| **MISSION 4: Animatronics**<br>**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.<br><br>**Learning Targets**<br>• I can move the 'bot forward a specific distance.<br>• I can spin the 'bot in a full circle.<br>• I can import the random module.<br>• I can generate a random integer between a start and stop value.<br>• I can use the random integer in a program.<br>• I can use a while loop that executes a specific number of times.<br>• I can nest a while loop inside another while loop.<br>• I can define and call a function with parameters. | **Key Concepts**<br>• 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.<br>• 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.<br>• 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.<br>• A function definition and function call always include (). |
| **Assessment Opportunities**<br>• Mission 4 Lesson 3 Log (digital)<br>• Submit completed program ***RobotMoves***<br>• Submit the program with extensions<br>• Mission 4 Obj. 8-12 Review Kahoot! | **Success Criteria**<br>☐ Move the 'bot forward a specified distance<br>☐ Spin the 'bot in a complete circle<br>☐ Use a while loop that executes a specific number of times<br>☐ Generate a random integer<br>☐ Use the random integer as an argument for pitch<br>☐ Define a function for playing a note<br>☐ Call the function for playing a note |
| **Teacher Materials in Learning Portal**<br>• Mission 4 Lesson 3 Slides<br>• Mission 4 Lesson 3 Log<br>• Mission 4 Lesson 3 Answer Key | **Additional Resources**<br>• Mission 4 Obj. 8-12 Review Kahoot!<br>• ***RobotMoves*** sample code (learning portal)<br>• 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`<br>`while count < 10:`<br>`  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():`<br>`  {indented block of code}`<br>`def note(freq, duration):`<br>`  spkr.pitch(freq)`<br>`  sleep(duration)`<br>`  spkr.off()`<br>`  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()`<br>`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.

**Preparing for the lesson:**

- Have the assignment from Mission 3 Lesson 3 available for students to use.
- Look through the slides and workbook. 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. The workbook (if used) can be printed or remain digital through your LMS and given to students.
- Be familiar with the mission log assignment and the questions they will answer. Prepare the assignment to give through your LMS.
- Look over the example code for Objectives 9 and 10.
- 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. You can have students write their answer and share with someone, and then have other students share answers they heard from their peers. You can randomly select students to answer.

👫 **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. These warm-up questions review code from Mission 4 Lesson 1. Students can share their answers, or compare with each other.

- Question: What lines of code play a beep?
- Question: What lines of code debounce a button?

💻 **Mission 4 Lesson 3 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.

💡 **Teaching tip: Mission Introduction** -- slides 3-4

This mission is divided up into four lessons. The third lesson completes most of the project, all except the complete fanfare. These slides review the task and goal. No questions in the mission log.

💡 **Teaching tip: Objective #8** -- slides 5-10

Students need to use their mission log from Mission 3 Lesson 3. They recorded data about the distance a ‘bot moves at given speeds and durations. Also data on spinning left and right. Students start a new file. This isn’t part of the instructions in CodeSpace or CodeTrek. But this way students can focus on the new concepts and not worry about a program getting too long or overly complicated.

The program starts with a way to pause program execution until a button is pressed (slide 8). It is very similar to their work in Mission 4 Lesson 2. Review if needed.

The code for moving forward and spinning is not given (slide 10). Students should use their data from Mission 3 Lesson 3 to move and spin. They can also try different numbers until they get the results they want. But they can work faster and with less frustration if they use their data.

💡 **Teaching tip: Objective #9** -- slides 11-19

Students add code near the top of their program so they can hear the beeps without moving the ‘bot. They do not need to press BTN-0 for the remainder of the lesson.

Students learn about while loops that execute a specific number of times instead of infinitely. A few slides discuss this type of while loop and give examples. Students practice identifying parts of a while loop in their mission log. The table in the mission log has them identify the control variable, its initial value, the specific number used to compare, and the increment. If you want to go a bit deeper, you can talk about how many times the loop will actually execute, or what the value of the control variable is when the loop stops.

Then students add code to their program, near the top. This objective has a couple of parts to it, ending with a nested while loop.

💡 **Teaching tip: Objective #10** -- slides 20-28

This objective introduces random numbers. Some slides discuss random numbers and their possible values when using randrange(). Some examples are given, and then students practice with random numbers by filling in the chart on their mission log.

When students add code to their program, they will no longer have a nested while loop. The random number replaces it. Make sure students are careful with their indenting.

The final step of the objective is to move the code from near the top to the bottom of the program. Review cut and paste with students if necessary.

💡 **Teaching tip: Objective #11** -- slides 29-36
This objective starts by having students play a single note (with code near the top of the program). Then the concept of a function is introduced.

Students take the code that plays a single note and converts it to a function. They will need to indent the current code inside the function definition. Also, use the two parameters in the code (slide 35).

When students call the function, the function call is NOT part of the function, and it should NOT be indented! (slide 36)

The program ends without completing the fanfare. The rest of the notes are programmed in Lesson 4, and all the parts of the animation program will be put together.

💡 **Teaching tip: Extensions** -- slide 37
If you have time, students can do an extension. If they do an extension, they might want to do a “File-Save As” so they have their original code, which they will use later. An extension isn’t really necessary for this lesson. No code solutions are given for the extensions.

**Optional:** 🗝️ Mission 4 Obj 8-11 Kahoot! Review.

A review Kahoot! Is available for these four objectives. You can do the Kahoot together as a class, or assign it independently.

## 👫 Post-Mission Reflection:

The post-mission reflection asks students to review the code used in the program.

- You can use an extension or cross-curricular activity as post-mission activity.
- Functions are a form of abstraction. Go over the real-world applications listed in the lesson plan. Have students come up with their own examples. You can even have students act out functions and function calls.
- End by collecting the Mission 4 Lesson 3 Log.

## 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