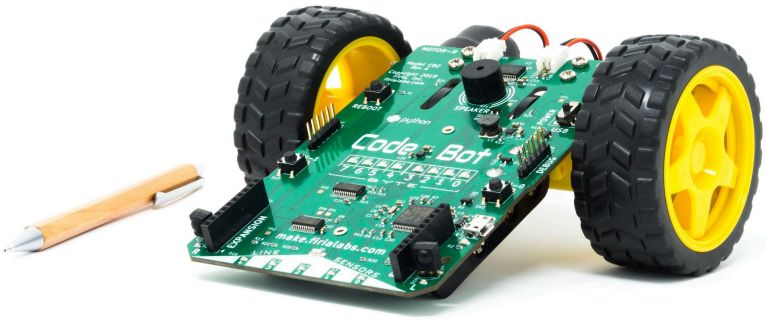
Python with Robots

Middle School Curriculum Guide



Unit 2

Animatronics Robot Application

Unit 2: Animatronics Robot Application (6-10 hours)

Students continue their programming journey by combining LED lights, movement and sound. They learn about loops and defining and calling functions. Students create three individual programs that accomplish a single task, and then combine them into one program for an animatronics robot application.

Summary of Mission 4: This mission is divided into four lessons, each approximately one class period. During the mission students are given the assignment to create an animatronic robot exhibit for a major theme park, with a sketch of the algorithm on a napkin. The tasks are divided into three lessons.

> **Lesson 1:** The first requirement for the animatronics exhibit is to flash the user lights in a cool pattern. Students use an infinite loop and a variable for the LED.
>
> **Lesson 2:** The second requirement is to count the guests. Students define and update a counter variable and learn about incrementing. They learn how to break out of a loop, and use basic speaker functions.
>
> **Lesson 3:** The remaining requirements are to move the CodeBot and play a fanfare. Students learn about a loop that repeats a specified number of times, and use random numbers to generate robot beeps. They define and call a function.
>
> **Lesson 4:** Students complete the fanfare music and then put together the three separate programs into one complete animation project.

Unit 2 Remix: Unit 2 is only four lessons from Mission 4. This is an excellent opportunity for a group project, meeting the CSTA standards for collaboration and working in teams. The project can be another animatronics program, but one that does something different than Mission 4.

> **Recommendation:** Since the lessons in this unit all involve separate programs that are then combined into one complete project, your students can devise a project similar to the animatronics project that has multiple tasks. Each student can create their own program that accomplishes one or two tasks. Then the group can combine the parts into a complete project. You can assign team roles and use multiple forms of communication and task tracking. Students can get feedback from multiple users and in various ways and then refine their project. You can emphasize accessibility and equity with the feedback and refinements.

Unit 2 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, the slides should be used instead of the instructions in CodeSpace and CodeTrek. They introduce things a little differently. 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 2 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.
- Have a ruler for measuring the distance traveled by the 'bot.

Assessment:

Mission 4 Obj 1-3 Kahoot Review	Mission 4 Obj 4-7 Kahoot Review	Mission 4 Obj 8-11 Kahoot Review
Mission 4 Obj 12 Kahoot Review		

Coming soon: Unit 2 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 2 Question Bank.

Unit 2 Vocab Kahoot Review	Unit 2 Coding Kahoot Review	Unit 2 Vocab Test (MS Form)	Unit 2 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-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-18 • 2-AP-19	• 3A-CS-01 • 3A-CS-03 • 3A-AP-13 • 3A-AP-15 • 3A-AP-16 • 3A-AP-17 • 3A-AP-18 • 3A-AP-19 • 3A-AP-21 • 3A-AP-22 • 3A-AP-23 • 3A-IC-25 • 3A-IC-27	• 3B-CS-02 • 3B-DA-06 • 3B-AP-10 • 3B-AP-14 • 3B-AP-16 • 3B-AP-17 • 3B-AP-20 • 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 4: Introducing CodeBot Lesson 1 (Objectives 1-3)	**Time Frame: 35-45 minutes**

Project Goal: Students will use an infinite loop to constantly blink the user LEDs in a cool pattern.

