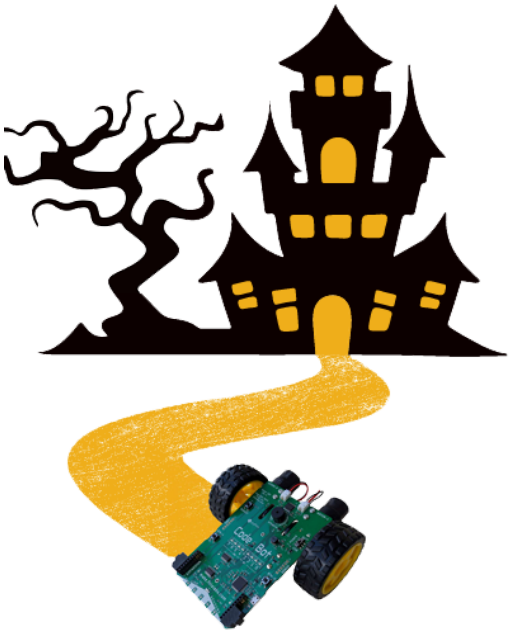
FIRIA LABS

Python with Robots

Haunted Code Chronicles 2025



Curriculum Guide

And Teacher Resources

CodeBot: Haunted Code Chronicles

Introduction

Welcome to the Haunted Code Chronicles – where coding meets the *eerie and unknown!* As the fog rolls in and the shadows lengthen, prepare to guide your CodeBot through mysterious mazes and terrifying trials, each more thrilling than the last. The Haunted Code Chronicles, back for its third year, is designed for all levels: beginner to advanced Python programmers. This free mission pack will spark a love for coding and add excitement to programming education. Anyone can access the free mission pack in CodeSpace.

This year teachers can request a free license for the mission pack. With a license teachers can set up a class in the dashboard, track student progress and control access to the missions. Teachers can choose to have students work independently at their own pace, or use the lesson plans and activity guides. You can even set up your own class competition and print certificates! Will you conquer the haunted halls and emerge victorious, or will the darkness consume you? There's only one way to find out. Join us in this year's Haunted Code Chronicles, *if you dare!*

Getting Started

As an educator, you have two ways to offer this course:

- Request a free one-month license for the mission pack. The license will let you set up a class for your students to join and enables you to track their progress.
- –OR– Each student can activate the free mission pack anytime and progress through the rooms independently.
- Once students have access to the curriculum, you can either let students work independently at their own pace, or work together as a class going through the missions. Students can even work with a partner.

Haunted Code Chronicles with a license:

1. Request a license and claim it in CodeSpace (make.firialabs.com). If this is your first time claiming a license, follow the instructions on this web page.
2. Set up a class in the dashboard and get a join code. For help, follow the instructions on this web page.
3. Have students log in to CodeSpace with a free account using any email address.
4. Students then use the join code to join your class to start their Python adventure.
5. Track their progress in the dashboard and use ideas from the lesson plans and activity guides provided. You can even lock/unlock missions using the Lesson Access Controller.

Students activate the mission pack individually:

1. Each student logs in to CodeSpace (make.firialabs.com) with a free account using any email address.
2. Each student selects the "CodeBot: Haunted Code Chronicles" mission pack from the class list by clicking on the green checkmark. Then start with "Mission 1: The Porch" and continue through the missions.
3. You can use ideas from the lesson plans and activity guides provided.

Additional Information

You can facilitate the mission pack in a variety of ways. Here are some suggestions:

- Students can use the activity guides for guided notes, or have them just follow the directions in CodeSpace.
- Keep track of student scores. If a student uses a more complex programming concept in their code solution, they get a higher score. Use them for a class competition.
- Give out certificates to students who complete the entire mission pack, for participating in the mission pack, or for each mission completed. Certificate templates are provided.
- Code solutions for the mission objectives are not provided. There are a number of ways each challenge can be met. Use the teacher tips included in the lesson plan for each objective for hints on possible solutions. If you and all your students are stuck or struggling for too long, you can send an email requesting help.

CodeBot: Haunted Code Chronicles

MISSION 1: The Front Porch

Time Frame: 45-120 minutes

Project Goal: Students will learn how to move CodeBot and play notes on the speaker.

Learning Targets

- I can use the features of CodeSpace (CodeTrek, Hints, text editor, objective panel, goals, etc.)
- I can identify and write code for CodeBot's LEDs and motors.
- I can use camera controls to view the scene.
- I can move the CodeBot forward, backward, rotated and in a curve.
- I can play notes using CodeBot's speaker.

Key Concepts

- The first objectives introduce CodeSpace and CodeBot (#1-9).
- There are many ways to program the 'bot to accomplish the task. Celebrate the creativity of your students.
- CodeBot's motors must be enabled before the wheels can move.
- Students cannot copy and paste from CodeTrek, but they can copy and paste code in the text editor.
- Comments are not required, but they are helpful.

Assessment Opportunities

- Complete the Mission 1 Activity Guide.
- Complete and turn programs:
  - ***ActionBot*** after Objective 9.
  - ***SpiderTag*** after Objective 10.
  - ***RatSpells*** after Objective 11.
  - ***DoorDash*** after Objective 13.
- Record the score for the mission, found in the Progress and Contests icon under Select Class.
- Write a paragraph about their coding experience and what they learned.

