B should guess the first number written. Partner A sees how many guesses it takes and writes the number of guesses underneath the guessed number. Repeat for the second and third number.

Switch roles. Partner B writes 3 numbers in his/her journal. Partner A guesses the numbers and then Partner B writes how many guesses it takes.

Discuss the outcomes together as a class. Ask students how many possible answers were there for each number. 10 possible answers– 0,1,2,3,4,5,6,7,8,9. How many possible answers are there for a 3-digit passcode? (10 x 10 x 10) [numbers 000 to 999]

Discuss together how you could make the password even more secure. Students should recognize that adding another number to make a four-digit password will be more secure (and more so with each number added).

Brick Password

Challenge students to create a brick-based password by creating a 5-step password using their bricks.

Each partner should create a password by first creating a stack of bricks with the first step of the password or first color being at the bottom of the stack and the last at the top of the stack. Students can make the stack in any order they want.

3. Explain

Discuss with students how their passwords worked.
Ask students questions like:

- Which password was more difficult to guess, the 3-digit number or the 5-brick password?
- What made guessing the last few colors of the brick password easier?
- What could you do to make the password even stronger?

4. Elaborate

Challenge students to come up with a more secure password.

Ask students to remove the hub from their SPIKE Prime set. Challenge students to create a password and indicate if the password is correct or not on the hub using images, words, or sounds.

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.

Brainstorm ideas for using the hub to create a password. Ideas might include setting the motion sensor orientation to a certain position (like up, forward, left), using the motion sensor gestures (like shake or tap), or pressing one of the buttons. Students might then think about showing a smiley face if the password is correct or print good job on the hub.

Ask students to write a pseudocode for how they plan to use the hub as part of their password. Students should then create their program according to the pseudocode. Remind students to use code comments in the program to indicate what should happen. Students should also watch the console for error messages as they create their program.

5. Evaluate

Teacher Observation:

Discuss the program with students.

Ask students questions like:

- What happened when you made your passwords more difficult?
- Why do we use passwords?
- How can you create a physical password?

Self-Assessment:

Have students answer the following in their journals:

- What did you learn today about using passwords?
- 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.

Make it Physically Safe

Grade 6-8 | **45 minutes** | **Beginner**

Make it Physically Safe

Students will use conditional statements to create a physically protected place to store information.

Questions to investigate

- How can vulnerable information be kept safe in a physical space?

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 safes and protecting physical items.

Spark a discussion with students about ways to physically protect important information. Prompt students to think about where this information could be kept and how locks might be used.

Ask students questions like:

- What type of devices can be used to lock something away?
- What makes locks strong?
- What are some ways that you can create a lock or a device that locks and unlocks (i.e. using a key, a password, etc.).
- What can make a locked door or box even stronger?

KEY OBJECTIVES

Students will:

- Investigate nested conditional statements
- Explore physical security measures

STANDARDS

CSTA

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.

Let students complete research and view images of different types of locks as needed.

2. Explore

Students will investigate different types of grabbers to consider the best tool for cleaning up trash.

Direct students to the **BUILD** section in the SPIKE App. Here students can access the building instructions for **Safe Deposit Box** 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.

> **Note**: this model is also used for the next lesson Make a Safer Safe. The motor attached will not be used in this lesson, but will be in the next.

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.

Have students investigate how the safe works to lock and then unlock when the appropriate action is taken. Ask students to type the sample program into their programming canvas and run it.

Sample program:

```
from hub import port, sound, light_matrix, button

import motor

import runloop


def pressed():

    return button.pressed(button.LEFT)


async def main():

    await sound.beep(262, 200)

    await sound.beep(523, 200)


    #this locks the safe

    motor.stop(port.B)
    motor.reset_relative_position(port.B, 0)

    await motor.run_to_absolute_position(port.C, 270, 500)

    light_matrix.show_image(light_matrix.IMAGE_NO)


    #this unlocks the safe

    await runloop.until(pressed)

    await sound.beep(523, 200)

    await motor.run_for_time(port.C, 1000, 500)

    light_matrix.show_image(light_matrix.IMAGE_YES)

    await runloop.sleep_ms(5000)


runloop.run(main())
```

Discuss the program together as a group paying attention to how the safe currently locks. Ask students to think about how protected any information in the safe would currently be. Note that there is only one step of protection using the safe.

Time for an Upgrade
Students will explore a nested loop to create a more secure safe.

Challenge students to change their code to include a **nested conditional statement**. Explain to students something is nested when one part is put in another part. Loops and conditional statements can be nested within each other to create a more complex program. Ask students to begin by creating a flowchart to show how the program should work. Allow students time to complete their flowcharts and discuss them together as a group.

Allow students time to create their new program that includes the nested loop. Remind students to use code comments in their program to explain what the program should do. Students should also watch the console for error messages.

Sample Program:

```
from hub import port, sound, light_matrix, button

import runloop

import motor
```

```
def pressed():

    return button.pressed(button.LEFT)

async def main():

    await sound.beep(262, 200)

    await sound.beep(523, 200)

    #this locks the safe

    motor.stop(port.B)

    motor.reset_relative_position(port.B, 0)

    await motor.run_to_absolute_position(port.C, 270, 500)

    light_matrix.show_image(light_matrix.IMAGE_NO)

    #this unlocks the safe

    while True:

        print('Open Me')

        if motor.relative_position(port.B) > 180:

            await runloop.until(pressed)

            await sound.beep(523, 200)

            await motor.run_for_time(port.C, 1000, 500)

            light_matrix.show_image(light_matrix.IMAGE_YES)

            await runloop.sleep_ms(5000)

            break

        else:

            print('Still Locked!')

            await runloop.sleep_ms(1000)

    print('You may enter!')

runloop.run(main())
```

Allow students time to share their final programs.

3. Explain

Discuss with students how the program worked.
Ask students questions like:

- How does the safe work to provide security?
- How safe do you think the safe is?
- How does the nested conditional statement work?
- What was difficult about this challenge? Where did you run into programming trouble?
- What does the line of code with "break" do?

Explain to students that the break statement terminates the loop.

4. Elaborate

Challenge students to change the order of the two step protection and identify the error in the program.

Ask students to consider how they can change the order of the steps for opening the safe. Share the sample program for students to investigate. There are no errors in this program. However, the program is not working to open the safe. Ask students to identify where the problem is and debug the program.

Sample Program:

```
from hub import port, sound, light_matrix, button

import runloop

import motor


def pressed():

    return button.pressed(button.LEFT)


async def main():

    await sound.beep(262, 200)

    await sound.beep(523, 200)
    #this locks the safe
    motor.stop(port.B)
    motor.reset_relative_position(port.B, 0)
    await motor.run_to_absolute_position(port.C, 270, 500)
    light_matrix.show_image(light_matrix.IMAGE_NO)

    #this unlocks the safe
    while True:
        print('Open Me')
        await runloop.until(pressed)
        if motor.relative_position(port.B) > 180:
            await sound.beep(523, 200)
            await motor.run_for_time(port.C, 1000, 500)
            light_matrix.show_image(light_matrix.IMAGE_YES)
            await runloop.sleep_ms(5000)
            break
        else:
            print('Still Locked!')
            await runloop.sleep_ms(1000)

    print('You may enter!')
runloop.run(main())
```

Ask students how this program is different?

Note: You must press both buttons at the same time to get the safe to open.

Allow time for students to re-work the program. Ask groups to share their final program.

5. Evaluate

Teacher Observation:

Discuss the program with students.
Ask students questions like:

- What happened when you created a nested conditional statement?
- How does nesting the conditional statements make the program more complicated and the safe safer?
- What are some errors to watch for when programming compound conditionals?

Self-Assessment:

Have students answer the following in their journals:

- What did you learn today about using nested 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.

Make a Safer Safe

Grade 6-8 | 45-90 minutes | Intermediate

Make a Safer Safe

Students will investigate using compound conditional statements with logic operators to create a physically protected place to store information.

Questions to investigate

- How can multiple conditions be combined into one single condition?
- How can logic operators be used and outcomes predicted?

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 Safe Deposit Box model built from the Make it Physically Safe lesson.

KEY OBJECTIVES

Students will:

- Investigate using logic operators to combine conditions
- Explore physical security measures

STANDARDS

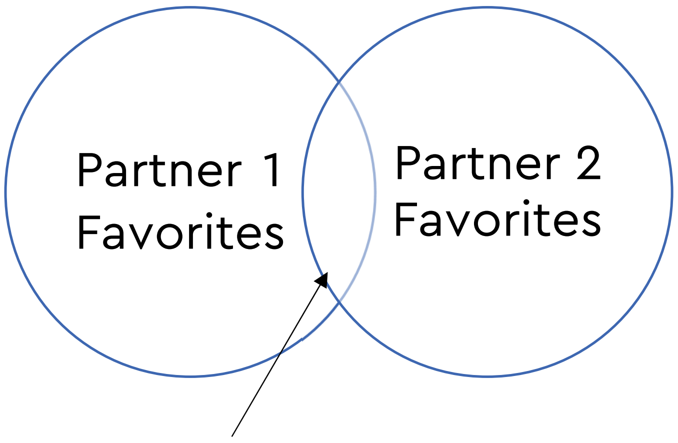
CSTA

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.

1. Engage

Engage students in thinking about how to combine multiple ideas into one.

Launch a discussion with students about their favorite things to eat. After students have a minute to think about their own favorites, have students come together in pairs to compare their lists. Ask students to create a Venn Diagram showing their favorite foods.



Shared favorites

Discuss the Venn Diagrams together. What did the two students have in common? Ask students to consider if they were eating lunch together and had to share, what they would want to eat. Would they want to eat something from one of the individual lists or something they both like?

Discuss students' ideas together. Have students think about how they could describe what they would eat and not eat encourage the use of the words and, or, and not. For example, I would eat something from my list or the shared list, not the other person's list.

2. Explore

Students will investigate combining two conditional statements to create a compound conditional statement using logic operators.

Direct students to the **BUILD** section in the SPIKE App. Here students can access the building instructions for **Super Safe Deposit Box** model. Ask students to build the model. If students have the Safe Deposit Box model built from the Make it Physically Safe lesson, then the students will only need to build and attach the arm. 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.

Combine Conditional Statements
Have students investigate how the safe works.

