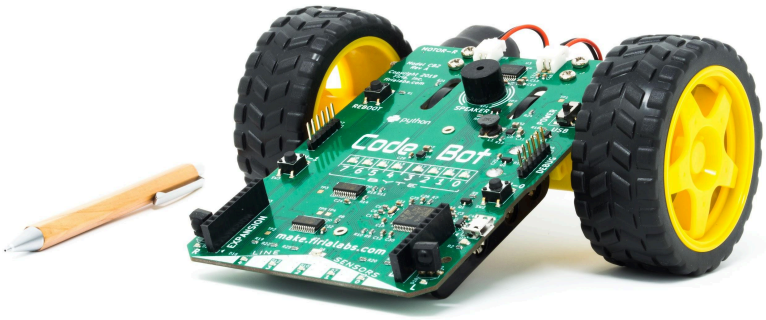
FIRIA LABS

Python with Robots

Middle School Curriculum Guide



Unit 3

Line Sensor Data

Unit 3: Line Sensor Data (8-12 hours)

Students expand CodeBot's abilities by utilizing data from the five line sensors mounted on the front of the 'bot. They learn about loops and defining and calling functions. Functions are defined with and without parameters, and also with and without a return. Line sensor data is used to control CodeBot, creating an autonomous robot.

Summary of Mission 5: This mission is divided into four lessons, each approximately one class period. During the mission students learn about line sensors and how to access the data. They also learn about using the console log to display data. The final program is an autonomous robot that moves inside a bounded area.

> **Lesson 1:** Line sensors are introduced. Students learn about analog and digital and why data must be converted. They use the console panel to display a line sensor reading for different surfaces.
>
> **Lesson 2:** Students define a function that reads a single LED and uses a parameter to determine which line sensor. Another function is defined that uses a loop to call the first function and read all the line sensors. A Boolean variable is used to turn on/off the corresponding LED.
>
> **Lesson 3:** Students add a "return" statement to each function so it can return a Boolean value that determines if a line was detected. A counter variable is incremented when a line is detected, and the number is displayed using the user LEDs. An if statement is added to reset the counter when it reaches its maximum value. Also, the "wait" safety feature used in Mission 4 is added to the program.
>
> **Lesson 4:** This lesson is only one objective. Students define functions for moving the robot and add the function calls in the main program. The Kahoot quiz reviews the entire mission.

Summary of Mission 6: This mission is divided into three lessons, each approximately one class period. During the mission students create a line following program using line sensor data for proportional control.

> **Lesson 1:** Students learn about using a list in code, and use line sensor data to turn on/off LEDs.
>
> **Lesson 2:** Students learn about using a tuple in code, and use the pre-coded ls.check() function to read line sensor data. A simple line follower program is created.
>
> **Lesson 2:** Tuples are used in conditions for proportional control. Also, students write a function for calibrating CodeBot.

Unit 3 Remix: The remix project can be an extension of a current program or a completely new project. Students can work individually with a partner, or in teams.

Unit 3 Classroom Materials that are provided with each lesson:

- **Lesson Plan:** A detailed lesson plan is provided for each lesson. It includes learning targets, success criteria, vocabulary and new code. It also has teaching tips for each objective.
- **Mission Slidedeck:** Each lesson includes PowerPoint slides for teacher-led instructions. You can use them to guide students through the material. In this unit, there are many times the slides should be used instead of the instructions in CodeSpace and CodeTrek. They introduce things a little differently, and often give more examples. All goals will be met, but the code in the slides looks a little different than the code in CodeTrek.
  Mission Log: Each lesson has an assignment, called a mission log, for students to complete as they go through the lesson. It includes warm-up and wrap-up questions as well as questions along the way for guided notes. An answer key for each mission log is provided.
- **Kahoot! Review:** Each lesson has a Kahoot! that reviews the concepts and codes.

Unit 3 Preparation:

- Have a computer / laptop with the Chrome web browser for each student.
- Make sure students can log into CodeSpace at http://make.firialabs.com with their email address.
- Have a CodeBot and USB cable for each student or programming pair.
- Four AA batteries are needed for each CodeBot.
- Print the Testing Surfaces paper for each student or programming pair.
- A driving surface is needed. Use white posterboard with black electrical tape.

