


MISSION 4: Display Games		Time: 50 minutes
<p>Overview:</p> <p>From car dashboards to giant stadium scoreboards, you see LED displays everywhere, and most of them are controlled by software. The CodeX display is small, but with <i>your code</i>, it can do a lot!</p> <p>In Mission 4, students will program the CodeX to display text and get input from the user by pushing buttons to create a game.</p>		<p>Objectives:</p> <ul style="list-style-type: none"> I can show and print text <i>strings</i>. I can program buttons to determine whether they are pressed. I can explain the difference between integer and string and use each appropriately. I can convert between string and integer data types. I can write an if:else conditional statement.
<p>Standards:</p> <p>2-CS-03 Systematically identify and fix problems with computing devices and their components</p> <p>2-AP-11 Create clearly named variables that represent different data types and perform operations on their values.</p> <p>2-AP-12 Design and iteratively develop programs that combine control structures, including nested loops and compound conditionals.</p>	<p>CSP Framework:</p> <p>Computational Thinking Practices:</p> <p>4.C Identify and correct errors in algorithms and programs, including error discovery through testing.</p> <p>6.A Collaborate in the development of solutions.</p>	<p>Key Concepts:</p> <ul style="list-style-type: none"> Arguments are introduced to students when calling built-in functions Students write code to perform a calculation (integer) Built-in functions: <code>sleep()</code>, <code>display.show()</code>, <code>buttons.is_pressed(BTN_U)</code> Values have different data types, like <i>integer</i> and <i>string</i>. Indenting after a colon is very important! Batteries Included! The CodeX can make your code portable.
<p>Preparation:</p> <p>Make a copy of the assignment or put it in the LMS.</p> <p>Prepare any formative assessments you want to use in the wrap-up</p> <p>Each student/pair needs</p> <ul style="list-style-type: none"> a computer/ Chrome a CodeX & USB cable assignment guide 	<p>Links:</p> <ul style="list-style-type: none"> Codespace: https://sim.firialabs.com/ Assignment Review Kahoot #2 Daily reflection form (use your link) 	<p>Agenda:</p> <ul style="list-style-type: none"> Warm-up (5 minutes) Mission 4 (40 minutes) Wrap-up & Assessment (5 minutes)
<p>Vocabulary:</p> <ul style="list-style-type: none"> Argument: Passing data to functions (information a function uses to complete its task) Integer: A whole number that can be positive, negative or zero String: A sequence of characters, like words or sentences Conversion Function: a built-in function that converts a value to a different (and specific) data type Branching: Decision points in code; a condition Selection: Decision points in code; a condition – this isn't in the documentation but is used in AP CSP Boolean: True or False data type (values that can be True or False; used with a condition) Indention: Structuring blocks of code in Python; statements ending with a colon (:) execute the block of code indented four spaces beneath it 		

Assessment:

- Daily reflection journal or Google form (use your own form)
- Review Kahoot:
<https://create.kahoot.it/share/firia-labs-ap-csp-mission-4/8dcfdc72-a9ba-415b-ba47-5db29d06003e>
- More suggestions listed below in the Walk-Through Wrap-up


Teaching Guide

Warm-up (5 minutes)

 **Discuss** – Use a discussion strategy, like journaling, working at boards, selecting random students, or a form of think-pair-share.

Review with students the concepts from Mission 3 – variables, RGB and pixels. If you have time, you could use the [Kahoot from the last lesson](#), or something similar. Or you might want to do a very short warm-up and get right to the coding.

Activity – Mission #4 (40 minutes)

 Randomly group students into pairs for pair programming.

Students log in to one computer. Two computers can be used if they want to see instructions on one computer and work on the other computer. However, the assignment document requires snippets, so it will need to be open on the same computer as CodeSpace.

Teaching tip – Before they start:

Review the [Mission Reminders slides](#).

Remind students that they need to document their errors and how they fixed them. There is a table at the end of the document for this.

Students go to sims.firialabs.com and should be at the beginning of Mission 3

Teaching tip – Objective 1:

The term “**argument**” is introduced. Students should click on the wrench to see documentation. The documentation includes positional and keyword arguments. They don’t need to worry about that – only the definition at the beginning of the paragraphs. You may want to expand on the definition (see definition above) and give examples.

- `sleep(1)` – argument is 1
- `sleep(delay)` – argument is delay
- `pixels.set(0, color)` – arguments are 0 and color
- `display.show(pics.PLANE)` – argument is `pics.PLANE`

Teaching tip – Objective 3:

Students have to add a line of code to calculate num and then use num in the display command. This is a possible source of confusion. They can use CodeTrek if they get frustrated.



Also, they will get an error in their code. The instructions clearly state this, but if students are not reading the instructions they may think they are doing something wrong.

Teaching tip – Objective 5:

The code is running **sequentially**, so only “world” shows on the screen. Try to get the students to remember vocabulary from one mission to the next, and especially emphasize words used on the AP CSP create performance task prompts.


Teaching tip –Objective 7:

This is the first time students use a Boolean condition and if statement. The instructions suggest they use the debugger to step through the code and see what it is doing. You may want to guide them through this, or you may believe they are understanding without stepping through. More unplugged practice will be coming with if statements.


Teaching tip – Extension:

This mission should take a regular class period. However, if some students finish early, or the mission is extended to two days, they may have time for an extension. Students can come up with their own idea, or try one of these:

- Fill the screen with red or green (or a short sleep) in addition to changing a pixel
- Add an image at the end of the game
- Display an image after each challenge for a short sleep before asking for the next button
- Add a counter and display a winning or losing message or image

 Assignment is complete and ready to turn in. Both students should include their names on the document.

Wrap-Up (5 minutes)

 **Discuss** – Use a discussion strategy, like journaling, working at boards, selecting random students, or a form of think-pair-share.


Discuss with students real-world applications. The skills they used in the project are used by professional software developers to build:

- Traffic Lights and People Counters
- Sports Event Scoreboards
- Games

IMPORTANT!!

Students should clear their CodeX by running their ClearCodeX program.

If students haven't created a “clear” program yet, they can follow the steps in the slide deck (mission 1).

 If this lesson is completed in one class period, the following can be used as a wrap-up. If you are on a block schedule and continuing to the next lesson, a wrap-up isn't necessary.

Formative Assessment:

- Daily reflection journal or Google form -- use your own form
- Kahoot #3 (in class or individual):
<https://create.kahoot.it/share/firia-labs-ap-csp-mission-4/8dcfdc72-a9ba-415b-ba47-5db29d06003e>
- Exit ticket on vocabulary (many to choose from)
- Group review on vocabulary (many to choose from)



- Students add to their programming journal (vocabulary and debugging chart).

SUCCESS CRITERIA:

- Define and use an argument in a function call
- Understand and use variable types, converting types when needed
- Use a Boolean condition in an if..then statement
- Receive input from the user through a push-button
- Program a push-button to make a fast-click game.
- Debug any errors in the code and keep a debugging table
- Write a program, run it, and save it to the CodeX