Sample program:

```
from hub import sound, port, light_matrix, button

import app

import runloop

import time

import motor


async def main():

    await sound.beep(262, 200)

    await sound.beep(523, 200)


    #this locks the safe

    await motor.run_for_time(port.C, 1000, -500)

    await motor.run_to_absolute_position(port.B, 0, 500, stop=motor.COAST)

    motor.reset_relative_position(port.B, 0)

    await motor.run_to_absolute_position(port.E, 0, 500)

    light_matrix.show_image(light_matrix.IMAGE_NO)


    #this unlocks the safe

    await unlock()


async def unlock():

    start_time = time.ticks_ms()

    while not button.pressed(button.LEFT) and motor.relative_position(port.B) < 180:

        await sound.beep(262, 200)
        await motor.run_for_degrees(port.E, 15, 500)

        await runloop.sleep_ms(800)

        if time.ticks_diff(time.ticks_ms(), start_time) > 5000:

            await app.sound.play('Bonk')

            return

        light_matrix.show_image(light_matrix.IMAGE_NO)


    light_matrix.show_image(light_matrix.IMAGE_YES)

    await motor.run_to_absolute_position(port.E, 0, 500)

    await motor.run_for_time(port.C, 1000, 500)

    await app.sound.play('Wand')


runloop.run(main())
```

Discuss the program together as a group paying attention to how the safe currently locks. Ask students to think about how protected any information in the safe would currently be.

Review the sample program carefully to identify new lines of code. Prompt students to look at the while statement.

```
if button.pressed(button.LEFT) and not motor.relative_position(port.B) < 180:
```

Students should recognize that there are two conditional statements being addressed in this line. Introduce the term **compound conditional statement** to students. Explain that a compound conditional statement is a statement that combines two Boolean expressions. The conditional statement will not be true unless all conditions listed are met.

Explain to students that **logic operators** are used to combine conditional statements. Point out the two logic operators used in this statement – "and" and "not. "or" is the third type of logical operator.

The program sets two conditions using the **and** logical operator. Explain to students the word and (lowercase) is used to indicate that both parts of a condition must be true for the entire condition to be true. Also explain that a

logic operator allows you to combine more than one Boolean expression or values. These will be evaluated in the program as one Boolean value.

Students will need to type this program into the programming canvas. Students should run the program. Discuss the program together after students run it.

3. Explain

Discuss with students how the program worked. Ask students questions like:

- How can you combine two conditions into one?
- What are the different types of outcomes that you can have using the and logic operator?

Work together to create a flowchart that explains this program. If one or both conditions are false, then the entire condition is false.

Introduce **truth tables** to students. These tables are helpful in Boolean Expressions to list all possible outcomes of program using logical operators.

Example Truth Tables:

And Truth Table

Condition 1 (true/false)	and	Condition 2 (true/false)	Total Condition
True	and	True	True
True	and	False	False
False	and	True	False
False	and	False	False

Or Truth Table

Condition 1 (true/false)	or	Condition 2 (true/false)	Total Condition
True	or	True	True
True	or	False	True
False	or	True	True
False	or	False	False

Not Truth Table

Condition (true/false)	Total Condition
True	False
False	True

Create a truth table for the sample program. Fill in the possible outcomes based on the combination of how each condition is evaluated. Prompt students to explain why each total condition evaluates as true or false.

Discuss with students how flowcharts and truth tables can be helpful resources when creating programs with combined conditions.

4. Elaborate

Challenge students to create different conditions to test the logic operator in their conditional statement.

Using their truth table as reference, ask students to try different combinations to create security to test the truth. Students should run their program several times when condition 1 and 2 are changed from true to false in all the combinations in the table to see if the outcomes are all true.

Discuss the outcomes together as a group.

5. Evaluate

Teacher Observation:

Discuss the program with students.

Ask students questions like:

- What happened when you changed the logic operators in your program?
- How can truth tables be used to support creating programs with logic operators?
- How does using logic operators allow you to create additional security?

Self-Assessment:

Have students answer the following in their journals:

- What did you learn today about using logic operators and combining two conditions?
- 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.

LEGO® Education SPIKE™ Prime

Security Operating with Logic

Grade 6-8 | 45-90 minutes | Intermediate

Security Operating with Logic

Students will investigate how to use logic operators to make a security alarm using sensors.

Questions to investigate

- How can logic operators be used with sensors to create a more complex conditional 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 a conversation about security.

Ask students to think about places that they keep secure (i.e. phones, computers, doors, etc.). Have students share how these areas are kept secure.

Launch a discussion about physical security options and ways to detect "break ins". Prompt students to think about ideas like detecting movement or setting passwords.

2. Explore

Students will investigate using two steps for more security.

KEY OBJECTIVES

Students will:

- Investigate using sensors for security
- Create two-step security programs

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 the **BUILD** section in the SPIKE App. Here students can access the building instructions for the **Quality Check Robot** 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.

Ask students to create a security device that alerts your movement and then requires a color-coded password. Discuss the two sensors included in the model and how these might be used to create a physical alert device.

Ask students to write a pseudocode program that explains how their program should work. The program should include:

- using the distance sensor first to detect motion
- then using the color sensor as a pass code
- and include at least one logic operator

Ask students to use their pseudocode to create their program. Remind students to use code comments in their program to indicate the expected actions.

Allow students time to test and modify their program as needed. Remind students to watch the console for error messages.

3. Explain

Discuss various program examples together as a class to find what is effective. Ask students questions like:

- How can sensors be used to create a security device?
- How did the logic operator work in your program?
- How can you create two step security in your device?

4. Elaborate

Challenge students to use a different logic operator in their security device.

Ask students to write a new pseudocode to change the way the program will work by using a different logic operator. Students can choose to add a second logic operator to their program or replace the one in their current program.

Allow students time to modify their program according to the new pseudocode. Students should test and refine their program. Remind students to watch the console for error messages while testing their program.

5. Evaluate

Teacher Observation:

Discuss the program with students.
Ask students questions like:

- What happened when you changed the logic operator?
- When adding more than one logic operator, what are potential bugs to consider?
- How were the sensors used to create security? What are other ways you could use sensors in your programs (other sensors to use)?

Self-Assessment:

Have students answer the following in their journals:

- What did you learn today about using multiple steps of security?
- 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.

LEGO® Education SPIKE™ Prime

Escape Room

Grade 6-8 | 90 minutes | Advanced

Escape Room

Students will create and program their own escape room security system.

Questions to investigate

- How can conditions be set to create a secure "room" to break out of?

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 escape rooms.

Ask students if they have ever participated in or heard of a breakout or escape room. Consider showing images or videos to students. Explain that in these games participants use clues and complete puzzles to figure out how to get out of the locked room.

KEY OBJECTIVES

Students will:

- Create a security device to simulate a break out game
- Design a device that meets given constraints

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.

Brainstorm

Brainstorm different ideas of puzzles that could be used in an escape room.

As a group, brainstorm several ideas for creating a secure device that could be used in a breakout or escape room.

Ask students to then work in their smaller groups to create a short survey to ask other students what they would like in a security system or something to break out of. 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 escape device 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 puzzle game to use in an escape room.

Design and Choose the Right Idea

Students should consider the input from their surveys and start to design their security device that will be used in a breakout room.

Students will design, build, and program a device that could be used in a breakout or escape room. The constraints 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 logic operator in the program
5. It must include at least two steps to crack the code.

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 ideas as they go, making improvements where needed. Students should test and evaluate their designs against the design criteria set and their flowcharts as they start making their solutions.

Ensure students use sketches and photos of their models to record in their design journey.

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 security device with at least two steps? 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 get 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 logic operators did you include and why?

Self-Assessment:

Have students answer the following in their journals:

- What did you learn today about creating your own design based on given criteria?
- 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.

LEGO® Education SPIKE™ Prime

Ideas to Help with the Escape Room

Grade 6-8 **30-45 min.** **Intermediate**

Ideas to Help with the Escape Room

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 Escape Room lesson

Prepare

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

KEY OBJECTIVES

Students will:

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

STANDARDS

CSTA

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

VOCABULARY

Feedback
Specific
Positive
Negative

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.

2. Explore

Have students work together to provide feedback to each other about the Escape Room 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.

Data and Math Functions

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 exploring different types of transportation. Students will investigate data types and mathematical functions and how they can be utilized in programming. The lessons are designed in an order that allows students to progress in their skills and knowledge in the following areas:

- Investigate types of variables and data
- Investigate ways to take in data from sensors
- Utilize mathematical functions in a program
- Use conditional statements in a program using sensors and motors
- Apply sensors to real-life problems

Unit Learning Promise
In this unit, students will explore different data types while investigating various types of transportation. Students will look at ways to gather data and how to use data collected from sensors. Students will also explore different mathematical functions and how to integrate into their programs. Students will experience how to effectively make conditional statements in a program using sensors and motors. Students will explore the real-world use of sensors as a method of data collection.

Investigation Questions:
How can you integrate mathematical functions into programs?
What are ways to gather and use data in your programs?
How can data be used to solve real-world problems?

Unit Lessons

Lesson 1	Lesson 2	Lesson 3	Lesson 4	Lesson 5	Lesson 6	Lesson 7
Getting Moving to Get Data Time: 45 min.	**Bike Riding for Data** Time: 45 min.	**Counting Your Steps** Time: 45 min.	**Make It Move** Time: 45 min.	**Parking Lot** Time: 45 min.	**My Transportation** Time: 45 min.	**Ideas to Help with My Transportation** 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-12 Design and iteratively develop programs that combine control structures, including nested loops and compound conditionals.
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-18 Distribute tasks and maintain a project timeline when collaboratively developing computational artifacts.
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)		
6th Grade	**7th Grade**	**8th 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	RST.6-8.3 Follow precisely a multistep procedure when carrying out experiments, taking measurements, or performing technical tasks.
L.6.6 Acquire and use accurately grade-appropriate general academic and domain-specific words and phrases; gather vocabulary	L.7.6 Acquire and use accurately grade-appropriate general academic and domain-specific words and phrases; gather vocabulary	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	knowledge when considering a word or phrase important to comprehension or expression	knowledge when considering a word or phrase important to comprehension or expression

Get Moving to Get Data

Grade 6-8 | 45 minutes | Beginner

Get Moving to Get Data

Students will investigate gathering, displaying, and using data.

Questions to investigate

- How can you gather and display data?

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.

KEY OBJECTIVES

Students will:

- Investigate ways to take in data from sensors
- Create a new program that will provide data using the force sensor

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.

1. Engage

Engage students in a conversation about transportation and ways that people get around.