Learning Targets

- I can use a while True: infinite loop.
- I can increment a counter variable.
- I can use the debugger to track a counter variable.
- I can use an if statement to avoid an out of range error.
- I can reset a variable to its initial value.

Key Concepts

- While loops are used to execute an algorithm constantly.
- Increments (and decrements) are used for updating variables like counters.
- CodeBot has 8 user LEDs, numbered 0 through 7. If you try to turn on a user LED with a different number, you get an out of range error.
- The debugger and console panel can be used to track the value of a variable.

Assessment Opportunities

- Mission 4 Lesson 1 Log (digital)
- Submit completed program ***SweepLEDs***
- Submit the program with extensions
- Mission 4 Obj. 1-3 Review Kahoot!

Success Criteria

- ☐ Use an infinite loop
- ☐ Use a variable to light a user LED
- ☐ Increment the variable to light the LEDs in sequence
- ☐ Use an if statement to check if the variable is a specific number
- ☐ Reset the variable to its initial value

Teacher Materials in Learning Portal

- Mission 4 Lesson 1 Slides
- Mission 4 Lesson 1 Log
- Mission 4 Lesson 1 Answer Key

Additional Resources

- Mission 4 Obj. 1-3 Review Kahoot!
- SequenceLEDs sample code (learning portal)
- SequenceLEDs_extensions sample code (learning portal)

Vocabulary

- **Loop:** Repeating a block of code, as long as a condition is True.
- **While condition:** The Boolean value, variable or expression used in a while loop.
- **Infinite loop:** A loop that never ends because the condition is always true.
- **Updating a variable:** Assign a new value to a variable, based on the old value.
- **Increment:** Update a variable by adding one (or a specific number) to the old value.
- **Single equal (=):** Assignment symbol; used to assign a value to a variable.
- **Double equal (==):** Comparison operator used to determine if two variables or values are the same.

New Python Code

`while True:`	Infinite loop; code block must be indented directly underneath the loop.
`leds.user_num(n_led, True)`	Use a variable to indicate the LED
`n_led = n_led + 1`	Update a variable; increment
`if n_led == 8:` `  n_led = 0`	Check if a variable is the same as a specific value. If so, reset the variable to its initial value.

Real World Applications

Using loops and infinite loops are used in many real-world applications.

- Blinking traffic lights.
- Sensors that detect movement or objects need to constantly be checking.
- Robotic arms in factories repeat their actions continuously.

Teacher Notes:	**Extensions / Cross-Curricular:**
• Objective 3 uses the debugger to track the variable. This can be done together as a whole class. Use the debugger and a large display screen. Note that the loop doesn't stop at 7, there just aren't any more LEDs left. • The debugger stops when an error occurs, and the value in the debug window will no longer show. Make note of the value before it disappears. • Review the difference between = and ==. • Students should refer to their data from Mission 3 Lesson 3 to help with moving forward and turning.	• Use two variables for the delay, one for turning on the LEDs and one for turning off the LEDs. • Use one delay variable but different amounts of delay for turning on and off the LEDs. • Increment the delay variable when resetting the n_leds variable. • Use two variables for the LEDs and create a sweeping pattern with two LEDs. • **MATH:** Use math to calculate how many times an LED will blink, or how long it takes to do 20 sweeps, or what the delay should be to do a specific number of sweeps per minute. • **LANGUAGE ARTS:** Write about a real-world application that uses an infinite loop. • Supports **language arts** through reading instructions, guided notes, and reflection writing.

MISSION 4: Animatronics
Lesson 2 (Objectives 4-7)

Time Frame: 35-45 minutes

Project Goal: Students count the number of guests using a button press, displaying the count using line sensor LEDs.

Learning Targets
- I can increment a counter variable.
- I can use buttons.was_pressed() in an if statement to control program flow.
- I can display a number using line sensor LEDs.
- I can break a loop.
- I can add a beep to a program.
- I can debounce a button press.

Key Concepts
- Increments (and decrements) are used for updating variables like counters.
- Button presses (inputs), LEDs (outputs) and speaker sounds (outputs) are part of the user interface. They allow the user to interact with the CodeBot.
- A button press can "bounce," or detect an additional press that didn't happen. Using an extra buttons.was_pressed() after a short delay can debounce a button press.
- A number can be displayed by using LEDs.