Assessment:

Mission 5 Obj 1-2 Kahoot	Mission 5 Obj 3-5 Kahoot	Mission 5 Obj 6-8 Kahoot	Mission 5 Kahoot Review
Mission 6 Obj 1-3 Kahoot	Mission 6 Obj 4-6 Kahoot	Mission 6 Obj 7-8 Kahoot	

Coming soon: Unit 3 reviews and multiple choice exams are provided. They are divided into two parts: vocabulary and coding. The reviews are available as Kahoots, and the exams are available as MS Forms. All review and test questions are provided in the Unit 3 Question Bank.

Unit 3 Vocab Kahoot Review	Unit 3 Coding Kahoot Review	Unit 3 Vocab Test (MS Form)	Unit 3 Coding Test (MS Form)

Standards addressed in this unit:

CSTA Standards Grades 6-8	CSTA Standards Grades 9-10	CSTA Standards Grades 11-12
• 2-CS-01 • 2-CS-02 • 2-CS-03 • 2-DA-08 • 2-DA-09 • 2-AP-10 • 2-AP-11 • 2-AP-12 • 2-AP-13 • 2-AP-14 • 2-AP-15 • 2-AP-16 • 2-AP-17 • 2-AP-19	• 3A-CS-03 • 3A-DA-11 • 3A-AP-13 • 3A-AP-14 • 3A-AP-15 • 3A-AP-16 • 3A-AP-17 • 3A-AP-18 • 3A-AP-19 • 3A-AP-21 • 3A-AP-23	• 3B-CS-02 • 3B-DA-06 • 3B-AP-10 • 3B-AP-12 • 3B-AP-14 • 3B-AP-16 • 3B-AP-17 • 3B-AP-21 • 3B-AP-22 • 3B-AP-23

Basic Lesson Plans for each lesson included here.

For complete lesson plans, see each mission in the Learning Portal.

MISSION 5: Fence Patrol **Lesson 1 (Objectives 1-2)**	**Time Frame: 35-40 minutes**

Project Goal: Students will read and display data from CodeBot's line sensors

Learning Targets

- I can use ls.read() to get real-time line sensor values.
- I can print values in the console panel while debugging to get real-time sensor values.
- I can print values to the console panel while running a program to display real-time values.
- I can assign a Boolean value to a variable by using an if statement as part of the assignment statement.

Key Concepts

- Analog sensors are non-contact sensors used in many industrial and commercial applications.
- The console panel can be used to print real-time data using the print() statement.
- A dark line on a light background will have a larger value, toward the high end of the range. A light line on a dark background will have a smaller value, toward the low end of the range.
- A variable can be assigned a Boolean value by using an if statement as part of the assignment statement.

Assessment Opportunities

- Mission 5 Lesson 1 Log (digital)
  - Includes a chart for recording reflective values of different surfaces
- Submit completed program ***LineSense***
- Mission 5 Obj. 1-2 Review Kahoot!

Success Criteria

- ☐ Read values from a single line sensor
- ☐ Use the debugger to see values in real time
- ☐ Print a value to the console panel
- ☐ Assign a Boolean value to a variable using an if statement

Teacher Materials in Learning Portal

- Mission 5 Lesson 1 Slides
- Mission 5 Lesson 1 Workbook
- Mission 5 Lesson 1 Log
- Mission 5 Lesson 1 Answer Key

Additional Resources

- Mission 5 Obj. 1-2 Review Kahoot!
- LineSense_obj2 sample code (learning portal)
- Testing Surfaces paper (learning portal)
- Also needed: additional surfaces for testing reflectivity, such as a student desk, a folder, a piece of carpet, etc.

Vocabulary

- **Line sensors:** Photo reflective sensors that detect lines and boundaries beneath your 'bot.
- **API:** Application Programming Interface – the details of how your program interacts with different services it needs.
- **Analog:** Infinite variation, like from dark to light or cold to hot.
- **Digital:** Incremental variation, within a specific range.
- **ADC:** Analog to Digital Converter.
- **REPL:** Read Evaluate Print Loop – the command line that lets you type Python statements directly and observe what happens

New Python Code

`val = ls.read(0)`	Read a line sensor. The number of the line sensor is the (argument). It returns an integer between 0 and 4095.
`print(val)`	Print the value of a variable to the console panel.
`print("Line sensor value = ', val)`	Print the value of a variable with a text message.