Prompts students to share the different ways they get from one place to another. Have them think about what types of transportation they take regularly. Discuss the situations that require the use of different types of transportation.

Provide different examples of places to discuss what transportation types might be used in each place. Be sure to include walking, biking, different types of vehicles, boats, planes, etc.

Ask students to consider when they want to get to these places what is important information or data to have. Discuss ideas like knowing the distance, how long it will take, and how fast they need to move.

2. Explore

Students will investigate ways to take in data from sensors and manage their data.

Direct students to the **BUILD** section in the SPIKE App. Here students can access the building instructions for **Smart Kettlebell** 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.

Prompt students to think about how to use their sensor to take in information. Students should look at the upper left area of the programming canvas. Point out where the live sensor data is displayed. Ask students to move their model around and press the force sensor to see the live data stream. Discuss how moving the model provides different data. Have students indicate ways that they could program the distance sensor to provide data.

Ask students to review the three sample programs. Discuss which will be the best to gather and manage data in the console. Allow students to investigate the programs as needed.

Sample Program 1:

```
from hub import port
import runloop
import distance_sensor

async def main():
    #read the distance from the sensor
    dist_mm = distance_sensor.distance(port.F)
    print('mm:', dist_mm)

runloop.run(main())
```

Sample Program 2:

```
from hub import port
```

```
import runloop
import distance_sensor

async def main():
    #read the distance from the sensor
    while True:
        dist_mm = distance_sensor.distance(port.F)
        print('mm:', dist_mm)

runloop.run(main())
```

Sample Program 3:

```
from hub import port
import runloop
import distance_sensor
import force_sensor

async def main():
    #read the distance from the sensor
    while True:
        if force_sensor.pressed(port.B):
            dist_mm = distance_sensor.distance(port.F)
            print('mm:', dist_mm)
            await runloop.sleep_ms(1000)

runloop.run(main())
```

Allow students time to investigate the three programs. Please note that the distance sensor returns a value in millimeters, not centimeters. Students can convert from millimeters to centimeters by changing this line of code by adding a /10 at the end to divide by 10.

```
dist_cm = distance_sensor.distance(port.F)/10
```

```
from hub import port

import runloop

import distance_sensor

import force_sensor


async def main():
    #read the distance from the sensor

    while True:

        if force_sensor.pressed(port.B):

            dist_cm = distance_sensor.distance(port.F)/10

            print('cm:', dist_cm)

            await runloop.sleep_ms(1000)


runloop.run(main())
```

3. Explain

Discuss with students how each program worked. Ask students questions like:

- What did the data output in the console look like for each program?
- Which program would provide the best way to gather different data points?
- What are some ways that you could organize your data outside the console?

Prompt students to display their data in a table.

Remind students that there are different data types: integer, float, and string. Consider referencing the Knowledge Base under the distance sensor to help students decide what type of data is being used here. Discuss what type of data has been provided from their sensor.

4. Elaborate

Challenge students to create a new program that will provide data using the force sensor.

Ask students to look at the ways that they can get data from the force sensor. Discuss ideas for getting the number of Newtons from pressing the force sensor and displaying the data so students can easily gather and organize the information into a table. Consider ideas for displaying in the console or on the hub.

Allow students time to explore and develop their program. Encourage students to share their ideas and data with other groups.

5. Evaluate

Teacher Observation:

Discuss the program with students.

Ask students questions like:

- What is the difference between the three sample programs you investigated? What is the same?
- What happened when you were adding the force sensor?
- What are some ways you can display data to reference or capture to create a table?

Self-Assessment:

Have students answer the following in their journals:

- What did you learn today about gathering data using your robot?
- 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.

LEGO® Education SPIKE™ Prime

Bike Riding for Data

Grade 6-8 | 45 minutes | Beginner

Students will investigate using mathematical functions in a program.

Questions to investigate

- How can we use mathematical functions in our 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.

1. Engage

Engage students in a conversation about riding a bicycle as a mode of transportation.

Ask students to specifically talk about riding a bicycle to get from place to place, making sure to talk about the pros and cons of bicycle transportation.

Discuss places you might ride a bike, especially places that might make more sense to ride a bike than walk or take another type of transportation. Prompt students with examples if needed.

KEY OBJECTIVES

Students will:

- 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

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. Explore

Students will investigate riding a bicycle as a mode of transportation.

Direct students to the **BUILD** section in the SPIKE App. Here students can access the building instructions for the **Smart Bike** 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.

Prompt students to think about how they program their Smart Bike to move. Students should write a program that allows the bike to move forward at a constant speed.

Sample Program:

```
from hub import port

import runloop

import motor_pair


async def main():

    #pair motors

    motor_pair.pair(motor_pair.PAIR_1, port.C, port.E)

    #move 720 degrees, steering straight, at a velocity of 250

    await motor_pair.move_for_degrees(motor_pair.PAIR_1, 720, 0, velocity=250)


runloop.run(main())
```

Moving with Math
Once students have their bike moving, challenge them to add to their program to change the velocity of their bike to move faster and slower. However, they should not just add new movements to the program. Instead, students should use variables and math functions to change the velocity.

Introduce the following math functions that can be used. These should all be common and familiar to students. Remind students that the order of operations will be followed in complex lines of math code.

Function	Symbol	Sample	Output
Addition	+	print (10+2)	12
Subtraction	-	print (10-2)	8
Multiplication	*	print (10*2)	20
Division	/	print (10/2)	5

Using these math functions, students should be able to create a program that will increase and decrease the velocity of the bike.

Sample Program:

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

async def ride(bike_velocity):
    #defines the ride function to write the velocity on the hub and move the bike farward
    await light_matrix.write(str(bike_velocity))
    await motor_pair.move_for_degrees(motor_pair.PAIR_1, 720, 0, velocity=bike_velocity)

async def main():
    bike_velocity = 250
    #pair motors
    motor_pair.pair(motor_pair.PAIR_1, port.C, port.E)
    #move 720 degrees, steering straight, at a velocity of 250
    while True:

        #if the left button is pressed the velocity is reduced by 50
        if button.pressed(button.LEFT):
            bike_velocity -= 50
            await ride(bike_velocity)
            await runloop.sleep_ms(2000)

        #if the right button is pressed the velocity is increaded by 50
        if button.pressed(button.RIGHT):
            bike_velocity += 50
            await ride(bike_velocity)
            await runloop.sleep_ms(2000)

runloop.run(main())
```

Allow students time to explore the program and try different values to add and subtract. Note that in line 15 you find this code: await light_matrix.write(str(bike_velocity))

The str before the bike_velocity tells the program to cast the integer to a string to allow it to be written on the hub. Casting is a way to convert data from one type to another in a particular situation. In this program, it doesn't permanently change the data type, it converts it just to write on the hub. The variable is still an integer.

3. Explain

Discuss with students how the program worked.
Ask students questions like:

- How can we use math functions in our program?
- How did using the math functions change the way your program worked?
- What is the benefit of controlling the speed with a variable?

The function ride was created to allow the same code to be run in two different places in the program. Creating functions helps to keep the program length at a minimum and it also allows us to make our program more efficient.

4. Elaborate

Allow students to continue modifying their program to try new math functions based on what was shared from other students.

Challenge students to try adding new functions to their program to see how it changes the movements. Ask students to share their programs.

Sample program:

```
from hub import port, light_matrix, button

import runloop

import motor_pair


async def ride(bike_velocity):

    #defines the ride function to write the velocity on the hub and move the bike farward

    await light_matrix.write(str(bike_velocity))

    await motor_pair.move_for_degrees(motor_pair.PAIR_1, 720, 0, velocity=bike_velocity)


async def main():

    bike_velocity = 250
    #pair motors

    motor_pair.pair(motor_pair.PAIR_1, port.C, port.E)

    #move 720 degrees, steering straight, at a velocity of 250

    while True:

        #if the left button is pressed the velocity is reduced by 50

        if button.pressed(button.LEFT):

            bike_velocity /= 2

            await ride(int(bike_velocity))

            await runloop.sleep_ms(2000)


        #if the right button is pressed the velocity is increaded by 50

        if button.pressed(button.RIGHT):

            bike_velocity *= 2

            await ride(bike_velocity)

            await runloop.sleep_ms(2000)


runloop.run(main())
```

Allow time for students to explore the program and use different math functions. Note: When a number is divided it automatically gets converted to a float, a decimal number. The velocity parameter must be an integer. This requires the casting of an integer for the velocity parameter.

5. Evaluate

Teacher Observation:

Discuss the program with students.

Ask students questions like:

- What happened when we added mathematical functions to our program?
- What did you learn about programming mathematical functions?
- What issue was discovered when division was used?
- Why was a motor pair used when there is only one motor moving the bicycle?
- What iterations can be made to our program?

Self-Assessment:

Have students answer the following in their journals:

- What did you learn today about using math functions 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.

LEGO® Education SPIKE™ Prime

Counting Your Steps

Grade 6-8 | 45 minutes | Beginner

Students will investigate using variables and data while counting their steps.

Questions to investigate

- How can you add mathematical calculations within 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.

1. Engage

Engage students in a conversation about walking and counting their steps. How many students have a device that can do that? How many students use the apps available to them?

Have small groups of students stand shoulder to shoulder and take 5-10 steps forward. Why do they end up at different locations? How would the applications deal with that? Is a step the same for everyone?

2. Explore

Students will investigate using a smart device to track their steps as they use walking as a mode of transportation.

KEY OBJECTIVES

Students will:

- Integrate mathematical calculations into their programs using variables
- Create a program that will measure footsteps

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 the **BUILD** section in the SPIKE App. Here students can access the building instructions for the **Pedometer** model. Students will only need to build the Pedometer and not the stand. 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.

Prompt students to think about how to determine how many steps they have taken. Ask students to brainstorm which sensor(s) could be used to determine the number of steps taken. Guide students to the motion sensor. Specifically, have the students think about the roll, pitch, and yaw of the hub.

Discuss the ways the model can move when attached to your body. The pitch of the hub is the hub's front side tilted forward or backward. (Like tips of the fingers of your outstretched hand tilting up or down.) The roll of the hub is hub's front side moving down to the right or down to the left. (Like your outstretched hand rolling so the little finger rolls down or up.) The yaw of the hub is the top of the hub moving to the right or left at an angle. (Like turning your outstretched hand so the fingers start pointing forward but move to point to the right or left.)

Students need to create a program that will measure their steps.

Sample Program:

```
from hub import light_matrix, motion_sensor, button
import runloop


async def main():

    step = 0

    # reset the yaw angle to 0

    motion_sensor.reset_yaw(0)

    await light_matrix.write('Walk!')


    while True:

        # get the tilt values

        tilt = motion_sensor.tilt_angles()
        # store the yaw value from the tilt tuple
        yaw = tilt[0]

        if yaw > 75:
            step += 1
        elif yaw <-75:
            step += 1
        elif button.pressed(button.LEFT):
            await light_matrix.write(str(step))
            break

runloop.run(main())
```

Allow students time to explore the program and try different values.

Introduce some new math functions to students that can be used when creating and comparing their data.

Function	**Symbol**
Less than	<
Less than or equal to	≤
Greater than	>
Greater than or equal to	≥
Equal to	==
Not equal	!=

Discuss how these new functions were used in the program to get your steps.

Refining your Program
When getting the yaw angle, the pedometer collected data multiple times per second. The program doesn't know that the angle is part of your stride. To help, we can add a variable used for the purpose of identifying when the stride so the yaw angle is only collected once which will add only one step for the students.

Sample Program:

```
from hub import light_matrix, motion_sensor, button
import runloop
import time

def time_since(some_time):
    return time.ticks_diff(time.ticks_ms(), some_time)

async def main():
    # define your variables
    step = 0
    true_step = False

    await light_matrix.write('Walk!')

    while True:
        # get the tilt values
        tilt = motion_sensor.tilt_angles()
        # store the yaw value from the tilt tuple
        yaw = tilt[0]

        if yaw > 75 and true_step is False:
            step += 1
            true_step = True

        elif yaw <-75 and true_step is True:
            step += 1
            true_step = False

        elif button.pressed(button.LEFT):
            await light_matrix.write(str(step))
            break
```

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

Allow students time to explore the program.

3. Explain

Discuss with students how the program worked.
Ask students questions like:

- What was used to count the steps?
- How accurate is your pedometer? Explain.
- Why do you think the pedometer counted too many steps?

4. Elaborate

Challenge students to add to the program mathematical computations that will calculate the speed. Remember that speed is measured in distance over time.

While testing a program it is good practice to print information into the console to verify the data is accurate and to make sure the program is running as planned.

Sample Program:

```
from hub import light_matrix, motion_sensor, button

import runloop

import time


def time_since(some_time):

    return time.ticks_diff(time.ticks_ms(), some_time)


async def main():

    # define your variables

    step = 0

    true_step = False
```

```
stride = 3

distance_walked = 0

time_walked = 0

speed = 0

start_time = 0

elasped = 0


await light_matrix.write('Walk!')


# start the timer

start_time = time.ticks_ms()


while True:

    # get the tilt values

    tilt = motion_sensor.tilt_angles()

    # store the yaw value from the tilt tuple

    yaw = tilt[0]


    if yaw > 75 and true_step is False:

        step += 1

        true_step = True

    elif yaw <-75 and true_step is True:

        step += 1

        true_step = False

    elif button.pressed(button.LEFT):

        await light_matrix.write(str(step))

        break

# stop timer

elasped = time_since(start_time)
```

```
    # make calculations
    distance_walked = step * stride
    time_walked = elasped/1000
    speed = distance_walked / time_walked

    # round the time walked and speed to two decimal places
    time_walked = str(round(time_walked, 2))
    speed = str(round(speed, 2))

    # output the data by casting the variables to a string to add the units
    print('You took ' + str(step) + ' steps')
    print('Your stride is ' + str(stride) + ' feet')
    print('You walked for ' + str(time_walked) + ' seconds')
    print('You walked ' + str(distance_walked) + ' feet')
    print('Your speed is ' + str(speed) + ' feet per second')

runloop.run(main())
```

5. Evaluate

Teacher Observation:

Discuss the program with students.

Ask students questions like:

- What happened when you added the true_step variable?
- How does the true_step variable keep the motion sensor from collecting
  a reading?
- What did you learn about programming calculations?
- What iterations can you make to improve your pedometer?

Self-Assessment:

Have students answer the following in their journals:

- What did you learn today about using calculations 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.

Make It Move

Grade 6-8 | **45 minutes** | **Beginner**

Make It Move

Students will investigate how an automobile-like model can move forward and change directions using mathematical functions.

Questions to investigate

- How can use mathematical functions to change directions?

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.

KEY OBJECTIVES

Students will:

- Program a driving base to move forward and change directions
- Create a program with mathematical functions

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.

1. Engage

Engage students in a conversation about how the driver causes an automobile to move.

Guide students in a discussion on the different ways a driver controls the movement of an automobile. For example, the driver can make the automobile move forward and in reverse. The driver can stop the vehicle. The driver can also turn left and right at a variety of degrees.

Show students a video of the driving base model they will work with in the explore moving to see how it can move similar to an automobile. The engage video from the Training Camp 1 lesson can be used. The video is available at https://education.lego.com/en-us/lessons/prime-competition-ready/training-camp-1-driving-around#ignite-a-discussion.

Discuss the video with the students.

2. Explore

Students will investigate an automobile-like model as a type of transportation.

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

Prompt students to think about how they program their driving base to move. Students should write a program that allows the vehicle to move forward, backward and turn by programming the motors. If students need inspiration for their program, suggest they look at the sample program from the Bike Riding for Data lesson. Similar to this program, students should include a method for adding (speeding up) and subtracting (slowing down) speed.

Remind students that when driving a vehicle like a car there is a speed limit restriction that must be followed. In addition to driving forward, challenge students to display the speed of their driving base on the hub of the model to act as a speedometer.

Sample Program:

```
from hub import port, light_matrix, button

import runloop

import motor_pair


async def main():

    driving_velocity = 250


    # pair motors
    motor_pair.pair(motor_pair.PAIR_1, port.C, port.D)

    # define the ride function
    async def ride():
        await light_matrix.write(str(driving_velocity))
        await motor_pair.move_for_degrees(motor_pair.PAIR_1, 720, 0, velocity =
driving_velocity)

    # set conditions for moving forward and setting velocity
    while True:
        if button.pressed(button.LEFT):
            driving_velocity -= 50
            await ride()
            await runloop.sleep_ms(3000)
        elif button.pressed(button.RIGHT):
            driving_velocity += 50
            await ride()
            await runloop.sleep_ms(3000)

runloop.run(main())
```

Allow students time to explore the program. Prompt them to continue pushing the right and/or left buttons while running the program to see how the driving base will continue to move adding or subtracting 50. Students will be able to see the new speed each time on the hub.

Driving Around
Since automobiles do not often drive in a straight line only, challenge students to modify their program to include moving in different directions. Encourage students to use their math functions to help set how they make turns.

Sample Program:

```
from hub import port, light_matrix, button
```

```
import runloop
importmotor_pair
import motor

async def main():
    # define the initial driving velocity
    driving_velocity = 250
    # defining the turn distance
    turning_left = -360
    turning_right = 360

    # pair motors
    motor_pair.pair(motor_pair.PAIR_1, port.C, port.D)

    #define the ride and turn functions
    async def ride():
        await light_matrix.write(str(driving_velocity))
        await motor_pair.move_for_degrees(motor_pair.PAIR_1, 720, 0, velocity = 
driving_velocity)

    async def turn_left():
        await light_matrix.write(str(turning_left/4))
        await motor.run_for_degrees(port.C, turning_left, 250)

    async def turn_right():
        await light_matrix.write(str(turning_right/4))
        await motor.run_for_degrees(port.D, turning_right, 250)

    # set conditions for moving forward and setting velocity
    while True:
        await ride()
        if button.pressed(button.LEFT):
            driving_velocity -= 50
            turning_left -= 360
            await turn_left()
            await runloop.sleep_ms(3000)

        elif button.pressed(button.RIGHT):
            driving_velocity += 50
            turning_right += 360
            await turn_right()
            await runloop.sleep_ms(3000)

runloop.run(main())
```

3. Explain

Discuss with students how the program worked. Ask students questions like:

- Why did the program allow you to continue to add or subtract speeds?
- How were you able to modify your program to include turning and moving in different directions?
- Why did the drive base continue to drive when the buttons were not pressed?

When displaying the turns a math function was used to convert the actual degrees of wheel rotation to how much the drive base turned. In this case, when the single motor rotates 360 degrees it makes a full wheel rotation which turns the drive base 90 degrees. Using math, you can determine the answer to 360/n=90. In this case n=4. Thus, you see this math function used in the line light_matrix.write(str(turning_left/4)) and light_matrix.write(str(turning_right/4)).

4. Elaborate

Challenge students to recognize bugs in their programs.

Show students each program below and its error message. Have students discuss what needs to change in each code to fix the bug or add the missing code. Students can type in the program or review together as a whole class. Consider making the changes as a class and ensuring the program runs correctly after each change.

Debug Activity 1:
Find the error in the program. Remind students sometimes that error is not found until the program is running and an action is taken to read that part of the program.

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

async def main():
    driving_velocity = 250

    # pair motors
    motor_pair.pair(motor_pair.PAIR_1, port.C, port.D)

    # define the ride function
    async def ride():
        await light_matrix.write(str(driving_velocity))
        await motor_pair.move_for_degrees(motor_pair.PAIR_1, 0, velocity = driving_velocity)

    # set conditions for moving forward and setting velocity
    while True:
        if button.pressed(button.LEFT):
            driving_velocity -= 50
            await ride()
            await runloop.sleep_ms(3000)

        elif button.pressed(button.RIGHT):
            driving_velocity += 50
            await ride()
            await runloop.sleep_ms(3000)

runloop.run(main())
```

Error Message:

```
Traceback (most recent call last):
File "drive base 1", line 27, in <module>
File "drive base 1", line 19, in main
File "drive base 1", line 13, in ride
TypeError: function missing 1 required positional arguments
```

The error message references several lines that are incorrect. However, it does not actually point to the one line that has the error. In line 13, the amount of degrees were removed that tell the driving base how far to move forward. This line of code should read `await motor_pair.move_for_degrees(motor_pair.PAIR_1, 720, 0, velocity = driving_velocity)`

When the button is pressed for the driving base to move, it references the function and does not have the degrees needed to know how to move forward.