Success Criteria

- ☐ Identify key features of CodeSpace (text editor, objectives, objective panel, toolbox, camera controls, CodeTrek, Hints)
- ☐ Identify an LED and a motor on CodeBot
- ☐ Use camera controls to view the front porch
- ☐ Write code to turn on an LED
- ☐ Write code to rotate CodeBot
- ☐ Write code to move CodeBot around the porch
- ☐ Use the CodeBot speaker to play notes

Vocabulary

- **CodeBot:** A computer on wheels with lots of sensors and controls built-in
- **Motors:** Programmable electric engines; powers the wheels
- **LEDs:** Light emitting diodes; tiny and efficient electronic components that produce light
- **Comment:** Notes to the programmer that are ignored by the computer

New Python Code

`from botcore import *`	Import the botcore library so you can access CodeBot's hardware.
`leds.user_num(0, True)`	Turns on user LED 0
`from time import sleep`	Import the sleep function from the time library.
`motors.enable(True)`	Turn on, or enable, the motors.
`motors.run(LEFT, 40)` `motors.run(RIGHT, 40)`	Move forward in a straight line at speed 40.
`sleep(1.5)`	Pause program execution, but keep the motors running.
`motors.enable(False)`	Turn off, or disable, the motors.
`spkr.pitch(400) / spkr.pitch(note)`	Plays a specific note on CodeBot's speaker.
`spkr.off()`	Turns off the speaker.

CodeBot: Haunted Code Chronicles

CSTA Standards

Grades 3-5	Grades 6-8	Grades 9-10	Grades 11-12
• 1B-CS-02 • 1B-CS-03 • 1B-AP-08 • 1B-AP-12 • 1B-AP-15 • 1B-AP-17	• 2-CS-02 • 2-CS-03 • 2-AP-13 • 2-AP-16 • 2-AP-17 • 2-AP-19	• 3A-CS-03 • 3A-AP-13 • 3A-AP-14 • 3A-AP-16	• 3B-DA-07 • 3B-AP-10 • 3B-AP-11 • 3B-AP-16

Teacher Notes:

- Each mission has its own sound track that plays continuously. If the music gets overwhelming in the classroom, you can have students use headphones, or students can go to their preferences and mute the sound.
- Objectives 1-9 introduce CodeSpace and CodeBot. Students program their first real challenge in Objective 10. They also have a programming challenge in Objective 12 and Objective 13.

Additional Resources / Extensions:

- Mission 1 Activity Guide
- Mission 1 Activity Guide Answer Key
- Promotional Trailer (YouTube)
- Haunted Code Chronicles information (blog - also has embedded video)
- Supports **language arts** through reading instructions and reflection writing.

Preparing for the lesson:

Students need an email address to use for creating an account.

- Decide if you want to track students by setting up a class in the dashboard. If so, request and claim a license.
- Look over the activity guide. Decide if you want students to use it. Will you print it or give it digitally?
- Decide on the structure of your class. Will students work independently, or as a class? Will you require students to turn in programs, or is completing goals appropriate for grading and assessment?

Lesson Tips and Tricks:

👫 Warm-up:

You can start by showing the mission pack trailer, or the blog page (links are above).

The warm-up questions can be written, or shared as a class discussion, or through a think-pair-share activity. Change the questions if you have a different perspective for students to focus on when starting the mission pack.

💻 Mission 1 Objectives:

Objectives #1-5

These objectives introduce students to the CodeSpace environment. The activity guide has specific instructions to help with meeting the goals of individual objectives.

Objectives #6-7

These objectives introduce students to CodeBot's motors and LEDs. Students will click on the parts in the 3D simulator.

Objective #8

This is the first time a student must create a new file. It must have the name spelled exactly the same as what is in the goal. The goal will be met by turning on 3 LEDs – the 0, 4 and 7. They can get a higher score by turning on more than 3. The answers to the activity guide questions are in CodeTrek.

Objective #9

Students continue to use the same file for this objective. Students will not delete any code, just add to it. CodeTrek shows the code only for the required LEDs to turn on. If students did more, they should keep the code. It doesn't have to match perfectly. Also, the speeds of the motors in the added code doesn't have to be exactly the same as CodeTrek.

The validator is only checking to see if the 'bot is rotated. However, students will move the 'bot A LOT during this mission pack, so they should practice all the different ways to move it. You can have students submit their ActionBot code for this objective if you want to check how they are doing.

Objective #10

Students create a new file for this objective. It does not have to be the same name as the one in the instructions.

The challenge is to move CodeBot so it tags at least four spiders. The spiders can be tagged in any order. Students get a higher score for tagging all six spiders, but it is not required. Also, students can get a higher score for speed. Important note: CodeBot doesn't move exactly the same way every time the code is run. If someone's code looks good, but the program run is a little off, just reset the scene and run it again.

Helpful tips for students:

- Try different camera angles to find the one most useful to you.
- The easiest way to get three spiders is to back up, then move forward. Three spiders are in a row.
- Work at getting one spider at a time. Adjust the sleep and/or motor speed a little at a time. A student may have to run the code a few times, since it runs a little differently each time.
- Slower speeds have a little more consistency, so if someone struggles, have them go slower at first.

Objective #11

This objective does not use any programming. Students use the universal camera to explore the front porch and locate the envelope. They click on the envelope to meet the goal. The envelope is on the rocking chair.

Objective #12

The answer to the activity guide question is in the message from the envelope. This objective uses the CodeBot speaker to put rats to sleep. Make sure the speaker is enabled or you will not hear the notes as they play.

The notes for the melody are given in the note. The code for the rats doesn't have to be complicated. The code for the first rat is given in CodeTrek. Students can follow the pattern for each additional rat, looking for the notes in the dictionary on lines 6-9. The boss rat must hear his melody two times. Don't forget the silence in between. Reset the scene each time you run the code to see the rats disappear.

If your students have programming experience or want to try a shorter solution, they can use the hints. Two more ways to play the melodies are given. Each level of complexity will give the student bonus points.