Assessment Opportunities
- Mission 4 Lesson 2 Log (digital)
- Submit completed program ***CountGuests***
- Submit the program with extensions
- Mission 4 Obj. 4-7 Review Kahoot!

Success Criteria
- ☐ Increment a variable to count a button press
- ☐ Use a variable to turn on a line sensor LED
- ☐ Add a beep when a button is pressed
- ☐ Debounce a button press

Teacher Materials in Learning Portal
- Mission 4 Lesson 2 Slides
- Mission 4 Lesson 2 Log
- Mission 4 Lesson 2 Answer Key

Additional Resources
- Mission 4 Obj. 4-7 Review Kahoot!
- CountGuests sample code (learning portal)
- CountGuests_ext sample code (learning portal)

Vocabulary
- **Break:** Exit the nearest enclosing loop.
- **Increment:** Increase by one.
- **Debounce:** Reset the internal status of a button so the press isn't counted twice.

New Python Code

`break`	Break out of a loop
`leds.ls_num(n_guests, True)`	Turn on a line sensor LED, using a variable to indicate which LED
`n_guests = n_guests + 1`	Increment
`spkr.pitch(440)`	Play a tone on the speaker; the argument is the pitch frequency in Hertz
`spkr.off()`	Turn off the speaker (usually after a short delay)
`buttons.was_pressed(0)`	Debounce a button press by resetting the internal status (after a delay)

Real World Applications
Computers and sensors are used in many real world applications, especially to automate tasks, improve efficiency and gather data for better decision making. Some examples are:
- In manufacturing, like 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

Teacher Notes:

- RECOMMENDATION: Use the slides instead of instructions in CodeSpace and CodeTrek. This mission is divided into 4 lessons that cover all the information, but chunked more. All goals will still be met, but students are asked to create a new program and only focus on the new material. The programs will be combined in Lesson 4. If you choose to use the instructions in CodeSpace and the CodeTrek, then don't use the slides.
- On Objective 5, students test the program by pressing the button and lighting up line sensor LEDs. They may find that a single press lights up more than one LED. I recommend bringing this up. Then see if the same thing happens for Obj. 6. This action is called "bouncing." Students learn about and fix this in Obj. 7.
- The quiz after Obj. 7 includes a question that may be difficult. The third question has a while loop with the condition i < 3. In this problem, the i variable is incremented outside the loop. It never changes, so the loop becomes infinite.

Extensions / Cross-Curricular:

- Use the user LEDs to display the number of guests. Go up to 8 guests before breaking the loop.
- Use the user LEDs to display the number of guests. Get a random number between 5 and 8 to break the loop, so the number of guests can be different.
- Change the tone of the beep with every button press by adding another variable.
- **MATH:** When programming, you can increment a variable with something other than 1. Incrementing by 2 can give you all the even numbers. Incrementing by other values is like skip counting. Practice skip counting, and then think about what the code would look like. What would the initial value be?
- **LANGUAGE ARTS:** Write about a real-world application that counts things using some kind of sensor or input. (See real world applications above.)
- **SCIENCE:** Learn about the mechanics of a button.
- Supports **language arts** through reading instructions, guided notes, and reflection writing.

MISSION 4: Animatronics **Lesson 3 (Objectives 8-11)**	**Time Frame: 45-55 minutes**

Project Goal: Students create a program with movement, random beeps, and a function that plays a note.

Learning Targets

- I can move the 'bot forward a specific distance.
- I can spin the 'bot in a full circle.
- I can import the random module.
- I can generate a random integer between a start and stop value.
- I can use the random integer in a program.
- I can use a while loop that executes a specific number of times.
- I can nest a while loop inside another while loop.
- I can define and call a function with parameters.

Key Concepts

- Python's random library makes it easy to work with random numbers. There are several functions in the module, but this program will use only one.
- The randrange() function has two parameters: start and stop. When generating a random integer, the lowest integer possible is the start number, but the highest possible value is one less than the stop number.
- A function is a named chunk of code you can run anytime by calling its name. A function can be called multiple times in a single program.
- A function definition and function call always include ().