Debug Activity 2:
Find the error in the program. Remind students sometimes that error is not found until the program is running and an action is taken to read that part of the program.

```
from hub import port, light_matrix, button

import runloop

import motor_pair
```

```
driving_velocity = 250

async def main():
    # pair motors
    motor_pair.pair(motor_pair.PAIR_1, port.C, port.D)

    # define the ride function
    async def ride():
        await light_matrix.write(str(driving_velocity))
        await motor_pair.move_for_degrees(motor_pair.PAIR_1, 720, 0, velocity =
driving_velocity)

    # set conditions for moving forward and setting velocity
    while True:
        if button.pressed(button.LEFT):
            driving_velocity -= 50
            await ride()
            await runloop.sleep_ms(3000)

        elif button.pressed(button.RIGHT):
            driving_velocity += 50
            await ride()
            await runloop.sleep_ms(3000)

runloop.run(main())
```

Error Message:

```
Traceback (most recent call last):
File "drive base 1", line 29, in <module>
File "drive base 1", line 25, in main
```

```
NameError: local variable referenced before assignment
```

The error message will not appear until students press one of the buttons. Students should recognize that the error is occurring in line 20. The error message indicates that local variable is referenced before the assignment. This is a typo in the program where the program is trying to reference a variable that is set in the wrong place. The variable should be defined in the main function.

5. Evaluate

Teacher Observation:

Discuss the program with students.

Ask students questions like:

- How were you able to use math functions to change how the driving base moved both in straight lines and when turning?
- What errors can occur when creating complex programs? What are strategies to debug these errors?
- How could you use the drive base to turn instead of a single motor?

Self-Assessment:

Have students answer the following in their journals:

- What did you learn today about using multiple math functions 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.

Parking Lot

Grade 6-8 | 90 minutes | Advanced

Parking Lot

Students will investigate programming a vehicle to park autonomously.

Questions to investigate

- How can math functions support the creation of autonomous vehicles?

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.
- Parking lot – printed or taped on floor
- Student journals

Prepare

- Ensure SPIKE Prime hubs are charged, especially if connecting through Bluetooth.
- Ensure students have the driving base model built previously used in the Make It Move Lesson.

KEY OBJECTIVES
Students will:

- Use conditional statements in a program using sensors and motors.
- Apply sensors to real-life problems.

STANDARDS
CSTA
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.

1. Engage

Engage students in a discussion about autonomous vehicles focusing on self-parking.

Show videos or images of vehicles that are autonomous parking. Then ignite a discussion with the students about how these vehicles work and how they are currently being used.

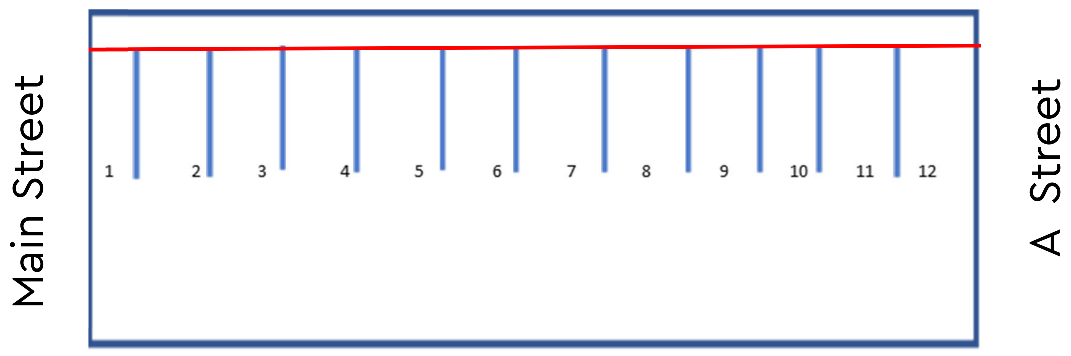
Focus on the example of cars that can park themselves. There are several models of cars that can parallel park themselves. Discuss how sensors play a role in how these vehicles work.

2. Explore

Students will plan and program their driving base to park without assistance.

Provide each team with an assigned parking space (1-12) and an assigned street to enter the parking lot from.

Show students a picture of the parking lot. You may use this example or use one taped on the floor. All students will need to see the parking lot for a class discussion, so it may be easier to show a picture on an overhead projector.



Review the rules with students:

- Both cars must enter at the same time into the parking lot
- Neither car can stop except in a parking space
- No one can touch the vehicle once it enters the parking lot.
- Cars can only back up when they are backing out of a space
- No part of the car can move over any of the lines

Discuss as a class how both cars be parked safely. Consider what happens if the vehicles need to move past each other. Go through each pair of cars entering at the same time.

Parking Challenge

Students will program their vehicles to autonomous park in their assigned space without colliding into another vehicle.

Direct students to add a distance sensor to their driving base in a position that the sensor is pointing down. Students can use the build section of the app for inspiration if needed.

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 time to program their vehicles to park. Provide students with the following elements that must be included in their program:

- The distance of the entire parking lot must be used
- More than one math function must be used
- The distance sensor must be used

Ask students to program their robots and work together as two teams to perfect the parking of the autonomous vehicles.

3. Explain

Discuss with students how the program worked. Ask students questions like:

- Ask students to explain how they determined how to handle the movement into the correct parking spaces. Was it easier to park in an empty parking lot or a full one? Did it matter?
- What issues were the most difficult to overcome?
- What was easy about the challenge?

4. Elaborate

Now that all the robots are in the parking lot, it's time for them all to leave. The class has a choice to make, they can choose one of the following scenarios to empty the parking lot:

A. All robots exit to the street from which they entered in the order they entered.
B. All robots exit to the street opposite from where they entered in the opposite order in which they entered.
C. All robots exit at the same time to Main Street in a synchronous movement.
D. All robots in odd numbered spaced exit to Main Street in a synchronous movement followed by all robots in even numbered spaces exiting to A Street in a synchronous movement.

Have the class discuss and choose a scenario. Then, write pseudocode and program the robot. When everyone is ready. Set up the parking lot and have all robots exit the parking lot.

5. Evaluate

Teacher Observation:

Discuss the program with students.

Ask students questions like:

- How were you able to come up with multiple ways to solve a problem?
- How were you able to use a sensor as part of your autonomous vehicle?
- How were you able to incorporate math functions in your program?

Self-Assessment:

Have students answer the following in their journals:

- What did you learn today about using multiple math functions 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.

LEGO® Education SPIKE™ Prime

My Transportation

Grade 6-8 | 90 minutes | Advanced

My Transportation

Students will create and program their own method of transportation to get them to school from their house.

Questions to investigate

- How can a vehicle be designed with programming math functions in mind?

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 how students get to school from their homes.

Ask students what the path to school looks like for them each day. Consider putting students into small groups based on location to discuss. Students might also want to create a map or share images with each other when describing the path.

As a whole group discuss what students' travel to school looks like. Do students need to travel across bodies of water or across a bridge? Over mountains or through a city? Ask students what other examples they can share.

KEY OBJECTIVES

Students will:

- Design, build, and program a transportation vehicle to bring them to school

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.

Brainstorm
Brainstorm different ideas for a vehicle that could be used to travel to school across all the different types of terrain.

In their small groups, brainstorm several ideas for creating a new type of vehicle to travel to school in that will easily travel over the needed terrain. Each group should come up with unique ideas that support their specific travel needs.

To further their ideas, ask students to work in their smaller groups to create a short survey to ask other students what they would like in a transportation vehicle to get to school. Students can share examples they are thinking about using and ask for other ideas. Students should include 3 questions about what would be effective, fun, and necessary to get to school safely.

Allow students time to create and complete their survey. Students can add additional ideas to their brainstorming list from the survey results.

2. Explore

Students will design, build, and program a transportation vehicle to bring them to school.

Design and Choose the Right Idea
Students should consider the input from their surveys and start to design their vehicle.

Students will design, build, and program a transportation vehicle that could be used to bring them to school. The constraints 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 math function in the program
5. It must work for at least two types of terrain

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 ideas as they go, making improvements where needed. Students should test and evaluate their designs against the design criteria set and their flowcharts as they start making their solutions.

Ensure students use sketches and photos of their models to record in their design journey.

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 transportation vehicle that will work for at least two types of terrain? Ask students to share their program comments to explain.
- What decisions did you have to make while creating your design?
- What type of math functions 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 get from the sharing session.

Allow students to share out their final designs and explain how they met the criteria.

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 math functions did you include and why?

Self-Assessment:

Have students answer the following in their journals:

- What did you learn today about creating your own design based on given criteria?
- 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.

LEGO® Education SPIKE™ Prime

Ideas to Help with My Transportation

Grade 6-8 | 30-45 min. | Intermediate

Ideas to Help with My Transportation
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 My Transportation lesson

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

Prepare
- Ensure SPIKE Prime hubs are charged, especially if connecting through Bluetooth.
- Ensure students have their built model from the My Transportation 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.

2. Explore

Have students work together to provide feedback to each other about the My Transportation 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,
    a. Remind students to be kind and clear in explaining why it is not clear or could be improved.
    b. Let the team receiving the feedback ask questions as needed for more clarity.
    c. 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.

Lists

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 stretching their mind with games and muscles with movements. Students will investigate how to create and use lists in their programs. Students will use sensors to gather data and then organize the data into lists. Students will learn how to make lists as well as how to modify lists. The lessons are designed in an order that allows students to progress in their skills and knowledge in the following areas:

- Create and utilize lists
- Code with compound conditionals using lists
- Create and compare two lists within one program
- Locate a given position or index value within a list
- Explore ways to generate data to include in lists
- Modify programs to add two lists together to make one list

Unit Learning Promise

In this unit, students will explore how to create and utilize lists in their programs that allow them to work their mind and muscles through stretches and games. Students will experience how to effectively utilize, compare, and combine lists with an emphasis on the importance of naming variables and lists to easily tell them apart as they program. Students will utilize sensors to assist them in gathering data for their lists as they explore different types of movement and mind games in this unit.

Investigation Questions:
How are lists used in programs? What information can be used from lists?
How can information be generated from sensors to use in lists?

Unit Lessons

Lesson 1	Lesson 2	Lesson 3	Lesson 4	Lesson 5	Lesson 6
Listing Letters Time: 45 min.	**Stretch Your Muscles and Lists** Time: 90 min.	**Mind Games** Time: 90 min.	**Jumping for Lists** Time: 90 min.	**Word Games with Lists** Time: 90 min.	**Ideas to Help with Word Games with Lists** 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-01 Recommend improvements to the design of computing devices, based on an analysis of how users interact with the devices.
2-CS-02 Design projects that combine hardware and software components to collect and exchange data.
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-11 Create clearly named variables that represent different data types and perform operations on their values.
2-AP-12 Design and iteratively develop programs that combine control structures, including nested loops and compound conditionals.
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-18 Distribute tasks and maintain a project timeline when collaboratively developing computational artifacts.
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)		
6th Grade	**7th Grade**	**8th 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	RST.6-8.3 Follow precisely a multistep procedure when carrying out experiments, taking measurements, or performing technical tasks.

