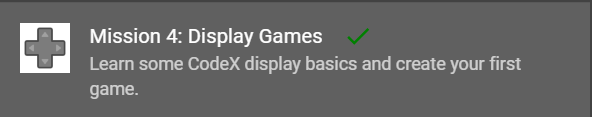
**Mission 4:**

**Display Games**

**Student Workbook**



****

**Greetings!**

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 *your code*, it can do a lot!



Go to the Mission 4 Log and fill out the Pre-Mission preparation.

**Mission 4: Display Games**



In Mission 4, you will program the CodeX to display text and get input from the user by pushing buttons to create a game.

**Mission 4: Get started**

* Go to <https://make.firialabs.com/> and log in.



* Go to Mission 4

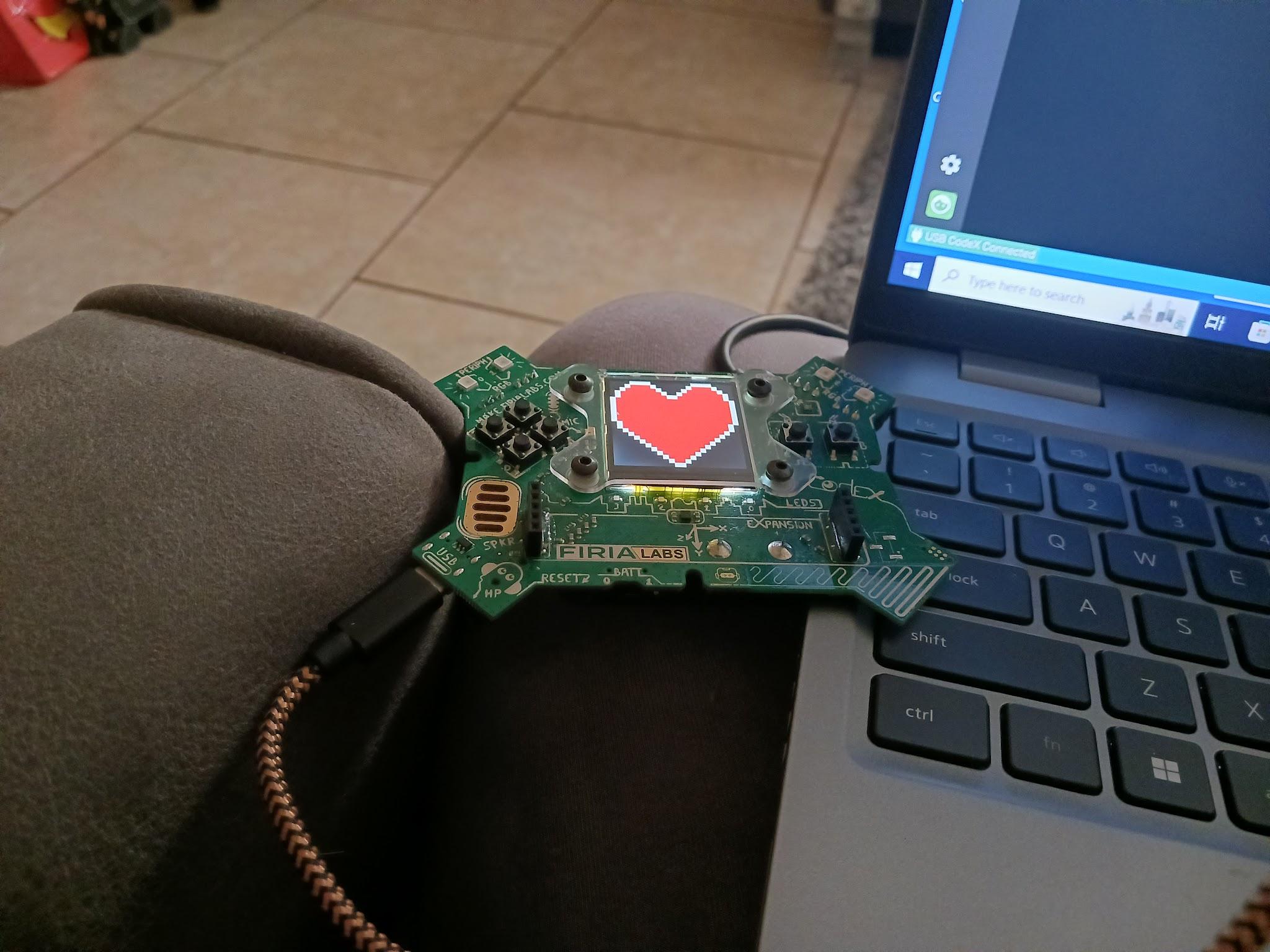
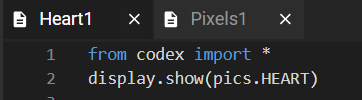