Objective #13

This challenge is fairly straightforward: drive the 'bot through the hole in the door. CodeBot doesn't need to rotate or anything, just drive straight. A faster time gives a higher score. Students should remember to reset the scene each time they run their code. Don't forget to enable the motors!

👫 **Reflection:**

The reflection question can be written, or shared as a class discussion, or through a think-pair-share activity. This is a good time to review what was learned during the mission.

CodeBot: Haunted Code Chronicles

MISSION 2: The Living Room

Time Frame: 45-90 minutes

Project Goal: Students will learn how to read the proximity sensors and use them to detect when the television is on or off.

Learning Targets

- I can move the 'bot in a curve.
- I can identify the proximity sensors.
- I can read the proximity sensors and print the reading on the console.
- I can use an infinite while loop to constantly read the proximity sensors.
- I can use an if statement to turn on and off LEDs.
- I can use the proximity sensor readings to start and stop CodeBot.

Key Concepts

- The REPL is also known as the console. It can be used to print data. Students will print to the console during this mission.
- A while loop defines a block of code that is repeated while a condition is True. A while loop definition ends with a colon (:) and the block of code to be repeated must be indented.
- An if statement has at least one branch. It uses a condition that is True or False. If the condition is True, the first branch is executed. If the condition is False, the else branch is executed.
- An if statement definition ends with a colon (:) and the blocks of code in the branches are indented.
- The proximity sensors return a tuple of Boolean values, which are either True or False. The tuple looks like this: (True, True) or (False, False)

Assessment Opportunities

- Complete the Mission 2 Activity Guide.
- Complete and turn programs:
  - ***PopcornMunch*** after Objective 2.
  - ***LivingroomEscape*** after Objective 7.
- Quiz after Objective 5.
- Record the score for the mission, found in the Progress and Contests icon under Select Class.
- Write a paragraph about their coding experience and what they learned.

Success Criteria

- ☐ Munch several pieces of popcorn by touching them with CodeBot
- ☐ Identify the proximity sensors
- ☐ Read the proximity sensors and print the reading to REPL (the console)
- ☐ Use an infinite while loop to read the proximity sensors
- ☐ Use an if statement and the proximity sensor reading to turn on/off a line sensor LED
- ☐ Guide CodeBot through the door, moving only when the television is off

Vocabulary

- **Proximity Sensor:** Infrared sensors that enable the 'bot to detect objects based on reflected IR light.
- **REPL:** Repeat Evaluate Print Loop, also called the console. It is a way to interactively enter commands and view output in a text format.
- **While Loop:** A statement that tells Python to repeat a block of indented code as long as the given condition is True.
- **Infinite Loop:** A while loop that never ends because the condition is always True.

Optional Computing Terms

- **Function:** A named chunk of code you can run anytime just by calling its name.
- **Parameters:** Local variables that hold information received by a function.
- **Arguments:** Information passed to a function's parameters. Arguments can be literal or variable values.
- **Boolean:** A data type with two possible values: True or False.

New Python Code

`prox.detect()`	Read both the proximity sensors; returns True or False for each sensor: (True, True) or (False, False)
`print(light_reading)`	Print the light reading on the console.

`if light_reading == (False, False):`	Compare the sensor reading to the value (False, False).
`else:`	The False branch of an if statement.
`leds.ls_num(2, True)`	Turn on line sensor LED 2.
`def function_name():` `def munch(left, right, tm)`	Define a function. Function names follow the same rules as variables. The block of code inside a function must be indented.
`function_name()` `munch(55, 50, 4)`	Call a function (no parameters) Call a function with parameters

CSTA Standards

Grades 3-5	Grades 6-8	Grades 9-10	Grades 11-12
• 1B-CS-02 • 1B-CS-03 • 1B-DA-07 • 1B-AP-08 • 1B-AP-10 • 1B-AP-11 • 1B-AP-12 • 1B-AP-15	• 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)	• 3A-CS-03 • 3A-DA-12 • 3A-AP-13 • 3A-AP-14 • 3A-AP-16 • (3A-AP-17) • (3A-AP-18)	• 3B-CS-02 • 3B-DA-06 • 3B-AP-10 • 3B-AP-11 • (3B-AP-14) • 3B-AP-16

Teacher Notes:

- New programming concepts have very specific syntax. Students need to be really careful with how they type in the code.
- An infinite while loop will be used starting with Objective #5. The code will not end until the STOP button is pressed.
- The  icon is used to open the console. It changes to  , which is used to close the console.
- Whenever CodeTrek has **# TODO:** students should not type the comment, but they should type the actual code to accomplish the task. Sometimes CodeTrek will give additional help, and sometimes students are expected to know the code. They can look back in the Objective Panel to remember what to do.

Additional Resources / Extensions:

- Mission 2 Activity Guide
- Mission 2 Activity Guide Answer Key
- Supports **language arts** through reading instructions and reflection writing.

Preparing for the lesson:

- Go through the Mission in advance, if you can. Try at least the first couple of objectives.
- Be familiar with opening and closing the console.

Lesson Tips and Tricks:

👫 Warm-up:

The warm-up has two questions. The first one is a review of the code from Mission 1.

Before students answer the second question, go over the Mission 2 introduction. Emphasize the second paragraph about the television. You can even go to Objective 1 and close the Objective Panel to see the television and the face that appears on it. Then have them guess how CodeBot can detect if the face is showing. If they don't have previous coding or robotics experience, they probably won't know. Any guess is good, and at the end of the mission they can compare their guess with the actual sensor.

💻 Mission 2 Objectives:

Objective #1

Students should answer the two questions on the activity guide. Then they type the code from CodeTrek and only have to add the sleep() statement. They should not type the **# TODO:** comment, but instead type the actual code.

