MISSION 3: Navigation Challenge
Lesson 4 (Objectives 9-11)

Time Frame: 40-60 minutes

Project Goal: Students will use button presses to control robot movement and follow an algorithm to move the 'bot in a square pattern.

Learning Targets

- I can check for a button press.
- I can use an if statement for branching.
- I can indent code inside a control flow.
- I can implement a safety feature in code when the CodeBot moves.
- I can write and follow an algorithm.
- I can increase code readability by using comments and blank lines.
- I can make the 'bot move in a square pattern.
- I can understand the control flow of an if statement.

Key Concepts

- The push buttons can be used to control program flow. One way to do this is by enacting a safety feature, so code is only run after a button is pressed. This will keep the 'bot from moving right when the program is run.
- Branching with if:elif:else statements controls the flow of the program.
- The colon (:) at the end of an if statement introduces a new block of code. Everything inside the block should be indented at the same level.
- An algorithm is a useful tool for planning a program. There are many ways to create an algorithm, such as pseudocode and flowcharts. In this lesson, only pseudocode is used.

Assessment Opportunities

- Mission 3 Lesson 4 Log
- Submit completed program ***NavSquare***
- Submit completed program ***WhatIf***
- Mission 3 Obj. 9-11 Review Kahoot!

Success Criteria

- [ ] Use buttons.was_pressed() in an if statement
- [ ] Implement a safety feature in code
- [ ] Plan a program using an algorithm
- [ ] Write code to drive CodeBot in a specific pattern
- [ ] Use buttons.was_pressed() in an if/elif/else statement

Teacher Materials in Learning Portal

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

Additional Resources

- ***MoveOut_safety*** sample code (learning portal)
- ***NavSquare*** sample code (learning portal)
- ***WhatIf*** sample code (learning portal)
- Mission 3 Obj. 9-11 Review Kahoot!

Vocabulary

- **Algorithm:** A list of instructions, in order, that the computer can follow to complete a task. (A precise sequence of instructions that the computer can follow exactly, one step at a time, to complete a task or solve a problem)
- **Comments:** Notes in the code about what you are doing; ignored by the computer
- **Whitespace:** Adding blank lines and space around symbols to make the code more readable
- **Control flow (branching):** Decision points in code; code takes a different branch depending on a condition
- **Condition:** A Boolean value (True or False), often the result of a comparison operator like <, >, or ==. Use an if statement, optionally followed by an elif or else, for branching
- **Indenting:** A way to structure blocks of code by offsetting a block of code four spaces; blocks of code are indented following a defining statement with a colon (:)

New Python Code

`buttons.was_pressed(0)`	Checks to see if BTN-0 was pressed; returns True or False

`leds.user(0b00011000)`
`sleep(10)`
`leds.user(0)`
`if buttons.was_pressed(0):`
`  {indented code}`
	Safety feature in code that only executes indented code if Button-0 was pressed.

`if buttons.was_pressed(0):`
`  # turn right`
`elif buttons.was_pressed(1):`
`  # turn left`
`else:`
`  # stop`
	Example of control flow, or branching, using if/elif/else.

Real World Applications

You've used some fundamental computer science and robotics principles: controlling motors with specific timing and sequencing, incorporating safety features, and programming different options, depending on conditions. This code is used in cars, robots, electric toothbrushes, and more!

Algorithms are part of our lives. We use algorithms all the time, for many of the tasks we complete each day. Have students think of a daily task, like getting ready for school or making a jump shot in basketball. Then have them break down the task into an algorithm.

Teacher Notes:

- This lesson covers objectives 9-11 in Mission 3. You do not need to use the instructions in CodeSpace. Some of the concepts are switched around and introduced in a different order, but all material is covered, and all goals for each objective will be met. You should choose to use either the CodeSpace instructions or the slides, but not both because even the code used in the slides is a little different than CodeTrek.
- Algorithms are used throughout this lesson. There are many unplugged activities for practicing algorithms, and they can be found in the learning portal under "CS Unplugged."

Extensions / Cross-Curricular:

- Suggested unplugged algorithm activities you can choose from:
  Algorithms lesson with drawings
  Algorithms lesson with LEGO
  PB&J Algorithm
  Hide and Seek
- **MATH:** Have a discussion about the algorithms used in math, and have students write algorithms for common math problems.
- Supports **language arts** through reading instructions and reflection writing.

Preparing for the lesson:

- All CodeBots will need batteries for this lesson. The 'bots will be moving around the room. Rechargeable batteries work fine.
- CodeBots will need space to move. Dedicate some floor space in your room for students to test their code.
- 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. Choose either the slides/workbook or the instructions in CodeSpace. Trying to use both could be confusing.
- Be familiar with the mission log assignment and the questions they will answer. Prepare the assignment to give through your LMS.
- Look over the unplugged activities to see if you want to use any of them. If so, when? Will you use an activity to introduce algorithms, or at the end of the lesson after they have had some practice with algorithms?

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.

- Question: How do you move CodeBot forward?
- Question: How do you rotate CodeBot clockwise?
- The questions can help students focus on the programming process and review the previous lesson.

💻 **Mission 3 Lesson 4 Activities:**

Each student will complete a Mission Log. Students could work in pairs through the lesson, or they can work individually.