* Click and start Mission 4.

**Objective #1: Back to the display**

In Mission 2, your program displayed an image. The first image you displayed was a HEART.

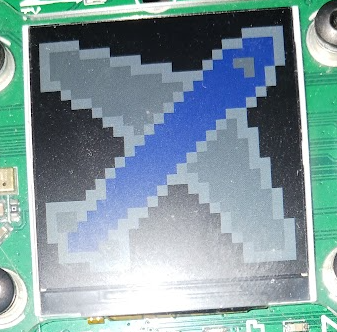
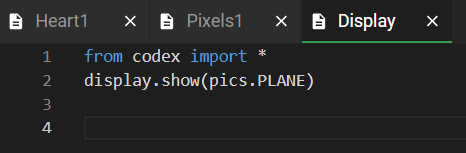
* You will practice displaying an image on the LCD screen.



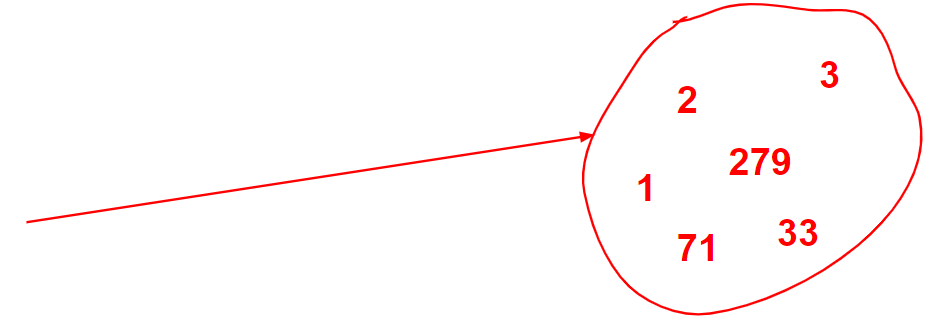
**Objective #1: Back to the display**

**DO THIS:**

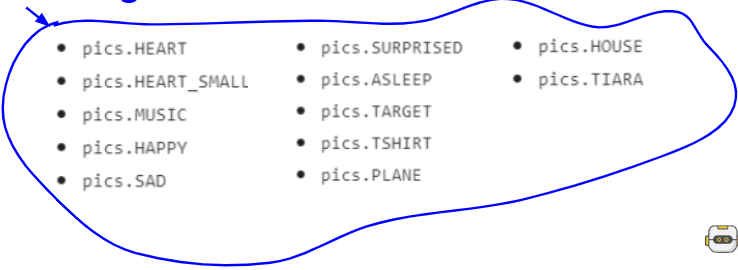
* Create a new file named **Display**
* Click the **File** menu button
* Select “New File…”
* Name the file **Display**
  + *no spaces in a file name*
* Click **Create**
* Add two lines of code to display a PLANE
* Run your code