Real World Applications

Many everyday items use sensing and control techniques. Two examples are:

- Robot vacuums

- Self driving cars

Teacher Notes:

- The slides should replace the instructions in CodeSpace. Code will be similar to CodeTrek, but a little different. All goals will be met.
- Objective 1 requires students to use the debugger. It has been awhile since students used the debugger. You may want to do this part together, or model it first.
- In objective 2, there is a slide on APIs. This isn't necessary for the lesson, and you can skip it if you want to.
- Extra slides are added that go into a little detail about analog and digital. You can use them or skip over them. This lesson is on the shorter side, so you have time to discuss the topic.

Extensions / Cross-Curricular:

- Expand the chart on reflectivity by adding more surfaces.
- **MATH:** Add an additional chart for distance from the sensor. Use the same surfaces, but increase the distance from the sensor. Compare the values. Create a line graph for the data.
- **SCIENCE:** Expand on the information about analog and digital. Look at examples of each and compare the data that each produces.
- Supports **language arts** through reading instructions, guided notes, and reflection writing.

MISSION 5: Fence Patrol **Lesson 2 (Objectives 3-5)**	**Time Frame: 35-40 minutes**
Project Goal: Students will use functions to read data from CodeBot's line sensors and use the data to control the line sensor LEDs. **Learning Targets** • I can assign a Boolean value to a variable using an if statement. • I can use a Boolean variable to turn on/off an LED. • I can define a function that reads a line sensor and uses the data to turn on/off an LED. • I can define a function that uses a loop to call another function.	**Key Concepts** • Use threshold comparisons to make decisions with sensor data. • A Boolean variable can get its value directly from an if statement. • A function can use a parameter to indicate which line sensor to read and LED to turn on/off. • A function can call another function as part of its code block. • A while loop that repeats a specific number of times can be used to call a function multiple times.
Assessment Opportunities • Mission 5 Lesson 2 Log (digital) • Quiz after Objective 5 • Submit completed program ***LineSense*** • Mission 5 Obj. 3-5 Review Kahoot!	**Success Criteria** ☐ Use an if statement to assign a Boolean value to a variable ☐ Use a Boolean variable to turn on/off an LED ☐ Define a function that reads a line sensor and then uses the data to turn on/off an LED ☐ Define a function with a while loop to call another function
Teacher Materials in Learning Portal • Mission 5 Lesson 2 Slides • Mission 5 Lesson 2 Log • Mission 5 Lesson 2 Answer Key	**Additional Resources** • Mission 5 Obj. 3-5 Review Kahoot! • LineSense_obj5 sample code (learning portal) • Testing Surfaces paper (learning portal)

Vocabulary

- **DRY:** Don't Repeat Yourself – never write the same code twice.
- **Function:** (Review) A named chunk of code you can run anytime just by calling its name; lets you reuse code without retyping it or copy/paste.
- **Parameter:** (Review) A variable that gets its value when the function is called; part of the function definition.
- **Argument:** (Review) A value that is passed to a function during a function call.

New Python Code

`is_detected = ls.read(0) > threshold`	Assign a Boolean value to a variable using an if statement.

`while true:`
`  is_detected=ls.read(0)>threshold`
`  leds.ls_num(n, is_detected)`
	Use a Boolean variable to turn on/off an LED.

`Surface detection`	Dark line on light surface – use val > threshold Light line on dark surface – use val < threshold

`def detect_line(n):`
`  is_detected=ls.read(n)>threshold`
`  leds.ls_num(n, is_detected)`
	Define a function with a parameter for detecting a line.

`detect_line(0)`	Call a function that has a parameter for detecting a line.

`n = 0`
`while n < 5:`
`  detect_line(n)`
`  n = n + 1`
	While loop that repeats 5 times.
	This loop uses n as the control variable, which is initialized outside the loop and incremented inside the loop. It is used to determine which line sensor to read and which LED to turn on/off.

`def scan_lines():`
`  n = 0`
`  while n < 5:`
`  detect_line(n)`
`  n = n + 1`
	A function that calls another function.

Real World Applications

Functions are a form of abstraction, which is used all the time in real-world applications. Abstraction enables you to simplify systems to focus on essential features while hiding the details.

- You can be given a list of chores to complete without step-by-step directions for each chore.
- You don't need to know how an engine works to drive a car.
- You can follow simple directions to get to a destination without needing all the details.
- The chorus of a song is like a function that is called multiple times.

Teacher Notes:

- The slides should replace the instructions in CodeSpace. Code will be similar to CodeTrek, but a little different. All goals will be met.
- This lesson has students define and use functions. Do you need to review this process?
- You may want to review incrementing a variable and how a counter can be used to control a while loop for repeating a specific number of times.
- You may want to review using a variable for lighting LEDs.

