MISSION 7: Hot Pursuit **Lesson 2 (Objectives 4-7)**	**Time Frame: 45-50 minutes**
Project Goal: Students can write calibration functions so the 'bot can adapt to its environment. **Learning Targets** • I can write code that calibrates the detection sensitivity threshold. • I can write a function for the calibration. • I can write a function that calibrates the power level.	**Key Concepts** • Using auto calibration functions for **power** and **thresh** allows the 'bot to adapt to a new environment. • An accepted programming convention is to group functions together, typically at the beginning of a program. • Print statements can take multiple arguments. It converts them to strings and prints them back-to-back to the console. Each argument is separated with a comma.
Assessment Opportunities • Mission 7 Lesson 2 Log • Submit completed program ***HotPursuit*** • Mission 7 Obj. 4-7 Review Kahoot!	**Success Criteria** ☐ Use an if statement to detect a button press ☐ Use if statements to determine the best detection sensitivity threshold ☐ Define a function to auto-calibrate thresh ☐ Define a function to auto-calibrate power ☐ Use the global command when assigning values to global variables inside a function ☐ Use multiple arguments in a print statement ☐ Test the program with multiple surfaces and record the results
Teacher Materials in Learning Portal • Mission 7 Lesson 2 Slides • Mission 7 Lesson 2 Mission Log • Mission 7 Lesson 2 Answer Key	**Additional Resources** • Mission 7 Obj. 4-7 Review Kahoot! • HotPursuit_obj4 sample code (learning portal) • HotPursuit_obj6 sample code (learning portal) • HotPursuit_obj7 sample code (learning portal) • HotPursuit_ext2 sample code (learning portal)

Vocabulary

- **Calibrate:** Using sensor readings to determine values of variables; adapting code to the environment using data.
- **Automate:** Use technology, like a computer, to do a task automatically.
- **Default:** A pre-selected option.
- **Globals:** Variables defined outside of a function; they are available during the entire program and can be accessed throughout the entire program.
- **Locals:** Variables defined inside a function; they only exist while the function is running and can only be accessed in the function.

New Python Code

`if sensed[LEFT] > 0:` `  det = sensed[LEFT]`	After reading the proximity sensors (sensed), check only the LEFT sensor reading to see if it detected a reflection. If so, assign its sensitivity value to a variable.

`det = min(det, sensed[RIGHT])`	The math function min() finds the lowest, or minimum, value of the arguments. In this example, it selects the lowest value of either the current det or the sensitivity value of the RIGHT sensor.

`if det > 100:`
`  thresh = 100`
`else:`
`  thresh = det - 5`
	Bug fix to make sure thresh is the correct value.

`global thresh` `global power`	Keeps a global variable as global, even when it is assigned a value inside a function.

`while power < 9:`
`  cal_thresh()`
`  if thresh < 100:`
`  break`
`  power = power + 1`
	Loop that cycles through the range of powers to determine the best power level for the proximity sensors.

`print("Power=", power)`	Print statement with multiple arguments.

Real World Applications

Many technical devices use auto-calibration to ensure the device maintains accuracy. Here are some examples:

- Adaptive cruise control on cars auto-calibrates to measure distances accurately and prevent ghost braking.
- Smartphone accelerometers and gyroscopes automatically calibrate for orientation.
- In labs, automated pipettes can perform self-calibration to ensure the volume of liquid matches the display.

Teacher Notes:

- This lesson is all about auto-calibration, which was also introduced in Mission 6. If you skipped auto calibration (Mission 6 Objective 8), then all the information you need is included here. If your students completed Obj. 8, then this will be review.
- This lesson can run long if students are doing a lot of testing, or if calibration is a new topic. You can consider running the lesson over two class periods and including a cross-curricular activity or the extension as well.
- This lesson goes over global and local variables. If your students did Mission 6 Obj. 8, then it will be review. Otherwise, you may want to take time on this and make sure students understand the concept.
- The code used during the lesson is slightly different from CodeTrek. All goals will be met.
- Students will need different surfaces for testing. You can use the Testing Surfaces paper with black/white/gray options, or any color or surface.
- There is a quiz after Objective 5. It doesn't really review the concepts from 4 and 5. You have the

Extensions / Cross-Curricular:

- After Obj 7, the code can still throw an error if the 'bot is not under a surface. Fix the bug by using an if statement.
- **SCIENCE/MATH:** Experiment with the proximity sensor using different lighting, from very bright to pretty dark. See what range of IR light and sensitivity work best in different conditions. Make a chart of the power/thresh.
- **MATH:** The proximity sensors can give a different reading, even with the same surfaces and lighting. Run the auto-calibration sequence several times without moving the 'bot and record all the power/thresh values. Find the mean, median and mode for the surfaces.
- Supports **language arts** through reading instructions, guided notes, and reflection writing.

option of skipping the quiz, or going over it together as a class.	

Preparing for the lesson:

- Look through the slides. Decide what materials you want to use for presenting the lesson. The slides can be converted to Google Slides. They can be projected on a large screen.
- Be familiar with the mission log assignment and the questions they will answer. Prepare the assignment to give through your LMS.
- Have at least three colors/surfaces available for each student or programming pair. You can use the Testing Surfaces paper from Unit 3 as an option. Any colors or surfaces can be used.
- If you have a word wall, or another form of vocabulary presentation, prepare the new terms.