**Objective #2: Text messages**

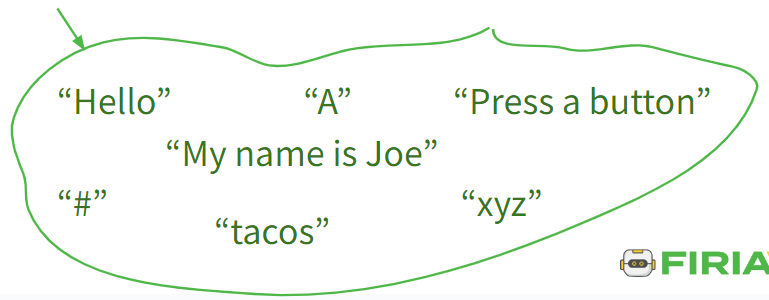
Computers work with different **types** of data. So far you have worked with:

* **Integers**   
  (counting numbers)
* **CodeX images**



You might also want to display words.

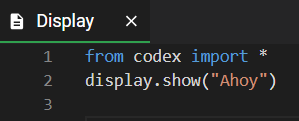
* Words, letters, and characters are the data **type** **string**
* Indicate a **string** by using **“quotation marks”**



**Objective #2: Text messages**

**DO THIS:**

* Go to the Mission Log
* Write examples of different data types
* Change the code in the display.show()   
  function to display the text “Ahoy”



**Objective #3: Good with numbers?**

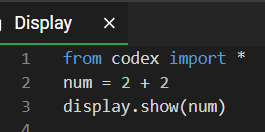
A computer is very good at doing math.

* When you define a **variable**, you assign it a value
* So far you assigned a **literal** value
* You can also assign a value by doing math

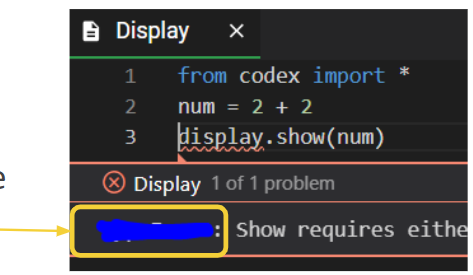


**DO THIS:**

Use a simple calculation to assign a value to a variable

* Add a line of code   
   that uses the   
   assignment statement   
   shown above
* Use the display.show()   
   statement to show the num variable

The code caused an error

* Go to your Mission Log and write the error message

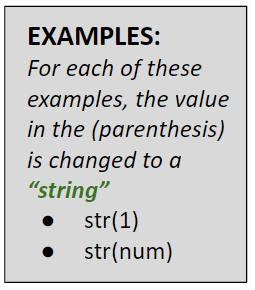
**Objective #4: Converting types**

Why does display.show(num) not work?

* display.show(“Ahoy”) works
  + **“Ahoy”** is a type **string**
* display.show(pics.HEART) works
  + **pics.HEART** is a type **CodeX image**
* display.show(num) does not work
  + **num** is a type **integer**

Why does display.show(num) not work?

It doesn’t work with an **integer**, but it will work with a **string**

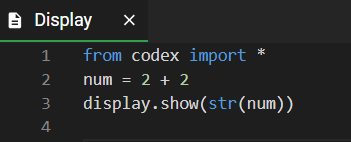
* If an integer is converted, or changed, to a string, then display.show() will work -- no error
* Python has a function that will convert (change) any value to a **string**
  + **str()**

**Objective #4: Converting types**

**DO THIS:**

Modify your code by using the **str()** conversion function.

* Change the display.show(num) code to use   
   the **str()** function
  + *Be careful to match your parenthesis*
* Run your code



**Objective #5: Second show message**

Can you display two messages?

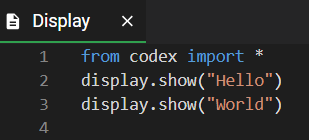
In Mission 3, you tried showing two (or four) different colors in a pixel. This didn’t work until you slowed down the program by using a sleep().

What do you think will happen if you try to display two messages?