Objective #2

This objective can take a long time if students want to munch a lot of popcorn. You may want to set a time limit.

The concept of a function is introduced here. If your students are beginners, you can skip this part, and students should not use CodeTrek. If students choose to use a function, they need to be very careful with the punctuation. Functions should be defined near the top of the code, just under the import statements.

Objective #3

The goal can be met with or without a function. There isn't a time limit. The 'bot will not make it through the door. The goal is met by bouncing off the door. To increase their score, students can munch popcorn before bouncing off the closed door.

Objective #4

The proximity sensors are introduced. The goal is met by clicking one of the two proximity sensors in the simulator.

Objective #5

Students learn the code for reading the proximity sensors. They print the reading to the console. Run the code for a little while to see the two readings. Pay attention to the reading when the TV is on, and when the TV is off.

Students have two questions in the activity guide to answer before they run their code, and two questions to answer after they run their code.

Quiz

The quiz questions are the same questions from Objective 5.

Objective #6

The code for turning on/off the line sensor LEDs is shown. It is very similar to code for the user LEDs. CodeTrek doesn't show an import for the sleep library. If students keep the library, it is okay. Students need to be very careful with the punctuation. The if statement uses == to compare, not a single =. It must end with :.

Because this program uses an infinite while loop, students must stop the code by clicking the STOP button.

Objective #7

There are many ways to complete the challenge. Students should be encouraged to try it their own way. If they get frustrated, you can give a hint. One way to complete the challenge is to set the motors to the same power above the while loop. Then inside the if statement, the motors can be enabled or disabled, depending on when the TV is on or off. This is just a few lines of code to add to the program.

👫 **Reflection:**

The reflection questions can be written, or shared as a class discussion, or through a think-pair-share activity. This is a good time to review their earlier predictions and what was learned during the mission.

CodeBot: Haunted Code Chronicles

MISSION 3: The Kitchen

Time Frame: 45-120 minutes

Project Goal: Students will learn about decimal and hexadecimal numbers and bitwise operators, and use them in code to control LEDs.

Learning Targets

- I can control the user LEDs using binary numbers.
- I can evaluate expressions with logical operators.
- I can evaluate expressions with bitwise operators.
- I can read a line sensor.

Key Concepts

- The binary number system has two digits: 0 and 1. Computers use them to represent ON and OFF, and they can also represent True and False.
- Hexadecimal numbers use base 16, and they are also used by computers to represent numbers.
- Binary and hexadecimal numbers can be used to turn on and off a string of LEDs.
- Logical operators allow for a combination of conditions. Logical operators are AND and OR.
- Bitwise operators are used to combine binary or hexadecimal numbers. Bitwise operators are & (and) and | (or).

Assessment Opportunities

- Complete the Mission 3 Activity Guide.
- Complete and turn programs:
  - ***BinaryBits*** after Objective 4.
  - ***UnhexPumpkins*** after Objective 5.
  - ***KitchenDoorDash*** after Objective 6.
- Record the score for the mission, found in the class dashboard.
- Write a paragraph about their coding experience and what they learned.

Success Criteria

- ☐ Turn on and off a string of LEDs using binary (0b)
- ☐ Evaluate an expression that has a logical operator
- ☐ Turn on and off a string of LEDs using hexadecimal (0x)
- ☐ Evaluate an expression that uses binary numbers and a bitwise operator
- ☐ Evaluate an expression that uses hexadecimal numbers and a bitwise operator
- ☐ Read a line sensor and use the value to unhex a pumpkin

Vocabulary

- **Binary:** Two digits – 0 and 1; used to represent OFF and ON or True / False.
- **Boolean:** The data type for binary numbers; named after the mathematician George Boole.
- **Logical Operators:** And and Or; enables programmers to handle multiple conditions.
- **Bit:** A single binary digit – is either 0 or 1.
- **Bitwise Operator:** Used for individual bits: & (and) and | (or)
- **Hexadecimal:** A number system with base 16.
- **Byte:** A collection of 8 bits.

New Python Code

`leds.user(0b10000101)`	Uses binary to turn on LEDs 0, 2 and 7, and turn off all other LEDs.
`leds.user(0x1A)`	Uses hexadecimal to turn on LEDs 1, 3 and 4 (1A = 0001 1010)
`0b10000101 & 0b01110011` `0x1A & 0x56`	Bitwise AND operator
`0b10000101 | 0b011110011` `0x1A | 0x56`	Bitwise OR operator
`line_senor = ls.read(2)`	Read line sensor #2.

CodeBot: Haunted Code Chronicles

CSTA Standards

Grades 3-5	Grades 6-8	Grades 9-10	Grades 11-12
• 1B-CS-02 • 1B-CS-03 • 1B-AP-08 • 1B-AP-10 • 1B-AP-11 • 1B-AP-12 • 1B-AP-15	• 2-CS-02 • 2-CS-03 • 2-DA-07 • 2-AP-10 • 2-AP-11 • 2-AP-12 • 2-AP-13 • (2-AP-14)	• 3A-CS-03 • 3A-DA-09 • 3A-AP-13 • 3A-AP-16 • (3A-AP-17) • (3A-AP-18)	• 3B-CS-02 • 3B-DA-06 • 3B-AP-10 • (3B-AP-14) • 3B-AP-16

Teacher Notes:

- This message introduces and uses binary and hexadecimal numbers. If you want additional practice, try the Binary Lesson.
- Objective 5 is a difficult challenge. Read the tips and tricks for the objective below to help your students complete the challenge.

Additional Resources / Extensions:

- Mission 3 Activity Guide
- Mission 3 Activity Guide Answer Key
- Optional: Binary Lesson
- Supports **language arts** through reading instructions and reflection writing.