Students should have access to their Mission 3 Lesson 3 Log, with the robot labs. They can use their data to help them navigate CodeBot in a square.

💡 **Teaching tip: Push Button Controls** -- slides 3-5

This information is added to Objective 9. It isn't in CodeSpace until Objective 10, but I felt this was a good time to introduce it. Students learn about checking if a button is pressed, and using this code to delay CodeBot movement until they are ready. The "safety feature" discussed here is new information but will be used in future lessons and programs so students have time to run their code and then move to a testing station on the floor without the CodeBot just taking off. They will be doing a lot of code testing for Obj. 9, so the safety feature can really save them some time and aggravation.

💡 **Teaching tip: Push Button Controls** -- slides 6-7

Help students determine a good countdown time. If they are far away from a testing station on the floor, they need more time. If they are close, they need less time. We don't want students running into each other trying to make their countdown time. The suggestion on the slides is 30 seconds. That should be plenty of time. You can reduce the amount of sleep() for your students' environment.

💡 **Teaching tip: Code with Push Buttons** -- slides 8-9

Students use the program from the last lesson (MoveOut) to try the safety feature code. They will need to indent their current code inside the if statement. A quick way to do this is to highlight the code and press <TAB>. That will indent all the highlighted code at the same time. You can model this for your students.

💡 **Teaching tip: Objective 9**-- slides 10-13

This is the beginning of the actual objective 9. The objective starts with discussing an algorithm. You may want to do an additional unplugged algorithm activity. Several suggestions are listed above.

💡 **Teaching tip: Objective 9 Activity** -- slide 14

Students write an algorithm in their mission log. An algorithm is given in CodeSpace, and you can show it to the students, or have them come up with their own.

💡 **Teaching tip: Objective 9**-- slides 15-17

These slides go over the goal for this objective – have CodeBot move in a square. It doesn't matter the length of the side or the direction of the rotations. Some concepts from earlier are introduced here: comments and white space by using blank lines. Students are also given the suggestion to "divide and conquer" by doing just one step at a time.

💡 **Teaching tip: Objective 9 Activity** -- slides 18-22
Students should work on one side and rotation of a square. Once they get that, they can copy and paste the code three more times to have a square. The slides break down this process, with sample code on slides 20 & 21.

Students will use the safety feature in this program.

💡 **Teaching tip: Objective 10** -- slides 23-24
This objective introduces button presses, which were discussed earlier in the slides. Now students see the possibility of using both buttons and not just one. The terms “control flow” and “branching” are introduced.

💡 **Teaching tip: Objective 10 Activity** -- slides 25-29
This activity uses a simple program that students will step through using the debugger. The code for the program is given on slide 25.

Students will “Step In” this program using the debugger three separate times. The first time pressing BTN-0, the second time pressing BTN-1, and the third time not pressing any button. The goal is for them to see the control flow of the if statement. Once a condition is true, the indented code is executed and the rest of the code is skipped. They will record their observations in their mission logs. The slides walk them through the debugging and stepping in process.

The last slide summarizes the control flow of the if / elif / else statement.

💡 **Teaching tip: Objective 11** -- slides 30-31
This objective applies what students have been learning about control flow and button presses. They can continue to use the WhatIf program, and add code to meet the goals. In CodeSpace, it asks students to use the NavSquare program and add to it. You can do that, but the slides will tell students to keep with their most recent program, which is WhatIf.

Slide 31 shows a possible algorithm for their completed program. They really can do anything for the second button press, but the example on the slide is to use one of their earlier Mission 3 programs, like SequenceLEDs or BinaryLEDs. That way the code is already done and they are focusing on the control structure and not a specific task for moving CodeBot. They will write their algorithm in their mission log.

💡 **Teaching tip: Objective 11 Activity** -- slides 32-35
The first instruction for modifying the code is to add four lines to the safety feature. The code they have is fine, but in order to meet the goal in Obj. 11, they need to have similar code to this. If you unlock the mission so they don’t have to meet the goals, you can skip this part.

The next two steps for the activity are to open previous programs and copy and paste code into the WhatIf program. The code for the square should already be indented, so no problem there. The code for the second branch, from a previous program, will not already be indented. All lines inside the elif branch need to be indented. Remember: the shortcut for this is to highlight the code and press <TAB>.

Optional: 🗝️ Mission 3 Obj 9-11 Kahoot! Review.

A review Kahoot! Is available for this lesson. You can do the Kahoot together as a class, or assign it independently. It includes two review questions that are frequently missed from an earlier Kahoot!

👫 **Post-Mission Reflection:**

The post-mission reflection asks students to summarize the control flow of an if/elif/else statement. This was done on slide 29, if students need a refresher.

You can use an extension or cross-curricular activity as post-mission activity.

You can use the Mission 3 Obj. 9-11 Kahoot as a lesson review. (link above)

End by collecting the Mission 3 Lesson 4 Log.

SUCCESS CRITERIA:

- ❏ Use buttons.was_pressed() in an if statement
- ❏ Implement a safety feature in code
- ❏ Plan a program using an algorithm
- ❏ Write code to drive CodeBot in a specific pattern
- ❏ Use buttons.was_pressed() in an if/elif/else statement