Assessment Opportunities

- Mission 4 Lesson 3 Log (digital)
- Submit completed program ***RobotMoves***
- Submit the program with extensions
- Mission 4 Obj. 8-12 Review Kahoot!

Success Criteria

- ☐ Move the 'bot forward a specified distance
- ☐ Spin the 'bot in a complete circle
- ☐ Use a while loop that executes a specific number of times
- ☐ Generate a random integer
- ☐ Use the random integer as an argument for pitch
- ☐ Define a function for playing a note
- ☐ Call the function for playing a note

Teacher Materials in Learning Portal

- Mission 4 Lesson 3 Slides
- Mission 4 Lesson 3 Log
- Mission 4 Lesson 3 Answer Key

Additional Resources

- Mission 4 Obj. 8-12 Review Kahoot!
- RobotMoves sample code (learning portal)
- No code for extensions is given

Vocabulary

- **While loop:** Exit the nearest enclosing loop.
- **Random number:** An integer generated using a function from the Python random module.
- **Function:** A named chunk of code you can run anytime just by calling its name; lets you reuse code without retyping or copy/paste.
- **Parameter:** A variable that gets its value when the function is called.

New Python Code

`while f < 1000:`	The basic structure of a while loop.

`count = 0`
`while count < 10:`
`  count = count + 1`
	A while loop starts a block of code, so all commands that are to be repeated must be indented below (inside) the loop. A loop has a control variable, which must be initialized before the loop starts and incremented inside the loop.

`from random import randrange`	Import the randrange() function from the random module (library).
`f = randrange(100, 1000)`	Generate a random integer within a given range. Possible numbers include the first value up to one less than the second value.

`def flashLEDs():`
`  {indented block of code}`
`def note(freq, duration):`
`  spkr.pitch(freq)`
`  sleep(duration)`
`  spkr.off()`
`  sleep(0.05)`
	Define a function. A function definition always uses (), even if there are no parameters. All code to be included in the function is indented inside the function definition.

`flashLEDs()` `note(349, 0.4)`	Function call. A function call uses the name and (), but not the "def". If the function has parameters, the function call includes arguments, or values, that get passed to the parameters.

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:

- RECOMMENDATION: Use the slides instead of instructions in CodeSpace and CodeTrek. All goals will still be met, but students are asked to create a new program and only focus on the new material. The programs will be combined in Lesson 4. If you choose to use the instructions in CodeSpace and the CodeTrek, then don't use the slides.
- Students should refer to their data from Mission 3 Lesson 3 to help with moving forward and turning.
- Start a new program for this lesson. The code will look shorter, and slightly different than in CodeTrek, but all goals will be met.

Extensions / Cross-Curricular:

- Add additional movement to the robot, such as moving forward or backward, or spinning again.
- Have the 'bot return to its starting position after it spins.
- Turn on a user LED when a note plays.
- **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 4: Animatronics
Lesson 4 (Objective 12)

Time Frame: 40-45 minutes

Project Goal: Students play a fanfare and combine three programs into one complete project.

Learning Targets

- I can define a constant.
- I can use a constant as an argument in a function call.
- I can call a function several times, with different arguments.
- I can combine code segments from one program with code from another program.

Key Concepts

- Python doesn't have a specific designation for a constant; it is just another variable. But you can visually distinguish a constant from a variable by using ALL CAPS for its name. This can help you remember not to update the value during the program execution.
- CodeSpace allows for several programs to be open at the same time. Using the text editor, it is easy to add code from one program into another program using copy/paste.

Assessment Opportunities

- Mission 4 Lesson 4 Log (digital)
- Submit completed program ***Animatronics***
- Submit the program with extensions
- Mission 4 Obj. 12 Review Kahoot!

Success Criteria

- ☐ Define a constant
- ☐ Use a constant as an argument in a function call
- ☐ Call the note() function multiple times to play a song
- ☐ Combine parts of three programs into one complete project that accomplishes all the tasks on the napkin sketch

Teacher Materials in Learning Portal

- Mission 4 Lesson 4 Slides
- Mission 4 Lesson 4 Log
- Mission 4 Lesson 4 Answer Key

Additional Resources

- Mission 4 Obj. 12 Review Kahoot!
- Animatronics sample code (learning portal)
- No code for extensions is given

Vocabulary

- **Constant:** A variable that holds a value that doesn't change during program execution; often designated by typing the name in ALL CAPS.