Extensions / Cross-Curricular:

- Change the while loop to start at the left LED and sweep right, instead of right to left. This means initializing the control variable to 5 and counting down.
- Play a tone when an LED turns on, or when an LED turns off.
- **MATH:** This program uses a function. Discuss what a function is in math. Compare and contrast.
- **LANGUAGE ARTS:** Have students write about an abstraction used in their daily lives.
- Supports **language arts** through reading instructions, guided notes, and reflection writing.

MISSION 5: Fence Patrol **Lesson 3 (Objectives 6-8)**	**Time Frame: 40-45 minutes**
Project Goal: Students will use data to count line detections and display the count in binary using the red user LEDs. **Learning Targets** • I can add a "wait" feature to a program. • I can define a function that returns a **Boolean** value. • I can make a contact **counter** to show each line-detect on the user LEDs. • I can use a programming technique to avoid a runtime error.	**Key Concepts** • Engineers build in safety features, like waiting for a button press before starting. • Reusing code is fundamental. This lesson uses the same or similar code from two previous programs. • The use of Boolean values (and variables) is also fundamental. They can be used as a toggle, like turning on/off LEDs, and also as a detection tool. • Functions can return a value as well as receive one. To return a value, use the "return" statement. If a "return" statement isn't explicitly given, the function will return None.
Assessment Opportunities • Mission 5 Lesson 3 Log (digital) • Submit completed program ***LineSense*** • Mission 5 Obj. 6-8 Review Kahoot!	**Success Criteria** ☐ Use a while loop to wait for a button press ☐ Use a return statement in a function ☐ Call a function that has a return statement ☐ Use a function's return Boolean value as a condition ☐ Define and increment a counter variable ☐ Display a number in binary using LEDs ☐ Avoid a runtime error by resetting a counter variable when it reaches its maximum number
Teacher Materials in Learning Portal • Mission 5 Lesson 3 Slides • Mission 5 Lesson 3 Log • Mission 5 Lesson 3 Answer Key	**Additional Resources** • Mission 5 Obj. 6-8 Review Kahoot! • LineSense_obj5 sample code (learning portal) • Testing Surfaces paper (learning portal)

Vocabulary

- **Return statement:** Exits the function and sends a value back to the code where the function was called.
- **Runtime error:** A coding error that happens when the program is actively running.

New Python Code

`while True:`
`  if buttons.was_pressed(0):`
`  break`
	(Review) Wait loop.
	A safety feature; the 'bot waits until BTN-0 is pressed before continuing the program.

`return is_detected` `return got_line`	Function return The value of the variable is returned to the function call.
`hit = scan_lines()` `if detect_line()`	Function call The value of the return is used in the assignment or if statement.
`leds.user(line_count)`	Use a variable to turn on user LEDs. Line count needs to have a value from 0 to 255.
`line_count = line_count + 1` `if line_count == 256:`	Reset a counter variable when it reaches its maximum number.

`line_count = 0`	

Real World Applications

Many everyday items count things. Computers and sensors are used in many real world applications, especially to gather data for better decision making. Some examples are:

- In manufacturing, like counting parts on an assembly line
- In retail, such as customers entering stores or how many items of something are on the shelf
- In agriculture, with crop monitoring and livestock counting
- In healthcare, such as counting blood cells
- In smart cities, controlling traffic flow and monitoring pedestrians

Also, many of the real world applications don't have a computer screen to display information. LEDs and other methods of visualizing data are seen throughout everyday items.

Teacher Notes:

- The slides should replace the instructions in CodeSpace. Code will be similar to CodeTrek, but a little different. All goals will be met.
- This lesson uses the "wait" code segment from Mission 4 Lesson 3 RobotMoves. Students should have access to the program so they can copy and paste the code.
- The lesson introduces returning a value from a function. Some examples are given, and students have some sample code to try in the mission log.
- The code for resetting a count variable when it reaches its max value is reviewed from Mission 4 Lesson 1 SweepLEDs. Students need to access the code, but you can review it if you want.

Extensions / Cross-Curricular:

- Play a tone when the counter variable is incremented.
- **MATH:** This program uses a function that returns a value. Discuss what a function is in math. Four sample functions are given in the mission log. Students can also write their own and practice evaluating functions and returning the value.
- **MATH:** Review binary numbers and why 255 is the maximum number for 8 LEDs. Come up with other problems and discuss how many bits, or LEDs would be required to display the max number. For example, how many bits are required to keep track of all student numbers, or books in a collection?
- Supports **language arts** through reading instructions, guided notes, and reflection writing.