**DO THIS:**

* What do you think will happen if you try to   
  print two messages?
* Go to the Mission Log and write your prediction
* Then change your code to display two   
  messages:



**Objective #6: Printing text**

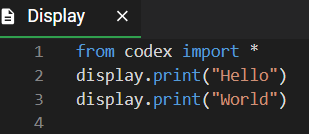
The display.show() command will only show one thing at a time. So, just like the pixels, the second thing is displayed on top of the first thing.

* CodeX has another way to display a **string**
* Use **display.print(“string”)**

All **display.print(“string”)** messages will be displayed, one after another -- each on its own line

**DO THIS:**

Change your code to **print** the **strings**.

* Change the display.show() command to   
   display.print()



**Mission Quiz: Typed and Printed**

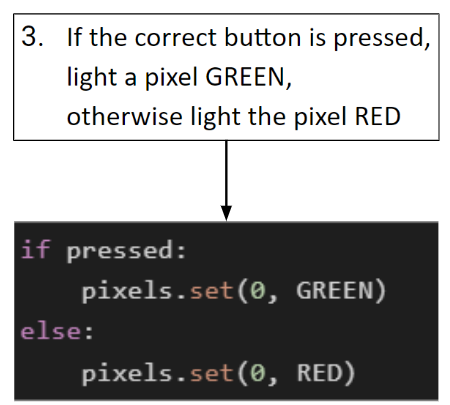
Test your skills by **taking the quiz**.

**Objective #7: Branching**

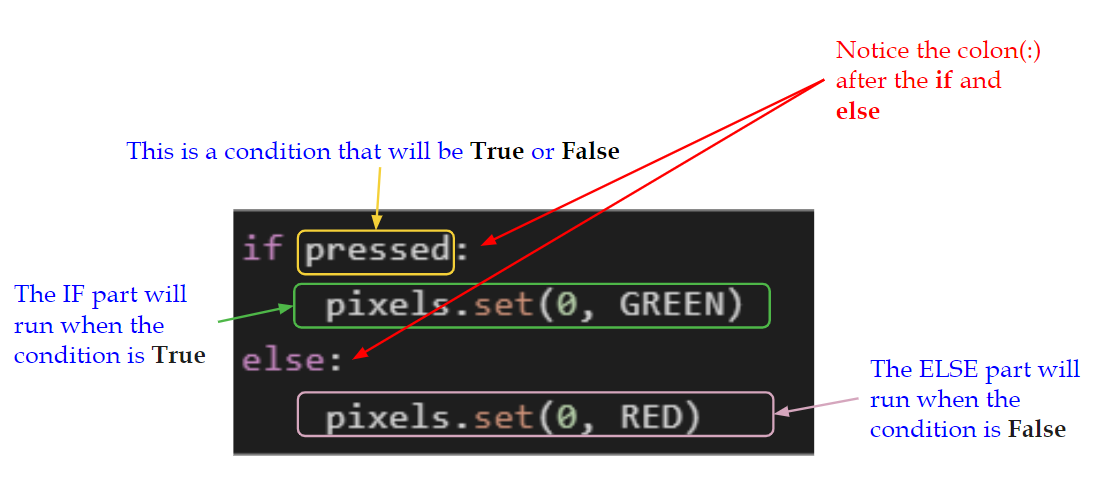
During the next objectives, you will create a button-pressing game. Here are the parts of the game:

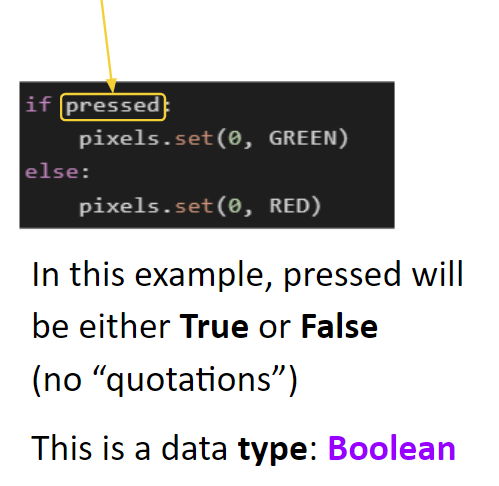
1. Display a button to press.
2. Press and hold the button. You will have one second.
3. If the correct button is pressed, light a pixel GREEN, otherwise light the pixel RED

Step #3 is a new concept -- branching.