Preparing for the lesson:

- Go through the Mission in advance, if you can. Try at least the first couple of objectives.
- Be familiar with using binary and hexadecimal numbers to control the user and line sensor LEDs.
- Objective 1 gives a quick lesson on binary. Look it over to see if you need to do any additional front-loading or if the instructions are enough.

Lesson Tips and Tricks:

👫 Warm-up:

The warm-up has two questions. Have students share their answers as a class discussion or think-pair-share. Go over the mission introduction to help them understand what will be expected during this mission.

💻 Mission 3 Objectives:

Objective #1

This objective is a short lesson on binary numbers and how they can be used to turn on user LEDs. For more information, you can have students click on “binary” in the Objective Panel to add it to their toolbox and read about binary numbers. To use binary numbers when controlling LEDs, use the prefix `0b`

You can also use the same technique to turn on the line sensor LEDs. The only difference is there are five of them instead of eight. Example: `leds.ls(0b00110)`

Objective #2

This objective shows a truth table for the logical operators OR and AND. You may need to review the table with your students. Students will not start a new file for the program. They can delete all the practice code except the import statements and then add new code below. If a student clicks “Reset”, they will get a new problem.

There are many ways to meet the goal. Students can use code from Mission 1 to turn on or off each LED individually. They can use binary to turn on or off the LEDs with a single line of code. For the highest score, they need to incorporate the logical operator in the code somewhere, but it is not required.
For example: answer = True or False.

Objective #3

Just like Objective 2, there are many ways to meet this goal. CodeTrek shows one way, which is to solve the problem on your own, and then write code to turn on/off the LEDs. The hints show another way, with a higher score. It uses a variable for each number and then the bitwise operator. The third hint goes into some detail on how the bitwise operators work. It is optional. The bitwise operators are | (or) and & (and).

Objective #4

This challenge uses hexadecimal numbers. The Hint gives a table of hex numbers. You can use it to convert the numbers to binary and then go from there. If you don't want to do any conversion, your students can use the second Hint, similar to the Hint for Objective #3, and use the hexadecimal numbers to solve the problem and turn on/off the LEDs. This will be the most useful way to do it, since further objectives will have similar challenges. To use hexadecimal numbers when controlling LEDs, use the prefix `0x`. Here is a sample code:

```
a = 0x26
b = 0xD5
leds.user(a | b)
```

Objective #5

This is one of the harder challenges in the mission pack. One difficulty is getting out from under the table. The activity guide encourages students to have a plan. Here are some tips to help complete the challenge:

- CodeBot can go under the chairs. This could be the easiest way to get out from under the table.
- You can keep all the codes as hexadecimal. There are a few instructions that mention changing the codes to binary, but that is not necessary.
- The code for moving CodeBot does perform differently from run to run. If it runs correctly once or twice, and then doesn't run correctly, that doesn't mean you need to change the code. Just keep running the code and it will perform correctly again.
- Try using different camera angles. Universal is good for seeing the entire room, but Rotate and Chase are good for seeing what CodeBot sees. You can switch back and forth between camera angles.
- If your students are comfortable with functions, this is a good time to use them. You can define a function for moving the 'bot and another function for unhexing the pumpkin, and pass as a parameter the code above the pumpkin. If students don't want to do a function for unhexing, they can just type the three lines of code needed to unhex the pumpkin, filling in the correct code for the pumpkin.
- You don't have to go to all four pumpkins. Any three will do. You can even avoid the pumpkin under the table all together if you want.

Objective #6

The objective can be completed on many different levels. For beginners, they just need to go through the door. The code can be just like the one used to get off the porch (Mission 1 Objective 13). If students take time to munch several snacks, this objective can take a lot of time for trial and error.

- For a higher score, do the same thing as ***DoorDash*** but use a function.
- For bonus scoring, travel around the kitchen munching on treats (with or without a function).

🧑‍🤝‍🧑 **Reflection:**

The reflection questions can be written, or shared as a class discussion, or through a think-pair-share activity. This is an opportunity to promote a growth mindset.

CodeBot: Haunted Code Chronicles

MISSION 4: The Hallway

Time Frame: 45-90 minutes

Project Goal: Students will use line sensor readings to follow a dark line on the floor to reach a doorway.

Learning Targets

- I can read the line sensors and return their values.
- I can print a value to the console.
- I can use a line sensor reading in a condition to control CodeBot.

Key Concepts

- CodeBot has five line sensors. Each one returns a reading of light reflectivity.
- All five sensors can be read at the same time and compared to a threshold. A tuple of five Booleans is returned.
- A tuple is a read-only version of a list. You can use the values but not change them.

Assessment Opportunities

- Complete the Mission 4 Activity Guide.
- Complete and turn the program:
  - ***HallwayStains*** after Objective 4.
- Record the score for the mission, found in the class dashboard.
- Write a paragraph about their coding experience and what they learned.

Success Criteria

- ☐ Read a single line sensor and print the value to the Console
- ☐ Determine a threshold value for line sensor readings
- ☐ Use a line sensor reading and threshold in an if statement
- ☐ Read all five line sensors at the same time and return a tuple of five Booleans
- ☐ Use the algorithm for a bang bang controller to guide CodeBot along a dark line to the door

Vocabulary

- **Line Sensors:** Photo reflective sensors that can detect how much light is reflected by the surface the 'bot is on.
- **Tuple:** A list that doesn't change its size. For example: (False, False, True, False, False).
- **Index:** A number, starting with 0, used to access a value in a tuple or list.
- **Constant:** A variable that doesn't change its value during program execution; usually defined in ALL CAPS.