MISSION 5: Fence Patrol **Lesson 4 (Objective 9)**	**Time Frame: 30-45 minutes**
Project Goal: Students will create a program that reads line sensors and uses the data to move CodeBot within a boundary. **Learning Targets** • I can define a function that moves the 'bot. • I can call the function that moves the 'bot at the appropriate time/place. • I can program the 'bot to stay inside a boundary.	**Key Concepts** • Autonomous robots use sensor data to make decisions and take action in its unique environment. • Functions can be written for 'bot movement as well as reading sensors. The movement functions will not return a value.
Assessment Opportunities • Mission 5 Lesson 4 Log (digital) • Submit completed program ***LineSense*** • Mission 5 Review Kahoot!	**Success Criteria** ☐ Define and call a function that moves the 'bot forward ☐ Define and call a function that moves the 'bot backward and then turns ☐ Program the 'bot to move inside a boundary line
Teacher Materials in Learning Portal • Mission 5 Lesson 4 Slides • Mission 5 Lesson 4 Log • Mission 5 Lesson 4 Answer Key	**Additional Resources** • Mission 5 Review Kahoot! • LineSense_final sample code (learning portal) • Batteries for each CodeBot • A white poster with black electrical tape for a boundary. You can start with a basic rectangle shape, but also try different shapes for the boundary on different posterboards.

Vocabulary

- **Algorithm:** (Review) A list of step-by-step instructions the computer can follow to complete a task.

New Python Code

`def go_forward():`
`  motors.run(LEFT, 45)`
`  motors.run(RIGHT, 45)`
	Define a function for 'bot movement.

`else:` `  go_forward()`	Call a function for 'bot movement.

Real World Applications

- Automatic Guided Vehicles (AGVs) use this kind of code to zoom around warehouse distribution centers, getting packages to you!
- Robots are used to clean up environmental waste, explore underground mines, discover shipwrecks, and do other tasks deemed unsafe for humans.

Teacher Notes:

- The slides should replace the instructions in CodeSpace. Code will be similar to CodeTrek, but a little different. All goals will be met.
- This lesson isn't very long, depending on the time students need for testing and debugging. Students can use extra time with extensions or cross-curricular activities.
- The simplest boundary to test the code is a rectangle. You can get posterboard at the dollar store and black electrical tape to create the boundary. After initial testing, try creating different shapes on posterboard and see how the 'bots do with other shapes. You can even try a white line on a dark surface.
- The Kahoot! for this lesson is a review of the entire mission, not just Objective 9.

Extensions / Cross-Curricular:

- Blink an LED while waiting for the BTN-0 press.
- Play a tone when line_count is incremented.
- Instead of sleeping when turning, play random beeps, similar to the animatronics program.
- Add code for pressing BTN-1 to stop the program.
- Program BTN-0 to test for a black line on a white surface, and BTN-1 to test for a white line on a dark surface.
- Use random numbers. This could be for speed, the amount of sleep, the pitch to beep, etc.
- **PHYSICAL SCIENCE:** Do experiments with different speeds and thresholds. Which ones work the best? Collect the data for the trials and create a chart or graph. Make predictions, and then test test code.
- **ART:** Attach a pen or marker to the 'bot and have it draw a graphic on paper as it stays within the lines.
- **LANGUAGE ARTS:** Have students write a poem about programming, or functions and abstraction, or robots.
- Supports **language arts** through reading instructions, guided notes, and reflection writing.

MISSION 6: Line Follower **Lesson 1 (Objectives 1-3)**	**Time Frame: 40-45 minutes**
Project Goal: Students will detect lines using line sensors and display the data on LEDs using a Boolean list. **Learning Targets** • I can define a list and access its items. • I can update items in a list using sensor data. • I can use REPL to gather real-time data from sensors. • I can use a list of Boolean values to turn on/off LEDs.	**Key Concepts** • There is no "hidden magic" going on here! The 'bot responds to programmed inputs and outputs. • The line follower 'bot will need to continuously check for the presence of a line beneath all five sensors. • The debug window in the console panel can display current values of variables. The console window can be used to test code.
Assessment Opportunities • Mission 6 Lesson 1 Log (digital) • Submit completed program ***CheckLines*** • Mission 6 Obj. 1-3 Review Kahoot!	**Success Criteria** ☐ Define a list with five Boolean values ☐ Define a function that returns a list of bools ☐ Update items in the list using sensor data ☐ Observe list values in the debug window ☐ Test a line of code in REPL (console window) ☐ Use a list of bools to turn on/off LEDs
Teacher Materials in Learning Portal • Mission 6 Lesson 1 Slides • Mission 6 Lesson 1 Log • Mission 6 Lesson 1 Answer Key	**Additional Resources** • Mission 6 Obj. 1-3 Review Kahoot! • CheckLines_obj3 sample code (learning portal) • Test Surfaces paper