* Branching is when the computer makes a choice between two things.
* Here is an example of branching.
* Notice the indenting -- this is very important!

Take a closer look at branching:

****



**Objective #7: Branching**

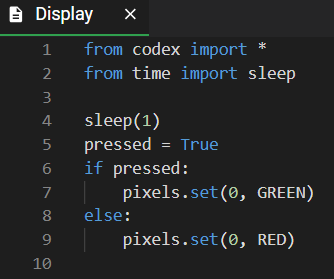
Now you know four data types:

* Integer -- Examples: 1, 54, 720
* CodeX image -- Examples: pics.HEART, pics.MUSIC
* String -- Examples: “Hello”, “Press A”, “cake”
* Boolean -- Examples: True, False



**DO THIS:**

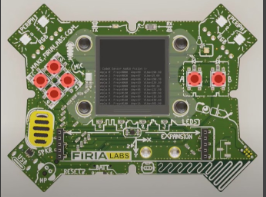
The best way to learn about branching is to try it:

* Delete most of your code
* Type the code below
* Run the code
  + Do you see a GREEN light?
* Change line 5 to **pressed = False**
* Run the code
  + Do you see a RED light?

**Objective #8: Button hunting**

The game you will make will use four of the six buttons.

* Look at the picture of the CodeX.
* Can you find all 6 buttons?

****



**DO THIS:**

* Close the instructions panel
* Use the camera to rotate the CodeX until   
  you see the front
* Click on all 6 buttons

**Objective #9: Gamer input**

There are 2 ways to check for a button press:

* buttons.**was**\_pressed(**BTN\_A**)
  + Checks to see if button A was pressed since the last check
* buttons.**is**\_pressed(**BTN\_A**)
  + Checks to see if button A is currently pressed

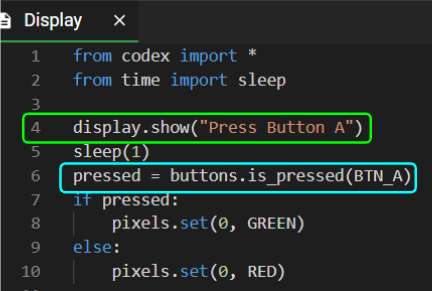
*Any of the 6 buttons can be checked in ( ).*



**DO THIS:**

For this game, you will check for currently pressed

* Add line 4
* Change line 6
* Run the code and press Button A
* Run the code again and do not press Button A
* Do you get the results you expect?

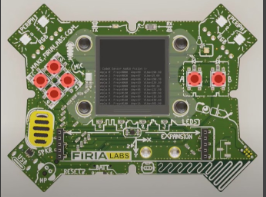


**Mission Quiz: Buttons and Branching**

Test your skills by **taking the quiz**.

**Objective #10: For the win!**

Now just check a few more buttons and you have a serious twitch game!

* You can use whatever buttons you want
* You have 6 buttons to choose from:
  + - BTN\_A
    - BTN\_B
    - BTN\_U
    - BTN\_D
    - BTN\_L
    - BTN\_R
* What 4 buttons do you want to use for your game?
* 

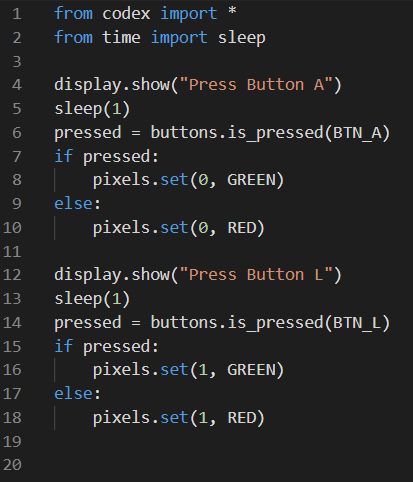
**DO THIS:**

* Go to the Mission Log and record the four   
   buttons you will use for your game