Lesson Tips and Tricks:

💡 **Teaching tip:**

You can use a variety of discussion strategies to get the most engagement from your students. For example, you can have students write their answers before asking anyone for an answer. You can use one of many think-pair-share methods.

👫 **Pre-Mission Warm-up:** -- slide 2

Students can write in their log first and then share, or discuss first and then write in their log. The warm-up question reviews proximity sensors. Students can share their answers, or compare with each other.

- Question: How is the proximity sensor similar to the line sensor?

💻 **Mission 7 Lesson 2 Activities:**

The Chrome browser works best, but other browsers also support CodeSpace. Each student will complete a Mission Log. Students could work in pairs through the lesson, or they can work individually. There is a lot of testing in this mission, so it is a good one for pair programming.

💡 **Teaching tip: Mission Introduction** -- slides 3-6

This mission is divided up into three lessons. This lesson completes the fourth goal. Students answer one question in their mission log.

💡 **Teaching tip: Objective #4** -- slides 7-10

These slides introduce the topic of auto calibration and give an example of how it should work.

💡 **Teaching tip: Objective #4** -- slides 11-14

These slides go over the algorithm for the first auto-calibration and show the code that goes with each step. The last step includes printing to the console panel. The line of code has a print statement with multiple arguments. All can be displayed on one line in the console panel. Students may have a question about this.

💡 **Teaching tip: Objective #4** Activity -- slides 15-19

Students open their program HotPursuit and add code. The code is given on slide 17. All the code, except the if statement that checks for a button press, was shown on slides 11-14.

Students need to open the console panel when running the code to see the thresh values. They will place the ‘bot on several different surfaces. They can use the Testing Surfaces paper from Unit 3, or any surfaces. They don’t have to stop running the code before changing the surface. They simply press BTN-1 each time they want a new reading. They will record their results on the mission log.

💡 **Teaching tip: Objective #5** -- slides 20-21
This objective fixes a bug in the code by adding another if statement.

💡 **Teaching tip: Objective #5 Activity** -- slides 22-23
Students modify their code by adding the if statement. They don't need to do a lot of testing. Just one test with the 'bot pointed in the air, or with no surface, is needed. Record the result in the mission log.

💡 **Teaching tip: Quiz** -- slide 24
I recommend skipping the quiz or going over it together in class. It reviews concepts from Lesson 1, so it could be a good review. Quiz questions are posted below.

💡 **Teaching tip: Objective #6** -- slides 25-26
Students can organize their code by taking the block under the if statement and making it a function.

💡 **Teaching tip: Objective #6** -- slides 27-29
This covers the concept of global and local variables. If this is your first calibration function, you should take time to go over this carefully. If your students did the calibration in mission 6, this will be review and can be skipped.

💡 **Teaching tip: Objective #6** -- slide 30
This reviews defining a function, specifically for the calibration function.

💡 **Teaching tip: Objective #6 Activity** -- slides 31-34
Students define a function for calibrating thresh. The code is given on slides 32-33.

There isn't any specific testing to do for this activity. The code should work exactly the same as Obj. 5. The questions in the mission log review global and local variables.

💡 **Teaching tip: Objective #7** -- slide 35
Discussion on defining another function for calibrating power.

💡 **Teaching tip: Objective #7** -- slides 36-38
These slides go over the algorithm for the second auto-calibration and show the code that goes with each step.

💡 **Teaching tip: Objective #7 Activity** -- slides 39-42
Students add the second calibration function and perform some testing. They will need to open the console panel to see the results, which they record in their mission log.

💡 **Teaching tip: Extension** – slides 43
One extension is given for this lesson. During the testing for Obj. 7, students may point the 'bot in the air, or have no surface underneath. This will cause the program to throw a runtime error. The extension is to fix this error by adding another if statement. Code is given for this extension.

Optional: 🗝️ Mission 7 Obj 4-7 Kahoot! Review. A review Kahoot! Is available for these four objectives.

👫 **Post-Mission Reflection:**

The post-mission reflection asks students to review what they learned during this lesson. Answers can vary widely, depending on each student's experience.

You can use a cross-curricular activity for a post-mission activity.

End by collecting the Mission 7 Lesson 2 Log.

SUCCESS CRITERIA:

- ❑ Use an if statement to detect a button press
- ❑ Use if statements to determine the best detection sensitivity threshold
- ❑ Define a function to auto-calibrate thresh
- ❑ Define a function to auto-calibrate power
- ❑ Use the global command when assigning values to global variables inside a function
- ❑ Use multiple arguments in a print statement
- ❑ Test the program with multiple surfaces and record the results

Quiz after Objective 5

Checkpoint

Which of these is the **best** criticism of this 🔧comment? +5 XP

```
# Set to 5% below minimum ground-reflection detected.
thresh = det - 5
```

- ☒ TLDR. You lost me at "minimum"...
- ☑ If someone later changes the code to a different % value, say 8%, the comment will be wrong.
- ☒ The word "bellow" is spelled with two l's, not just one.
- ☒ The % character is considered an offensive rune in Elvish.

What does the 🔧function `prox.detect()` **return?** +5 XP

- ☒ A **boolean**.
- ☒ A **tuple** of *two* **integers**.
- ☑ A **tuple** of *two* **boolean** values.
- ☒ *Nothing.*

What does the **function** `prox.range()` return *if* ***no reflection was detected.*** +5 XP

- ☒ `(True, True)`
- ☒ `(False, False)`
- ☒ `(False, True)`
- ☑ `(-1, -1)`