Vocabulary

- **List:** A sequence of items you can access with an index.
- **Index:** The position of an item in the list; used to access a specific item.
- **REPL:** Read Evaluate Print Loop – the command line that lets you type Python statements directly and observe what happens

New Python Code

`a_list = [4, 2, 5, 3, 6, 9, 1, 0]` `detected = [False, False, False, False, False]`	Define a list
`num_items = len(a_list)`	Length of a list (number of items)
`first_item = a_list[0]`	Access a single item in a list.
`if is_detected:` `  Detected[n] = True`	Update the item at index n.
`leds.ls([True, True, False, False, False])`	Use a list of bools to turn on/off LEDs.
`vals = check_lines(2000)` `leds.ls(vals)`	Return a list of Boolean values, and then use the list to turn on/off LEDs.

Real World Applications

Lists are used in the real-world and in programming in many ways. They are often used to manage collections of data, such as items in an inventory system or locations in a game simulation. Lists are just one type of data abstraction.

Teacher Notes:

- The examples given in the slides are more inline with variables and examples from Mission 5.
- The code used in the slides is similar to CodeTrek, but a little different. It uses code and variable names from the program in Mission 5. All goals will be met.
- Slides go into a lot more detail about lists, giving more examples and practice.
- I do not recommend using the quiz after Obj. 2. Since the examples and program code are slightly different, the quiz questions aren't applicable if you are using the slides instead of the instructions.

Extensions / Cross-Curricular:

- Add code or pressing BTN-1 to stop the program.
- Add a sound when a line is detected, and a different sound (or no speaker) when no line is detected.
- **SCIENCE:** Test the sensitivity of CodeBot's sensors by trying different threshold values and tracking the data.
- **MATH:** Create a chart from the data collected during the experiment listed for science.
- Supports **language arts** through reading instructions, guided notes, and reflection writing.

MISSION 6: Line Follower **Lesson 2 (Objectives 4-6)**	**Time Frame: 40-45 minutes**
Project Goal: Students will create a line follower that uses two sensors for turning and three sensors for moving forward. **Learning Targets** • I can call the ls.check() function to use the ADC hardware channel scanning feature. • I can use an item in a tuple as a condition. • I can use logic operators to increase the reliability of the 'bot.	**Key Concepts** • The line follower 'bot will need to continuously check for the presence of a line beneath all five sensors. • A list and a tuple are similar, but different. A list is mutable, or can change, and tuple is immutable, or cannot be modified. • Pre-coded functions from the botcore library can use the ADC hardware, making your program even faster!
Assessment Opportunities • Mission 6 Lesson 2 Log (digital) • Submit program ***LineFollow*** (through Obj. 6) • Mission 6 Obj. 4-6 Review Kahoot!	**Success Criteria** ☐ Call the ls.check() pre-coded function. ☐ Use an item from the **vals** tuple as a condition. ☐ Turn if an outside sensor detects a line. ☐ Use the logical operator "or" to check multiple conditions. ☐ Move forward if a line is detected.
Teacher Materials in Learning Portal • Mission 6 Lesson 2 Slides • Mission 6 Lesson 2 Log • Mission 6 Lesson 2 Answer Key	**Additional Resources** • Mission 6 Obj. 4-6 Review Kahoot! • LineFollow_obj6 sample code (learning portal) • Test Surfaces paper • A track for the 'bot to follow

Vocabulary Review

- **int data type:** A value that is an integer; designated by **int** in Python; can be positive or negative.
- **float data type:** A value that is a decimal, also known as a floating point; can be positive or negative.

Vocabulary

- **Tuple:** Read-only form of list.
- **Immutable:** Unable to be changed.
- **Logical operators:** And, or and not; operators that allow for multiple conditions in a comparison. Logical operators handle combinations of Boolean results.
- **Or:** Allows for multiple conditions where only one must be true for the entire comparison to be true.