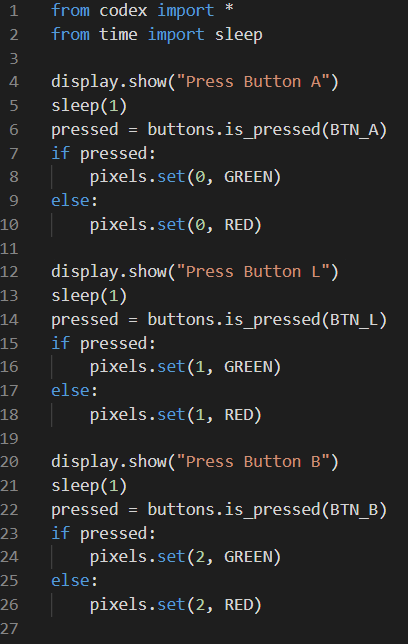
**Objective #10: For the win!**

**DO THIS:**

* In your code, copy line 4 through line 10
* Paste the code below line 10
* Change Button A to the second button you   
   want to use
  + Change it in the display.show()
  + Change it in buttons.is\_pressed
* Change the pixel from 0 to 1



* Paste the code again, below the   
   current code
* Change the button to the third button   
   you want to use
  + Change it in the display.show()
  + Change it in buttons.is\_pressed
* Change the pixel to 2

Paste the code one more time,   
below the current code

* Change the button to the fourth button you want to use
  + Change it in the display.show()
  + Change it in buttons.is\_pressed
* Change the pixel to 3

At this point you should have code for the four buttons you chose.

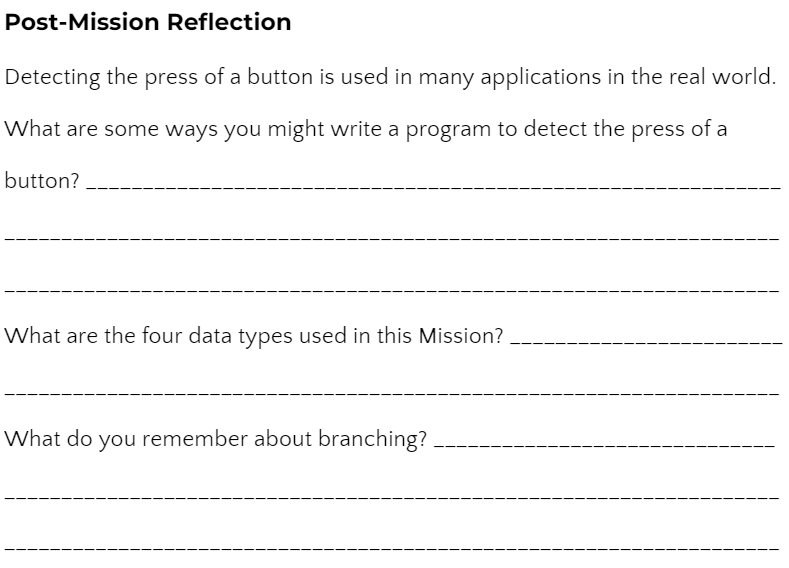
**DO THIS:**

* Run the code
* If you have any errors, fix them
* Try pressing all the buttons and getting all green lights
* Try the code again, missing some of the buttons
* Do you get the results you expect?
* Make any changes you need to so that your program works correctly
* Have someone else try your game

**Mission Complete**

You have completed the fourth mission. 

**Do this:**

* Read your “Completed Mission” message
* Complete your Mission 4 Log
  + Post-Mission Reflection
* Get ready for your next mission!

**Wait! Before you go … Clear the CodeX**

Go to FILE -- BROWSE FILES

Select the “**Clear**” file and open it

Run the program to clear the CodeX

**Okay. Now you can go.**