L.6.6 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	L.7.6 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	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

LEGO® Education SPIKE™ Prime

Listing Letters

Grade 6-8 | 45 minutes | Beginner

Listing Letters

Students will investigate how to create lists through an unplugged activity then transfer this knowledge to creating a list and displaying letters on the hub.

Questions to investigate

Why would we need to provide more than one value for a variable?

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 lists.

Spark a discussion with students about ways that they use lists or see others use lists. Ask students to list types of lists (such as a grocery list, a to-do list, a roster or list of players).

Ask students to think about:

- Why do we use lists?
- What does a list provide us?
- How do we reference items on a list?
- How can we organize lists?

KEY OBJECTIVES

Students will:

- Create and utilize lists.
- Code with compound conditionals using lists.

STANDARDS

CSTA

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.

VOCABULARY

List

As time allows, challenge students to create different types of lists and then share the lists seeing if others have items or elements to add to the list. Types of lists could include: types of fruit or food, colors, types of places to live, types of music, etc.

2. Explore

Students will investigate how to create lists through an unplugged activity.

Creating Lists: Unplugged

Direct students to take all the 2x4 bricks from the set. Each person will have 5 bricks (yellow, green, violet, red, and blue) and needs to have 5 small pieces of paper or sticky notes with one number from 1-5 written on each.

Ask students to list or name each color that they have represented by the 5 bricks from their set. Students should each identify yellow, green, violet, red, and blue as their colors.

Explain to students that these are the values of their **list**. A list is a variable that can store multiple values. To program a list in python, students will need to provide
a name to the list, list the items or **elements** included, and then be aware of the position or **index value** of each element in the list.

Ask students to arrange their five bricks on one of the numbered pieces of paper or sticky notes. There is no particular order that they have to follow. Ask each person to write out the full list in the order they have chosen including the numbered position or index value of each brick in the list.

Round 1

Working together, have students play a game to create a stack of bricks using the different indexes in their lists.

Each partner will randomly select a number from a bag, bowl, or similar object holding the slips of paper or sticky notes from their own list. The number, representing the index in the list, will provide the order in which to stack their bricks. Each student should have a stack of 5 bricks at the end.

After the partners create their stacks, have them compare the color patterns. Ask students if the color patterns are the same or different. Encourage them to discuss why.

Round 2

Ask students to now place both sets of paper or sticky notes into the bag, bowl or similar object. Now they will work together to create a stack of 5 bricks using both sets of papers, which will allow for repeated items. Have the students select 1 of the two lists to follow and then repeat the stacking game.

Once students have finished, have them examine their new pattern. Ask students to identify if any colors are repeated. Discuss why this happened.

Programming Lists: Listing Letters
Students will program the hub to display a random letter from the alphabet.

Direct students to obtain the hub from their SPIKE Prime set. Then ask 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.

Discuss with students how to create a list in python. Show students the sample code to support your discussion. Ask students to type this code into their programming canvas and run the program.

Sample code:

```
from hub import light_matrix, button

import runloop


async def main():

    #define your list

    letters = ['A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I', 'J', 'K', 'L', 'M', 'N', 'O', 'P', 'Q', 'R', 'S', 'T', 'U', 'V', 'W',
'X', 'Y', 'Z']


    #write the fourth index in the list

    await light_matrix.write(letters[4])


runloop.run(main())
```

Students should see the letter D printed on their hub. Discuss the program together. Share with students the [4] is the index. In Python, the first item is a list is in index 0, which makes index 4 the fifth item in the list.

3. Explain

Discuss with students how their lists and program worked.
Ask students questions like:

- How did we create a list during our unplugged activity? How did we use the list to create our stacks?
- How can we program a list in python?
- What determined the letter shown on the hub?
- How could you make a random letter appear on the hub?

Review lists and their components again with students to ensure they understand what is needed to create a list.

4. Elaborate

Challenge students to a game of Brick-tionary using their lists to randomly generate a letter that their item need to begin with.

Each group will need to modify their code to allow for a random letter to be generated when the left button is pushed. Introduce the random.choice line of code to students and discuss how this is generating a random letter for them.

Sample code:

```
 from hub import light_matrix, button

import runloop

import random


async def main():

    #define your list

    letters = ['A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I', 'J', 'K', 'L', 'M', 'N', 'O', 'P', 'Q', 'R', 'S', 'T', 'U', 'V', 'W',
'X', 'Y', 'Z']


    while True:

        if button.pressed(button.LEFT):

            await runloop.sleep_ms(500)

            #write the fourth index in the list
        await light_matrix.write(random.choice(letters))

runloop.run(main())
```

Working in groups of 4, allow students to play several rounds of the game. Each student will take a turn playing the Brain Game Master, whose job is to run the program to randomly generate a letter. Each of the other students will then have 2 minutes to build an item that starts with that letter using bricks from their sets. The Brain Game Master will then try to guess what each item is at the end.

5. Evaluate

Teacher Observation:
Discuss the program with students.
Ask students questions like:

- What happened when we created the random letter generator?
- How can lists be used?
- What is important to remember when creating lists in Python?

Self-Assessment:
Have students answer the following in their journals:

- What did you learn today about using lists?
- 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.

LEGO® Education SPIKE™ Prime

Stretch Your Muscles and Lists

Grade 6-8 | 90 minutes | Intermediate

Stretch Your Muscles and Lists

Students will investigate using lists to determine the moves they will make.

Questions to investigate

How can a physical fitness trainer create a list of movements?

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.

KEY OBJECTIVES

Students will:

- Create a program using values for the motion sensor as the variables in their list
- Use a list to create a yoga routine

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-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.

1. Engage

Engage students in a conversation about stretching.

Discuss different ways that we can stretch our muscles. Consider showing images or video clips of people stretching to help students visualize what stretching looks like.

Prompt students to get up and move in order to stretch their muscles. Play a follow-the-leader type game where students have to match your movements exactly as you either demonstrate from the front or move around the room. Consider selecting a student to act as the leader and then passing the role to another student.

Now that students are all stretched out, have them think about ways that they moved their body. Discuss ways to create lists of the movements or muscles used and how they might be categorized.

2. Explore

Students will investigate different ways to move and stretch their muscles.

Direct students to the **BUILD** section in the SPIKE App. Here students can access the building instructions for the **Yoga Ring** 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.

Work with students to create a program that uses the values for the motion sensor as the variables in their list. Direct students to find **motion sensor** in the knowledge base. Under **up_face**, students can reference the values that can be used for the motion sensors orientation. Allow students time to try their program in with different options from the list.

Sample Program:

```
from hub import motion_sensor

import runloop


async def main():

  # Create a list with the hub's orientation

  yoga_moves = [motion_sensor.FRONT, motion_sensor.BACK, motion_sensor.LEFT,
motion_sensor.RIGHT]


  orientation = motion_sensor.up_face()


  while True:

    await runloop.until(lambda: motion_sensor.up_face() != orientation)

    orientation = motion_sensor.up_face()
```

```
        if orientation == yoga_moves[3]:

            print('Good Job!')

            await runloop.sleep_ms(1000)

runloop.run(main())
```

Note: 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.ROGHT, the light matirx is motion_sensor.TOP, the battery side is motion_sensor.BOTTOM.

Discuss the program with students. Ask students to identify what each position or index value is in the list. For example, the motion_sensor.LEFT is index value 3 and motion_sensor.BAXK is index value 1. Explain to students that the index values always start with 0 in Python.

Create a Yoga Routine
Challenge students to create a yoga routine by creating a list of the options and then printing the moves in order chosen in the console.

Ask students to modify their programs to include several moves from their list. They should print the list of moves in the console so that another team could follow their yoga routine. Allow students time to practice their moves.

Sample Program:

```
from hub import motion_sensor, light_matrix

import runloop


async def main():

    # Create a list with the hub's orientation

    yoga_moves = [motion_sensor.FRONT, motion_sensor.BACK, motion_sensor.LEFT,
motion_sensor.RIGHT]


    # Follow this routine

    print('LEFT')

    print('RIGHT')
    print('BACK')

    print('FRONT')

    print('READY GO!')

    orientation = motion_sensor.up_face()


    while True:

        await runloop.until(lambda: motion_sensor.up_face() != orientation)

        orientation = motion_sensor.up_face()

        if orientation == yoga_moves[2]:

            print('Left Side')

            light_matrix.show_image(light_matrix.IMAGE_ARROW_W)

        elif orientation == yoga_moves[3]:

            print('Right Side')

            light_matrix.show_image(light_matrix.IMAGE_ARROW_E)

        elif orientation == yoga_moves[1]:

            print('Back')

            light_matrix.show_image(light_matrix.IMAGE_ARROW_N)

        elif orientation == yoga_moves[0]:

            print('Front')

            light_matrix.show_image(light_matrix.IMAGE_ARROW_S)

runloop.run(main())
```

Encourage students to mix up the moves and have several steps in their routine.

Get Random with Your Moves
After students practice their routine, challenge them to add a random choice line of code to change the routine with an unexpected move.

Sample Program:

```
from hub import motion_sensor, light_matrix
```

```
import runloop

import random


async def main():

    # Create a list with the hub's orientation

    yoga_moves = [motion_sensor.FRONT, motion_sensor.BACK, motion_sensor.LEFT,
motion_sensor.RIGHT]

    random_move = random.choice(yoga_moves)


    # Follow this routine

    print('LEFT')

    print('RIGHT')

    print(random_move)

    print('BACK')

    print('FRONT')

    print(random_move)

    print('READY GO!')


    orientation = motion_sensor.up_face()



    while True:

        await runloop.until(lambda: motion_sensor.up_face() != orientation)

        orientation = motion_sensor.up_face()



        if orientation == yoga_moves[2]:

            print('Left Side')

            light_matrix.show_image(light_matrix.IMAGE_ARROW_W)

        elif orientation == yoga_moves[3]:

            print('Right Side')

            light_matrix.show_image(light_matrix.IMAGE_ARROW_E)
        elif orientation == random_move:
            print('Random')
            light_matrix.show_image(light_matrix.IMAGE_SILLY)
        elif orientation == yoga_moves[1]:
            print('Back')
            light_matrix.show_image(light_matrix.IMAGE_ARROW_N)
        elif orientation == yoga_moves[0]:
            print('Front')
            light_matrix.show_image(light_matrix.IMAGE_ARROW_S)

runloop.run(main())
```

Allow time for students to try their programs and play with different ways to include the random choice line of code.