New Python Code

`vals = ls.check(2000, False)` `vals = ls.check(thresh, is_reflective)`	Botcore line sensor function (similar to check_lines but faster) Thresh is the threshold for detecting a line. False for a black line, True for a white line. The function returns a tuple of bools (not a list)
`ls.check(0)`	Returns the line sensor readings when entered in the Console Panel.
`elif vals[1] or vals[2] or vals[3]:`	Uses the logical operator "or" for multiple conditions. If any condition is true, the statement will evaluate to true.

Real World Applications

Self-driving cars, autonomous flying drones and other computing systems that navigate on their own have some basic

principles in common. Whether writing code for a vehicle with a high-powered vision processing system or for CodeBot's low-power sensors, you will face many of the same challenges to achieve the objective. Based on sensor inputs, what actions should you take to stay on the path? Robots that zip through warehouse distribution centers often follow lines as they pick up and pack items you order when shopping online.

Teacher Notes:

- The examples given in the slides are more inline with variables and examples from Mission 5.
- The code used in the slides is similar to CodeTrek, but a little different. It keeps consistent variables and sections of code. All goals will be met.
- Slides go into a lot more detail about tuples, giving more examples.
- Students will need lines for their 'bots to follow. You can get white posterboard from Dollar Tree and black electrical tape. Give the posterboard and tape to the students to make their own paths. You want a variety of paths for testing.

Extensions / Cross-Curricular:

- Add code for pressing BTN-1 to stop the program.
- Add a sound when the 'bot turns left, another sound when the 'bot turns right, and a different sound (or no speaker) when no line is detected.
- **SCIENCE:** Look up the schematics for the CodeBot. Look for the ADC and study the electronics chart. Learn more about the ADC.
- **MATH:** Compare the line sensor readings for check_lines() and ls.check() using different shades of gray and surfaces. Create a table from the data. Make a graph from the table.
- Supports **language arts** through reading instructions, guided notes, and reflection writing.

MISSION 6: Line Follower
Lesson 3 (Objectives 7-8)

Time Frame: 40-45 minutes

Project Goal: Students will create a line follower program that uses all five sensors for proportional steering control.

Learning Targets

- I can define a function for driving at variable speeds.
- I can represent a tuple of bools with 1s and 0s.
- I can compare a variable to a tuple.
- I can use tuples to create proportional speed control.
- I can calibrate the 'bot by setting the thresh and is_reflective variables.
- I can use the "global" keyword to keep variables global.

Key Concepts

- A tuple can be used in a comparison, making it easier to check for multiple conditions at the same time.
- Auto-calibration algorithms can make your 'bot adaptable to any environment.
- Global variables exist outside a function, while local variables are created inside a function. You can declare a variable global in a function.
- Python has many built-in functions for math.

Assessment Opportunities

- Mission 6 Lesson 3 Log (digital)
- Submit program ***LineFollow*** (obj7 and/or obj8)
- Mission 6 Obj. 7-8 Review Kahoot!

Success Criteria

- ☐ Define the drive() function with two parameters
- ☐ Use cases of detection for proportional speed control
- ☐ Compare a variable to a tuple
- ☐ Use pre-coded math functions abs() and round()
- ☐ Define the calibrate() function
- ☐ Call the calibrate() function when a button is pressed

Teacher Materials in Learning Portal

- Mission 6 Lesson 3 Slides
- Mission 6 Lesson 3 Log
- Mission 6 Lesson 3 Answer Key

Additional Resources

- Mission 6 Obj. 7-8 Review Kahoot!
- LineFollow_obj7 sample code (learning portal)
- LineFollow_obj8 sample code (learning portal)
- A track for the 'bot to follow

Vocabulary

- **Calibrate:** Using sensor readings to determine the values of variables; adapting code to the environment using data.
- **Locals:** Variables defined inside a function; they only exist while the function is running and can only be accessed in the function.
- **Globals:** Variables defined outside of a function; they are available during the entire program and can be accessed throughout the entire program.

New Python Code

`def drive(left, right):`
`  motors.run(LEFT, left)`
`  motors.run(RIGHT, right)`
	Define a function or driving the 'bot that uses parameters or the left and right wheel speeds.

`if vals == (1, 0, 0, 0, 0):` `  drive(0, 30)`	Use the tuple in a comparison. Call the drive() function, selecting left and right speeds as the arguments.
`sensors = ls.check(0)`	Return a tuple of integers for the line sensor readings.