New Python Code

`val = ls.read(2)`	Read line sensor #2
`print(val)`	Print the line sensor reading on the console
`ls.check(threshold)`	Check all five line sensors and return a tuple of five Boolean values
`if middle:`	An if statement that uses a Boolean value, returned by the line sensor reading, to select the branch of code.
`elif vals[4]:`	An additional branch for an if statement. The condition for the elif (short for else if) is evaluated, and if true, the code is executed.

CSTA Standards

Grades 3-5	Grades 6-8	Grades 9-10	Grades 11-12
• 1B-CS-02 • 1B-CS-03 • 1B-DA-07 • 1B-AP-09 • 1B-AP-10 • 1B-AP-15	• 2-CS-02 • 2-CS-03 • 2-DA-08 • 2-AP-10 • 2-AP-13 • 2-AP-12	• 3A-CS-03 • 3A-DA-12 • 3A-AP-13 • 3A-AP-14 • 3A-AP-16	• 3B-CS-02 • 3B-DA-06 • 3B-AP-10 • 3B-AP-16

CodeBot: Haunted Code Chronicles

Teacher Notes:	**Additional Resources / Extensions:**
• Students will do all their coding in one program. The code from the first three objectives will be deleted and replaced with code copied from the instructions for Objective 4. If students want to keep their code from objectives 1-3, they need to either start a new file for Obj 4 or do a “Save As”. • Subtle changes are made in their code during objectives 1-3. Students need to pay attention to CodeTrek to avoid syntax and typing errors.	• Mission 4 Activity Guide • Mission 4 Activity Guide Answer Key • Supports **language arts** through reading instructions and reflection writing.

Preparing for the lesson:

- Go through the Mission in advance, if you can. This is a fairly short mission.

Lesson Tips and Tricks:

👫 **Warm-up:**

The warm-up has two questions. Have students share their answers as a class discussion or think-pair-share. Objective 1 gives a quick lesson on binary. Look it over to see if you need to do any additional front-loading or if the instructions are enough for your students to continue.

Go over the mission introduction to help them understand what will be expected during this mission.

💻 **Mission 4 Objectives:**

Objective #1

This objective gives a short lesson about line sensors. The code is given in CodeTrek, with only one # TODO: Students should open the Console and observe the readings. The goal should be met with one try. If for some reason the goal isn’t met and the ‘bot goes past the red stain, just have the student do a reset and run again.

Objective #2

Students will select a value halfway between the floor reading and the spot reading to be the threshold. Then they use the threshold in an if statement to stop the ‘bot when it reaches the spot. They just need to follow CodeTrek. A suggestion for the threshold is 3300, but it can be any number that is basically in between their readings.

Objective #3

This objective uses the term tuple. Students only need to know it is a read-only version of a list. If you (or the students) want to know more, click on the tool and add it to your toolbox.

CodeTrek guides students in modifying their code. There are subtle changes that students might miss. They need to check each line of code from the current program to what is in CodeTrek. Every time the scene is reset, CodeBot will move its position slightly.

Objective #4

Students are given code for this challenge. They can click on the icon next to the code in the Objective Panel to copy the code, and then paste it in the text editor. The copied code includes imports, so all their previous code can be deleted and replaced with the new code. If students want to keep their previous code, they should do a “File ➜ Save As...” After they copy the code, they still need to follow CodeTrek.

- Beginning students do not need to read the hints. The copied code and CodeTrek, with slight modifications, will complete the challenge.

- There are two different configurations for the hallway stain. One scene uses the door on the left, and the other uses the door on the right. Both configurations have a short path and a long path, and the same code should work on both scenes.
- Students will do a lot of trial and error, which means resetting the scene multiple times. Sometimes the ‘bot will be positioned in front of the longer trail. Students can try the longer trail, or do another reset to position the ‘bot in front of the shorter trail.
- The easiest modification to make is to try a slower speed. Experiment with the speed until you find one that keeps the ‘bot on the trail.
- Students who want to try the challenges can read the hints and add more elif statements for more precise turning, and also try the longer trail.

👫 Reflection:

The reflection questions can be written, or shared as a class discussion, or through a think-pair-share activity. This is an opportunity to promote a growth mindset.

MISSION 5: The Library

Time Frame: 45-90 minutes

Project Goal: Students will use an equation with modulo to turn on lamps in a pattern and use an accelerometer to drive up a ramp.

Learning Targets

- I can label the parts of a division problem.
- I can use modulo in an equation.
- I can use the int() function in an equation.
- I can read the accelerometer.
- I can use an accelerometer reading to keep CodeBot moving up when its surroundings are turning.

Key Concepts

- Programmers can use different types of division. In addition to float (or decimal), there is also integer division and modulo.
- Modulo returns the remainder only of a division problem.
- CodeBot has an accelerometer that tracks its position through gravitational acceleration in three dimensions. This allows the 'bot to detect motion and also use the values for navigation.
- The accelerometer returns three values: x, y and z. The values are returned in a tuple, which can be unpacked for easy access.

Assessment Opportunities

- Complete the Mission 5 Activity Guide.
- Complete and turn the programs:
  - ***BookBox*** after Objective 1.
  - ***Lamplighter*** after Objective 4.
  - ***BookshelfRamp*** after Objective 6.
- Record the score for the mission, found in the class dashboard.
- Write a paragraph about their coding experience and what they learned.

Success Criteria

- ☐ Move the 'bot to knock a box off a book
- ☐ Turn on a line sensor LED to turn on a lamp
- ☐ Use a loop to light all lamps in a sequence
- ☐ Use an equation with modulo to light all lamps in a sequence
- ☐ Drive CodeBot straight up a fallen shelf
- ☐ Drive CodeBot straight up a fallen shelf using an accelerometer reading

Vocabulary

- **Epoch Time:** The number of seconds since Jan. 1, 1970, a frame of reference computers use to count time.
- **Floor:** Truncating a value to make it a whole number; like a quotient.
- **Modulo:** The remainder of a division problem; use the symbol %
- **Integer Division:** Division that gives the quotient only; use the symbol //
- **Accelerometer:** A sensor that detects motion and gravitational acceleration.