3. Explain

Ask students to demonstrate their routine and discuss with students how the program worked. Ask students questions like:

- How did you use a list to create your yoga routine?
- What happened when you added random moves to your routine?
- What are some ideas for how you could change your program to create a new routine?

4. Elaborate

Challenge students to stretch this list and their routine to add some new moves.

Ask students to modify their program to include some new moves. First, students will need to add two new options to their list that allows for the motion sensor to be oriented in the top and bottom position. However, students need to insert these new moves without changing the current list programed (i.e. they cannot just add the two elements to the end of the existing list).

Introduce the **insert** line of code **list_name.insert(index location, value)** to the students and then ask them to add the sample program below to the end of their current program. Students should run the program to see how the newly inserted lines work.

Sample Program:

```
from hub import motion_sensor, light_matrix

import runloop

async def main():

    # Create a list with the hub's orientation

    yoga_moves = [motion_sensor.FRONT, motion_sensor.BACK, motion_sensor.LEFT,
motion_sensor.RIGHT]


    yoga_moves.insert(0, motion_sensor.TOP)

    yoga_moves.insert(1, motion_sensor.BOTTOM)


    orientation = motion_sensor.up_face()


    while True:

        await runloop.until(lambda: motion_sensor.up_face() != orientation)

                orientation = motion_sensor.up_face()


        if orientation == yoga_moves[0]:

            print('Top')

            light_matrix.show_image(light_matrix.IMAGE_TRIANGLE)

        elif orientation == yoga_moves[1]:

            print('Bottom')

            light_matrix.show_image(light_matrix.IMAGE_SQUARE)

runloop.run(main())
```

Point out to students where the new value is inserted into the list.

After practicing their own new yoga routine, allow groups to challenge each other in preforming their yoga routines.

Consider also sharing with students how to delete an element from their list using the **del listname[index value]** line of code, which in this example might look like **del yoga_moves[1]** to remove 'motion_sensor.BOTTOM' from the list.

```
        yoga_moves = [motion_sensor.FRONT, motion_sensor.BACK,
motion_sensor.LEFT, motion_sensor.RIGHT]

        print(yoga_moves)

        del yoga_moves[1]

        print yoga_moves
```

5. Evaluate

Teacher Observation:
Discuss the program with students.
Ask students questions like:

- How are you able to add new elements to or delete elements from your list?
- How were you able to create your yoga plan using your list?
- How were you able to use lists and sensors together?

Self-Assessment:
Have students answer the following in their journals:

- What did you learn today about using lists to create a plan, like a yoga routine, to follow?
- 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.

Mind Games

Grade 6-8 | 90 minutes | Intermediate

Mind Games

Students think outside the box to discover how to use multiple lists and compare lists.

Question to investigate

- How can indexing compare two lists for common components?

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 stretching their mind to think about how different lists are similar.

KEY OBJECTIVES

Students will:

- Create two lists in one program
- Compare two lists within the program

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.

Ask students to remove all 10 of the 2x4 colored bricks in their set. Ask each student in the group to take 1 of each color including red, magenta, yellow, blue, and green.

Partner 1 should create a stack of their 5 bricks in any order without showing Partner 2. Ask Partner 1 to describe the build to Partner 2 in order to create the same stack (i.e. show the build, tell the order without showing, etc.). Once the pair thinks the stacks are the same, they are finished.

Ask the partners to switch and do the exercise again. This time, the partners have to come up with a new way to communicate about the stack (i.e. different than their first approach which could be describing in a different way, not using words, etc.).

After groups complete several rounds, discuss together as a class how the game is played and ask students to write a pseudocode program for this game.

2. Explore

Students will investigate programs that include multiple lists and how to locate a given position or index value in the list.

Direct students to the **BUILD** section in the SPIKE App. Here students can access the building instructions for **Game Master** 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.

Students will use the Game Master model to read the colors of the candy stick bricks. Provide students with the sample program below. Review the program together to discuss what the program does. Ask students to run the program and then feed one of the candy sticks into the model.

Sample Program:

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

import motor

import color_sensor

import color

from app import sound


def feed_candy_1():

    return button.pressed(button.LEFT)


def feed_candy_2():
    return button.pressed(button.RIGHT)

async def main():
    # create lists
    candy1 = []
    candy2 = []

    # this makes the Game Master eat candy stick 1
    await runloop.until(feed_candy_1)
    light_matrix.clear()
    candy1.clear()
    await motor.run_for_time(port.A, 2000, -500)
    await sound.play('Bite')
    await sound.play('Bite')

    # this will read and record its sequence of colors in the list called candy 1
    for x in range(5):
        candy1.append(color_sensor.color(port.B))
        await runloop.sleep_ms(1000)
        await motor.run_for_degrees(port.A, 95, 500)

    # this makes the Game Master eat candy stick 2
    await runloop.until(feed_candy_2)
    light_matrix.clear()
    candy2.clear()
    await motor.run_for_time(port.A, 2000, -500)
    await sound.play('Bite')
    await sound.play('Bite')

    # this will read and record its sequence of colors in the list called candy 2
    for x in range(5):
        candy2.append(color_sensor.color(port.B))
        await runloop.sleep_ms(1000)
        await motor.run_for_degrees(port.A, 95, 500)

    # light up the position of red bricks if it is in the same position in both candy sticks
    candy1_red_index = -1
    If color.RED in candy1:
        candy1.index = candy1.index(color.RED)
    candy2_red_index = -1
    If color.RED in candy2:        candy2_red_index = candy2.index(Color.RED)

    for x in range(5):
        print(candy1[x])

    if candy1_red_index == candy2_red_index:
        for x in range(5):
            light_matrix.set_pixel(x, candy1_red_index, 100)
            await sound.play('Win')
    else:
        light_matrix.show_image(light_matrix.IMAGE_NO)
        await sound.play('Oops')

runloop.run(main())
```

Allow students time to investigate the program.

3. Explain

Discuss with students how the program worked.
Ask students questions like:

- How did the program work?
- How is this program using lists?
- How does this program index and compare the two lists?

4. Elaborate

Challenge students to create a new program that indexes and compares all of the colors to check for identical block stacks.

Discuss with students how to modify their program in order to check for the other four colors.

1. Need to index the other colors
2. Need to compare the lists for the other colors
3. Need to set the pixels for the other colors if both are in identical location on the stack.

Sample Program:

```
from hub import light_matrix, button, port

import runloop

import motor

import color_sensor

from app import sound


def feed_candy_1():

    return button.pressed(button.LEFT)


def feed_candy_2():

    return button.pressed(button.RIGHT)



async def main():

    # create lists

    candy1 = []

    candy2 = []
```

```
# this makes the Game Master eat candy stick 1

await runloop.until(feed_candy_1)

light_matrix.clear()

candy1.clear()

await motor.run_for_time(port.A, 2000, -500)

await sound.play('Bite')

await sound.play('Bite')

# this will read and record its sequence of colors in the list called candy 1

for x in range(5):

    candy1.append(color_sensor.color(port.B))

    await runloop.sleep_ms(1000)

    await motor.run_for_degrees(port.A, 95, 500)

# this makes the Game Master eat candy stick 2

await runloop.until(feed_candy_2)

light_matrix.clear()

candy2.clear()

await motor.run_for_time(port.A, 2000, -500)

await sound.play('Bite')

await sound.play('Bite')

# this will read and record its sequence of colors in the list called candy 2

for x in range(5):

    candy2.append(color_sensor.color(port.B))

    await runloop.sleep_ms(1000)

    await motor.run_for_degrees(port.A, 95, 500)

# light up the position of red bricks if it is in the same postion in both candy sticks

# the number correlates to the color
```

```
    candy1_red_index = candy1.index(color.RED) if color.RED in candy1 else -1

    candy2_red_index = candy2.index(color.RED) if color.RED in candy1 else -1

    candy1_yellow_index = candy1.index(color.YELLOW) if color.YELLOW in candy1 else -1

    candy2_yellow_index = candy2.index(color.YELLOW) if color.YELLOW in candy1 else -1

    candy1_green_index = candy1.index(color.GREEN) if color.GREEN in candy1 else -1

    candy2_green_index = candy2.index(color.GREEN) if color.GREEN in candy1 else -1

    candy1_magenta_index = candy1.index(color.MAGENTA) if color.MAGENTA in candy1 else -1

    candy2_magenta_index = candy2.index(color.MAGENTA) if color.MAGENTA in candy1 else -
1

    candy1_blue_index = candy1.index(color.BLUE) if color.BLUE in candy1 else -1

    candy2_blue_index = candy2.index(color.BLUE) if color.BLUE in candy1 else -1


    for x in range(5):

        print(candy1[x])

    if candy1_red_index == candy2_red_index:

        for x in range(5):

            light_matrix.set_pixel(x, candy1_red_index, 100)

            await sound.play('Win')


    if candy1_yellow_index == candy2_yellow_index:

        for x in range(5):

            light_matrix.set_pixel(x, candy1_yellow_index, 100)

            await sound.play('Win')


    if candy1_green_index == candy2_green_index:

        for x in range(5):

            light_matrix.set_pixel(x, candy1_green_index, 100)

            await sound.play('Win')


    if candy1_magenta_index == candy2_magenta_index:
        for x in range(5):

            light_matrix.set_pixel(x, candy1_magenta_index, 100)

            await sound.play('Win')


    if candy1_blue_index == candy2_blue_index:

        for x in range(5):

            light_matrix.set_pixel(x, candy1_blue_index, 100)

            await sound.play('Win')


    if candy1_red_index == candy2_red_index and candy1_yellow_index ==
candy2_yellow_index and candy1_green_index == candy2_green_index and
candy1_magenta_index == candy2_magenta_index and candy1_blue_index ==
candy2_blue_index:

        await sound.play('Triumph')


runloop.run(main())
```

Allow students time to investigate the program.

5. Evaluate

Teacher Observation:

Discuss the program with students.

Ask students questions like:

- What happened when you elaborated on the original program?
- How did the program index and compare lists?

Self-Assessment:

Have students answer the following in their journals:

- What did you learn today about using indexing to compare lists?
- 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.

LEGO® Education SPIKE™ Prime

Jumping for Lists

Grade 6-8 | **90 minutes** | **Intermediate**

Jumping for Lists

Students will create a program using lists to capture data of physical performance.

Questions to investigate

- How can you create a list that represents your personal statistics for performance?

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 the motion of your body when you do a squat. (For example, your legs bend, and your torso gets closer to the ground.) Have students stand up and do a squat.

Next, have students jump in place straight up. Ask students what body movements were made doing the jump.

What are ways to measure how low a body gets during a squat or how high a body is at the apex of a jump?

KEY OBJECTIVES

Students will:

- 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)

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-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.

2. Explore

Students will investigate ways to generate data to include in lists.

Direct students to the **BUILD** section in the SPIKE App. Here students can access the building instructions for **Smart Kettlebell** 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.

Part 1: Create data for your list

Students should program their Kettlebell model to provide several heights for their jumps. Students will use this information to create a list to use later.

Review the sample program with students. Discuss the way that the force and distance sensors are used in this program. Students will use the force sensor to indicate when the jump happens and the distance sensor to measure the height of the jump. Ask students to run the program and see what values they get for their heights.

Sample Program:

```
from hub import port, light_matrix

import runloop

import distance_sensor

import force_sensor


def is_force_sensor_pressed():

    # collect input from force sensor

    return force_sensor.pressed(port.B)


def is_force_sensor_released():

    # collect input from force sensor

    return not force_sensor.pressed(port.B)


async def main():
    runloop.until(is_force_sensor_pressed)

    await light_matrix.write('JUMP!')


    runloop.until(is_force_sensor_released)

    dist_cm = distance_sensor.distance(port.F)/10

    print(str(dist_cm) + ' cm')


runloop.run(main())
```

Note: students should press the force sensor, hold it, jump, and then release the sensor when they are at the height of their jump. The distance sensor may not read properly when looking at some carpet floors. If the jump reads −0.1 cm, try a different location.

After students explore the program, ask each person on the team to complete three jumps. Students should record the values for each jump in a table.

Example Table:

	Person 1	Person 2
Trial 1		
Trial 2		
Trail 3		

Part 2: Create your list

Students will create two lists in a new program using the data they generated in part 1.

Discuss with students how to create a new program that will include a list for each student and the data from each of their jumping trials as the values in the list. Remind students of the importance of naming variables and lists to easily tell them apart as they program. Here students might use their own names for their individual lists.

Ask students to create their program including both lists, indicating what the maximum value from each list is and then sharing each value on the hub. Introduce the max(name1) line of code to students to indicate their max value in their list.

Sample program (Students should use their own names and data from their table):

```
from hub import port, light_matrix

import runloop

import force_sensor


def is_force_sensor_pressed():

    # collect input from force sensor

    return force_sensor.pressed(port.B)


async def main():

    #create a list for each person using the values from the table

    name1 = [trial1, trial2, trial3]

    name2 = [trial1, trial2, trial3]


    #print the maximum value from each person's list

    print('Name 1: ' + str(max(name1)))

    print('Name 2: ' + str(max(name2)))



    #display the max values on the hub when the force sensor is pressed

    await runloop.until(is_force_sensor_pressed)

    await light_matrix.write(str(max(name1)))

    await runloop.sleep_ms(2000)

    light_matrix.show_image(light_matrix.IMAGE_YES)

    await runloop.sleep_ms(1000)

    await light_matrix.write(str(max(name2)))

    await runloop.sleep_ms(2000)

    light_matrix.show_image(light_matrix.IMAGE_YES)


runloop.run(main())
```

Allow students time to investigate the program trying different ways to show the output or the max value for each list.

Part 3: Putting lists together

Students will modify their program to add their two lists together to make one list.

Explain to students that **concatenating** lists allows two lists to be combined into one using the + sign. Share with students the example line of code list3 = list1 + list 2 as a way to add lists together. Challenge students to concatenate their lists and find the new max or highest number from both lists.

Sample program:

```
from hub import port, light_matrix

import runloop

import force_sensor


def is_force_sensor_pressed():

    # collect input from force sensor

    return force_sensor.pressed(port.B)


async def main():

    #create a list for each person using the values from the table

    name1 = [trial1, trial2, trial3]

    name2 = [trial1, trial2, trial3]


    #print the maximum value from each person's list

    print('Name 1: ' + str(max(name1)))

    print('Name 2: ' + str(max(name2)))



    #combine the lists when the force sensor is pressed

    #display the sum of the new list
    await runloop.until(is_force_sensor_pressed)

    combo_list = name1 + name2

    print(combo_list)

    await light_matrix.write(str(sum(combo_list)))

    print(sum(combo_list))

    light_matrix.show_image(light_matrix.IMAGE_YES)


runloop.run(main())
```

Allow students time to investigate the program trying different ways to show the output of the sum value for the combined list. Students might also try showing the sum of each list as well as the combined sum.

3. Explain

Discuss with students how the program worked.
Ask students questions like:

- How were you able to generate your own data to include in a list?
- How were you able to incorporate using the force sensor as a button to push for an action to happen?
- How were you able to find different information about your lists (such as sum or max)? What other values do you think you could find?
- How can you combine lists together? When might this be useful?

4. Elaborate

Challenge students to indicate if a certain number is included in their list.

Discuss with students how to look for information within their lists. While students can easily see the list in their program, there are times when you might need to investigate what has been included in a list. Discuss ways and reasons this could happen as a group.

Ask students to modify their program that has the combo list to identify if the number 20 is included in the combo list. Note: select an appropriate number according to students' data.

Share the sample program with students and have them identify how the program is checking the list to see if the number is included.

Sample program:

```python
from hub import port, light_matrix
import runloop
import force_sensor

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

async def main():
    #create a list for each person using the values from the table
    name1 = [trial1, trial2, trial3]
    name2 = [trial1, trial2, trial3]

    #print the maximum value from each person's list
    print('Name 1: ' + str(max(name1)))
    print('Name 2: ' + str(max(name2)))

    #combine the lists when the force sensor is pressed
    #display the sum of the new list
    await runloop.until(is_force_sensor_pressed)
    combo_list = name1 + name2
    print(combo_list)
    await light_matrix.write(str(sum(combo_list)))
    print(sum(combo_list))
    light_matrix.show_image(light_matrix.IMAGE_YES)
    await runloop.sleep_ms(2000)
    light_matrix.clear()

    if (20 in combo_list):
        print('The number 20 is in the list.')
        light_matrix.show_image(light_matrix.IMAGE_YES)
    else:
        print('The number 20 is not in the list')
        light_matrix.show_image(light_matrix.IMAGE_NO)

runloop.run(main())
```

Ask students to modify their program similar to the sample program and search for various values to see what is included or not in the program. Students can modify the program in additional ways to indicate if it is or is not included.

5. Evaluate

Teacher Observation:

Discuss the program with students.

Ask students questions like:

- What are some ways that you can see using multiple lists?
- What are some ways that lists can benefit you in a program?
- What are different types of information you can take from your lists?

Self-Assessment:

Have students answer the following in their journals:

- What did you learn today about using multiple lists?
- 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.

LEGO® Education SPIKE™ Prime

Word Games with Lists

Grade 6-8 | 90 minutes | Advanced

Word Games with Lists

Students will create lists in order to complete a story based on a word game.

Questions to investigate

- How can multiple lists be used and accessed at the same time?

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.

KEY OBJECTIVES

Students will:

- Create multiple lists within a program to complete a word game
- Program a color sensing model to coordinate with their word 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-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.

1. Engage

Engage students in a word game where they have to provide certain types of words that will later be used to finish a story.

Ask students to provide example words for the following examples:

- Place
- Adjective
- Tool or Machine
- Color
- Type of animal

Read the sample story to the class including some of the words they chose in place of the blanks.

> Kate and Kyle decided to go fishing one day while sitting around in the __(place)______. Kate really wanted to catch a ___(Adjective)___ fish. First, they needed to find a ___(tool or machine)____ to catch the fish with. Luckily, Kyle knows his neighbor has a __(color)_____ fishing net. They ask to borrow the net and go straight to the pond where instead of fish they see a ___(animal)___.

Consider reading the story a couple of times using different words for fun.

Challenge students to create their own story with at least 5 words that can become the words that need added in during the game. Be sure to set the expectation for how long the story should be, if it should be on a certain topic, or if certain types of words (noun, verb, adjective, etc.) should be the ones left blank.

2. Explore

Students will investigate using lists with sensors to create their own word game.

Direct students to the **BUILD** section in the SPIKE App. Here students can be inspired by the building instructions from several models in order to design their own color detector to use in completing their word game. Ask students to design and build their model.

Program your Story

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.

Challenge students to create a program where they can use lists of the word types like nouns, verbs, and adjectives to have word options to substitute in the blanks in their story. Remind students that the lists should match the types of words that need to be used in their individual stories which means everyone will not create the same lists.

Additionally, students will need to program their color sensing model to coordinate with their word game. Using five different colors, students should create a conditional statement that allows each different color sensed to provide a word for your story (i.e. blue is a noun, red is a verb, etc.).

Test and Iterate

Allow time for students to test and analyze their ideas as they go, making improvements where needed. Students should test and evaluate their designs against the design criteria set and their flowcharts as they start making their solutions.

Ensure students use sketches and photos of their models to record in their design journey.

Allow students to receive feedback on their designs as time allows. This can be from other groups or the teacher.

3. Explain

Discuss with students how the program worked.
Ask students questions like:

- How does your color sensing model work?
- How does your model interact with your story?
- What decisions did you have to make while programing your story?
- 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 get 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?
- How can sensors and lists work together?

Self-Assessment:

Have students answer the following in their journals:

- What did you learn today about creating lists and working with sensors?
- 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.

LEGO® Education SPIKE™ Prime

Ideas to Help with Word Games with Lists

Grade 6-8 | 30-45 min. | Intermediate

Ideas to Help with Word Games with Lists

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 Word Games with Lists lesson

Prepare

- Ensure SPIKE Prime hubs are charged, especially if connecting through Bluetooth.
- Ensure students have their built model from the Word Games with Lists 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 Word Games with Lists 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,
    a. Remind students to be kind and clear in explaining why it is not clear or could be improved.
    b. Let the team receiving the feedback ask questions as needed for more clarity.
    c. 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.