`if abs(gap) < 500:`	Use the absolute value function.
`is_reflective = line < ground`	Use a condition to set a Boolean value.
`thresh = round(ground - (gap/2))`	Use the round function; the result is an integer.
`global thresh, is_reflective`	Used at the beginning of a function, the “global” keyword makes the variables global instead of local. The variables can be listed in any order.
`elif buttons.was_pressed(1):` `  calibrate()`	Call a function when a button is pressed.

Real World Applications

Calibration is vital to many physical devices. Calibrated items are tools and instruments that measure physical or electrical properties to ensure accuracy against a known standard. Common examples include temperature sensors, pressure gauges, weighing scales, calipers, micrometers, torque wrenches, pH meters and electrical multimeters.

Teacher Notes:

- This lesson covers objectives 7 and 8. Look over objective 8 on calibration and decide if you want your students to do it. The program works fine without it. If you do cover obj. 8, it gets into global and local variables and built-in math functions. The quiz is also optional. You should skip it if you don’t do objective 8.
- The slides for objective 7 go into more detail about the tuples used, and give additional examples.
- The CodeSpace instructions for objective 8 have a lot of reading. The slides simplify the instructions but still complete the code from CodeTrek.
- The extensions are the same as the previous lesson. Students may think of their own extension for the program.

Extensions / Cross-Curricular:

- Add code for pressing BTN-1 to stop the program.
- Add a sound when the ‘bot turns left, another sound when the ‘bot turns right, and a different sound (or no speaker) when no line is detected.
- **SCIENCE:** Discuss friction and turning speeds. Experiment with the ‘bot on slow turning speeds vs fast turning speeds and what type of angles they work best on. Try different surfaces for friction.
- **SCIENCE:** Have a discussion about calibration. Look at scientific tools and instruments that either must be calibrated or are used for calibration.
- **ENGLISH LANGUAGE ARTS:** Write a short story about an autonomous physical device that follows a line.
- **ENGLISH LANGUAGE ARTS:** Write a technical report on how the line follower program works.
- Supports **language arts** through reading instructions, guided notes, and reflection writing.

Unit 3 Remix Project	**Time Frame: 2-5 hours**
Project Goal: Students will use the skills and concepts they learned in the unit to create their own project. **Learning Targets** • I can summarize the programming concepts from Missions 5 & 6. • I can plan an original program. • I can create an original program using concepts and code from previous programs. • I can get feedback on my project.	**Key Concepts** • Code segments from previous programs can be reused and repurposed in a new project. • The program development in the planning guide follows the software design process. • Creating a new project from the beginning, without CodeTrek or starter code, is an excellent way for students to master their learning and gives them an opportunity to express themselves and work on something that interests them.
Assessment Opportunities • Unit 3 Remix Planning Guide • Unit 3 Remix Project	**Success Criteria** ☐ Plan an original program ☐ Create an original program ☐ Incorporate feedback in a program
Teacher Materials in Learning Portal • Unit 3 Remix Project slides • Unit 3 Remix Planning Guide	**Additional Resources** • Students can use their previous programs as a guide throughout this project.
Teacher Notes: • A remix for this unit is optional. However, it is an excellent opportunity for students to create their own original program by doing something that interests them. And the remix project gives students a chance to practice and apply what they have learned. • Students can work with a partner for this project. Collaboration is an important skill. As an option, if working with a partner, each student could write code for one of the buttons, and they can combine the code into a finished project. • A set of slides is prepared to explain the project and give step by step guidance. The slides also give some suggestions for the project. The suggestions are meant to help students think of their own ideas and should not be required. They can be used for students who are drawing a complete blank, or as inspiration. • A planning guide is provided to help students know where to start, and to guide them throughout the process. You can modify the planning guide as needed by changing or adding to the questions. • Consider how you want to end the remix project. You can have students present them to the class, have a “gallery walk” of projects, have students create a slide show about the project, etc. • A checklist for the remix project is below. You can modify the checklist for specific requirements you want to look for.	

Remix Project Checklist:

- ☐ Filename is descriptive
- ☐ Uses one or more variables, each with a descriptive name
- ☐ Uses line sensor data
- ☐ Moves the CodeBot
- ☐ Turns on at least one LED light
- ☐ Uses at least one button for input
- ☐ Defines and calls at least one function
- ☐ Includes comments and blank lines for readability
- ☐ Code runs without errors