New Python Code

`leds.ls_num(2, True)`	Turn on line sensor 2 LED
`time()`	A function that returns the running count of seconds
`int(3.14) = 3`	The int() function truncates a value to a whole number
`17 % 5 = 2`	Modulo (remainder only)
`light = (int(time() * 3) % 5`	Equation for spectral dance, using modulo and int()
`x, y, z = accel.read()`	Read the accelerometer and unpack its three values.
`motors.run(LEFT, 20)` `motors.run(RIGHT, 30)`	Steer left
`motors.run(LEFT, 30)` `motors.run(RIGHT, 20)`	Steer right

CodeBot: Haunted Code Chronicles

CSTA Standards

Grades 3-5	Grades 6-8	Grades 9-10	Grades 11-12
• 1B-CS-02 • 1B-CS-03 • 1B-DA-07 • 1B-AP-09 • 1B-AP-10 • 1B-AP-15	• 2-CS-02 • 2-CS-03 • 2-DA-08 • 2-AP-10 • 2-AP-11 • 2-AP-12	• 3A-CS-03 • 3A-DA-12 • 3A-AP-13 • 3A-AP-14 • 3A-AP-16	• 3B-CS-02 • 3B-DA-06 • 3B-AP-10 • 3B-AP-16

Teacher Notes:

- There are two new concepts in this mission: modulo (objectives 1-4) and using the accelerometer (objectives 5-6).
- If you would like to give your students more practice with types of division, you can use the Types of Division unplugged lesson.

Additional Resources / Extensions:

- Mission 5 Activity Guide
- Mission 5 Activity Guide Answer Key
- Optional: Types of Division unplugged lesson
- Supports **language arts** through reading instructions and reflection writing.

Lesson Tips and Tricks:

👫 Warm-up:

This mission uses modulo. The warm-up questions prepare students for this type of division. You can model long division for your students and do a whole class review on the terms. For more instruction, use the unplugged Types of Division lesson.

💻 Mission 5 Objectives:

Objective #1

No new concepts in this objective. Students move the ‘bot to knock off the box. The ‘bot can still drive unpredictably. If the code looks good but the goal isn’t met, just try running the code again.

The completion message for Objective #1 has the tool “Modulo”. Students need to click on the term and read about it to answer a question for this objective.

Objective #2

This objective has three parts. Two of them are mentioned in the instructions: try each lamp individually and try all five lamps at the same time. The third part is in CodeTrek.

Objective #3

Students follow CodeTrek to complete the challenge. It is an intermediate step to understanding the next objective. You may need to remind students that don’t type **# TODO:**, but type the actual code instead.

Objective #4

Students are given the equation for Hopper’s dance. The instructions break down the equation into its parts. CodeTrek gives a lot of help. There are subtle changes in CodeTrek students may not notice. For example, the from time import statement adds time. The light variable is deleted. Make sure students are paying careful attention.

The Hints go over modulo and integer division and give examples. The third Hint has the actual code for the equation. Students need to be very careful with parenthesis when typing the equation. Even one misplaced parenthesis can cause the program to fail.

Objective #5

Students follow CodeTrek to complete the challenge. It is an intermediate step to understanding the next objective. You may need to remind students that don't type **# TODO:**, but type the actual code instead.

Objective #6

Students will add a while True: loop to their code. All the code inside the loop must be indented!

The second hint gives a lot of help with the if statement. It shows an if / else statement. If students use an if / elif / else statement they will get a higher score. A function can be used, but it doesn't affect the score.

👫 **Reflection:**

The reflection questions can be written, or shared as a class discussion, or through a think-pair-share activity. This is an opportunity to think about applications of the new concepts, outside of the challenge.

CodeBot: Haunted Code Chronicles

MISSION 6: The Attic

Time Frame: 45-120 minutes

Project Goal: Students will use encryption and conversion techniques to solve puzzles.

Learning Targets

- I use a loop to sweep LEDs.
- I can use the slope-intercept equation to encrypt a message and display the results using LEDs.
- I can convert letters to Morse Code using CodeBot's speaker.
- I can convert letters to ASCII Code and display the numbers on LEDs.

Key Concepts

- Two of the challenges use codes to convert data: Morse Code and ASCII Codes. This is called character encoding. Documents with the codes are on the floor of the attic.
- Computers represent data numerically using codes. ASCII, the American Standard Code for Information Interchange, is one of the oldest codes still in use today. Each symbol on the keyboard can be converted to a number.
- Symmetric encryption uses a single shared key to both encrypt and decrypt data.

Assessment Opportunities

- Complete the Mission 6 Activity Guide.
- Complete and turn the programs:
  - ***Hasselhoff*** after Objective 1.
  - ***TuringCar*** after Objective 2.
  - ***AtticTrinkets*** after Objective 3.
  - ***OpenTrunk*** after Objective 4.
  - ***ASCIISpell*** after Objective 5.
- Record the score for the mission, found in the class dashboard.
- Write a paragraph about their coding experience and what they learned.