New Python Code

`F4 = 349` `C5 = 523`	Define a constant (usually with ALL CAPS)
`note(F4, 0.1)` `note(C5, 0.8)`	Use a constant as an argument in a function call

Real World Applications

Constants are used all the time in real-world applications. Some examples are:

- Minimum and maximum values
- A constant temperature to be maintained
- A specific number to be reached

This program also uses parts of different programs and combines them into a complete project. This happens in real life all the time. Can you think of a time when you combined different parts to make something new?

Teacher Notes:

- RECOMMENDATION: Use the slides instead of instructions in CodeSpace and CodeTrek. This is the final lesson that combines the previous programs into one complete project. The final

Extensions / Cross-Curricular:

- Students can use extension ideas from lessons 1-3 or the end-of-mission remix suggestions. A remix project follows this lesson, and that is also an opportunity for extensions.

code works the same but is in a different order from CodeTrek. Just use the slides or CodeTrek, but not both. • Students will not start a new program for this lesson. They start with the last program (RobotMoves) and save it with a new name.	• **PERFORMING ARTS:** This program uses a function to play a short fanfare. Have a lesson on reading sheet music. Write a program that uses constants for the tones and plays a song. • Supports **language arts** through reading instructions, guided notes, and reflection writing.

Unit 2 Remix Project	**Time Frame: 3-5 hours**
Project Goal: Students will use the skills and concepts they learned in the unit to work as a cooperative team to create their own project. **Learning Targets** • I can summarize the programming concepts from Mission 4. • I can plan an original program. • I can create an original program using concepts and code from previous programs. • I can work cooperatively in a team. • 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 2 Remix Planning Guide • Unit 2 Remix Team Planning & Review Guide • Unit 2 Remix Project	**Success Criteria** ☐ Plan an original program ☐ Work collaboratively in a team ☐ Create an original program ☐ Incorporate feedback in a program
Teacher Materials in Learning Portal • Unit 2 Remix Project slides • Unit 2 Remix Planning Guide • Unit 2 Remix Team Planning & Review Guide	**Additional Resources** • Students can use their previous programs as a guide throughout this project.

Teacher Notes:

- A remix for this unit is optional. Unit 2 is very short, and a remix is an excellent opportunity for students to create their own original program by doing something that interests them.
- This remix is planned as a team project with students working collaboratively and with a timeline. The team plans the project, and then each student works independently on their own program that accomplishes a single task. The team combines code for a complete project. This follows the steps they use in Unit 2 throughout Mission 4.
- Suggested team size is 3 or 4 students for this project. Collaboration is an important skill and a team project meets several CSTA standards.
- 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.
- Two planning guides are provided. Students start with an individual planning guide 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. The second is a team planning guide. Once each individual program is ready, team members will work together to combine their code.
- Students will need a way to share their code. There are many ways to do this. They can email the code to each other, download the code to a flash drive, use the class LMS, copy to a shared Word document, etc.
- In the team planning guide, each team is asked to get more peer reviews. They can ask other students or adults in the class, or you might want to bring in a different class and have those students review projects to get a different perspective.
- A new question is asked on the second peer review. This is about accessibility. A CSTA standard requires students to think about the impact of technology and how it can be improved to help all people.
- 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 or video about the project, etc.
- A checklist for the remix project is below.

Remix Project Checklist:

- [ ] Filename is descriptive
- [ ] Uses one or more variables, each with a descriptive name
- [ ] Uses one or more constants, each with a descriptive name
- [ ] Moves the CodeBot forward and/or backward at least once
- [ ] Rotates the CodeBot at least once
- [ ] Uses a sleep delay at least once
- [ ] Turns on at least one LED light
- [ ] Uses at least one button for input
- [ ] Uses the CodeBot speaker
- [ ] Defines and calls at least one function
- [ ] Modify program based on user feedback
- [ ] Includes comments and blank lines for readability
- [ ] Code runs without errors