Success Criteria

- ☐ Sweep the user LEDs back and forth
- ☐ Use the equation for a line to encrypt a 4-digit message
- ☐ Display an integer using line sensor LEDs
- ☐ Convert letters to speaker beeps using Morse code
- ☐ Convert letters to ASCII values and then apply an algorithm to shift them
- ☐ Display numbers using binary on user LEDs

Vocabulary

- **Encryption:** The process of converting information or data into a code to prevent unauthorized access.
- **Symmetric Encryption:** An encryption technique that uses a single shared key to encrypt and decrypt data.
- **Morse Code:** A way to code letters with dots and dashes.
- **Character Encoding:** A mapping of characters, such as letters and symbols, into numbers.
- **ASCII:** American Standard Code for Information Interchange; a form of character encoding.
- **List:** A type of variable that can hold more than one value, with items in a specific order that are accessed by using an index.
- **Index:** The position of an item in a list; used to access the item. The first index is 0.

New Python Code

`leds.ls(y)` `leds.ls(10)`	Uses an integer (y) to turn on line sensor LEDs Turns on line sensors 3 and 1 because 10 = 0b01010
`ord()  /  ord('A')`	A function that converts a string to its ordinal (ASCII) value
`shift = shift + 2`	Increase a variable by 2
`names = ["George", "Grace", "Alan"]` `numbers = [2, 0, 2, 4]`	Define a list of strings. Define a list of numbers.
`name = names[1]  # "Grace"` `num = numbers[0]  # 2`	Access an item using an index.

`morse = {`
`  "a": ".-",`
`  "b": "-...",`
`  "t": "-",`
`}`
	Define a dictionary with keys.

`for letter in word:` `for symbol in code:`	Iterate through a dictionary or list.

CSTA Standards

Grades 3-5	Grades 6-8	Grades 9-10	Grades 11-12
• 1B-CS-02 • 1B-CS-03 • 1B-DA-06 • 1B-DA-07 • 1B-AP-09 • 1B-AP-10 • 1B-AP-11 • 1B-AP-12 • 1B-AP-15	• 2-CS-02 • 2-CS-03 • 2-DA-07 • 2-DA-08 • 2-AP-10 • 2-AP-11 • 2-AP-12 • 2-AP-13 • 2-AP-14	• 3A-CS-03 • 3A-DA-09 • 3A-DA-12 • 3A-AP-13 • 3A-AP-14 • 3A-AP-16 • 3A-AP-17 • 3A-AP-18	• 3B-CS-02 • 3B-DA-06 • 3B-AP-10 • 3B-AP-14 • 3B-AP-16

Teacher Notes:

- Each of the challenges can be solved using brute force, meaning you don't need any for loops or lists or functions. So even beginners can complete the goals by repeating a lot of code. More experienced programmers should use programming concepts like loops, lists, functions, and parameters. Hints provide ideas and some code.

Additional Resources / Extensions:

- Mission 6 Activity Guide
- Mission 6 Activity Guide Answer Key
- Knight Rider trailer
- Binary bracelets
- Binary bracelets
- Supports **language arts** through reading instructions and reflection writing.

Lesson Tips and Tricks:

👫 **Warm-up:**

The warm-up questions help them think about how computers store data, and the need for data security. It is a good time for discussion. You can even circle back to these questions during or after the mission pack.

💻 **Mission 6 Objectives:**

Objective #1

This objective references the 1980s show Knight Rider and the car KITT. If you want to give students some context, you can show the trailer (linked above).

Objective #2

The challenge uses the algebraic equation for a line to encrypt a 4-letter message. Each digit in the numeric code 2024 is used as the four values for x. The same m values are the 4 digits in Turing's birth year. The b values are the 4 digits in Turing's death year. Each code is displayed on the line sensor LEDs for 2 seconds.

CodeTrek shows how to do the first digit. This “brute force” code works fine. Students can continue the code for the other three digits. For a higher score, students can either define a function with a parameter and call it four times, or use a list and loop.

Objective #3

The challenge for this objective is similar to other challenges: move CodeBot around the floor and collect trinkets. There is no time limit, so you may need to monitor student progress and not give them too much time.

This objective is a good opportunity for a function to drive CodeBot. That is all the code needed to meet the goal. CodeBot needs to end up near the window but not crash into the wall.

Objective #4

Students first find the pocket signal disk, which is by the two lit candles. The disk has morse code for each letter. The note indicates the keyword is “debug”. Students use the pocket signal disk to find the morse code for each letter. CodeTrek shows the values for the constants: MORSE_TONE, T_DOT, T_DASH, T_GAP and T_PAUSE.

CodeTrek shows what the code looks like for a single letter. However, the letter is ‘A’, which is not in the word “debug”. So they need to adjust the code accordingly. They can continue with this pattern for all five letters, or they can try the code given by CodeBat. It uses a dictionary. If students want to use the dictionary, they need to add the letters e, d, u and g to it. They can be added in any order. They can get an even higher score by adding another function that plays a tone (either dash or dot).

Objective #5

The instructions guide students through using the ord() function to convert a letter to a number and then display the number using the user LEDs. Students should do the practice code before trying the challenge.

This challenge can be a little tricky, because if students don’t read the second hint, they will not complete the goals. The algorithm requires a shift in ASCII number. If they do the code without the shift, it should be a fairly easy fix to add it in. They subtract the shift to display the number, but increase the shift with each letter. Using a loop (and a list) works really well and gives bonus points.

🧑‍🤝‍🧑 Reflection:

The reflection questions can be written, or shared as a class discussion, or through a think-pair-share activity. This is an opportunity to think about programming in general, what they like and don’t like about it, and the skills they develop from coding.

🧑‍🤝‍🧑 Post Mission Pack:

Other computer science topics can be discussed and studied during or after this mission, particularly data security, which can meet even more CSTA standards.

A fun follow-up activity is for students to create binary bracelets using ASCII. There are many resources available that give instructions. Two are linked above. An alternative is to create binary posters. Girls Who Code has an activity for bracelets or posters.