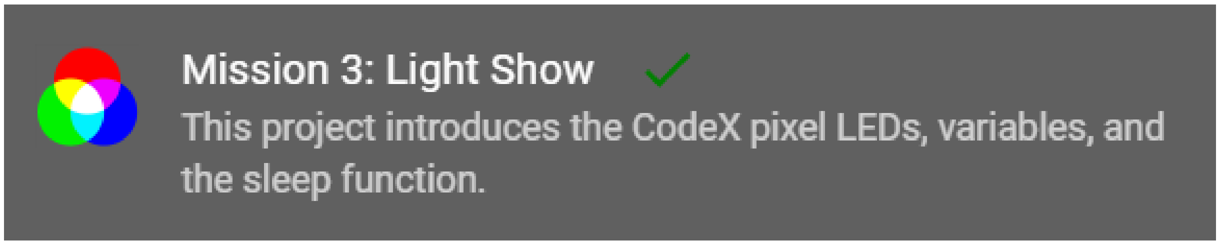
Mission 3:
Light Show

Student Workbook

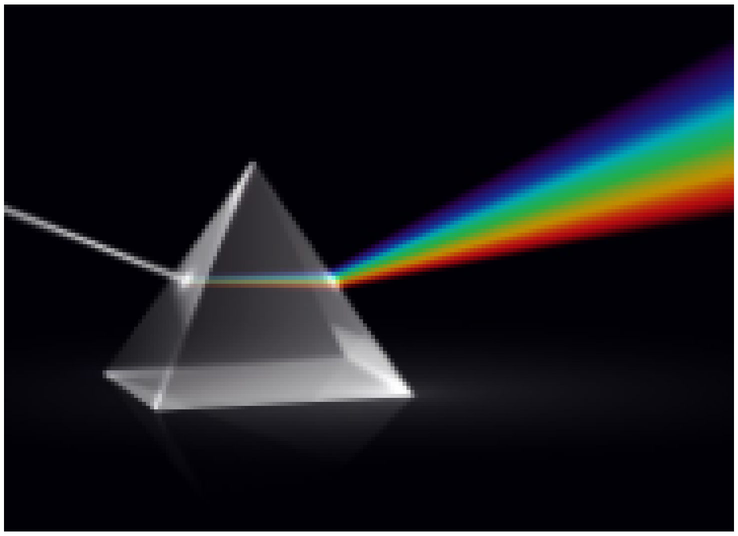


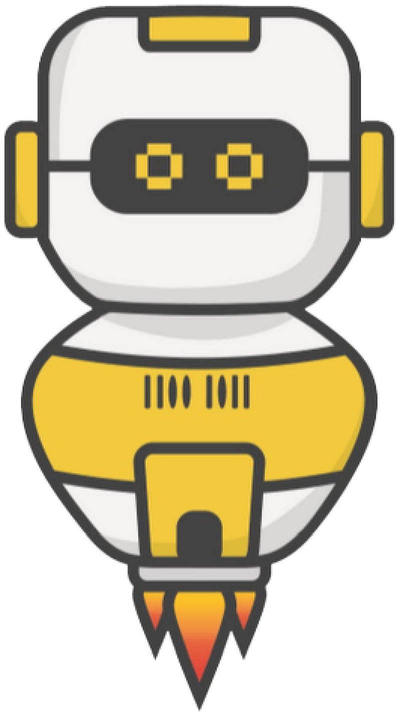
Mission 3: Light Show ✓

This project introduces the CodeX pixel LEDs, variables, and the sleep function.

Welcome back!

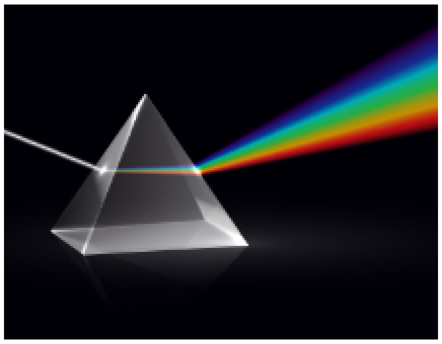
This mission will explore the colorful lights at the top of the CodeX. You will learn how light can make all the colors you see.





Go to the Mission 3 Log and fill out the Pre-Mission preparation.

Mission 3: Light Show



The CodeX has 10 LEDs (little lights). Four of the lights are Smart RGB LEDs -- also called pixel LEDs.

- What is a Smart RGB LED (pixel LED)?
- With only three colors -- **red**, **green** and **blue** -- the LED can display any color
- Click on **Pixel LEDs** to add it to your toolbox.
- Click on the picture and watch the video on pixels:

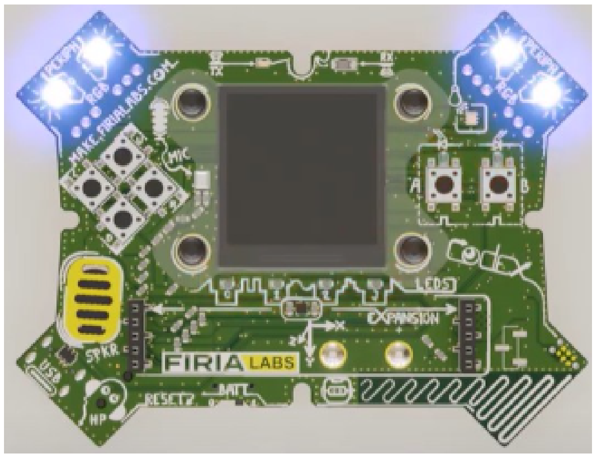
Mission 3: Get started

- Go to https://make.firialabs.com/ and log in.
- Click **NEXT** and start Mission 3.

Objective #1: Find the pixels

The CodeX has four Red, Green, Blue (RGB) LEDs along the top. These are the Smart RGB LEDs, or pixel LEDs.

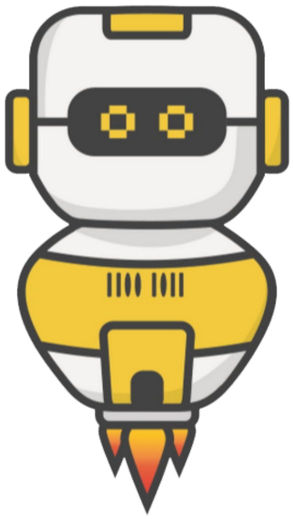
- You can set these LEDs to any color under the sun.



The CodeX library gives you some built-in colors to use:

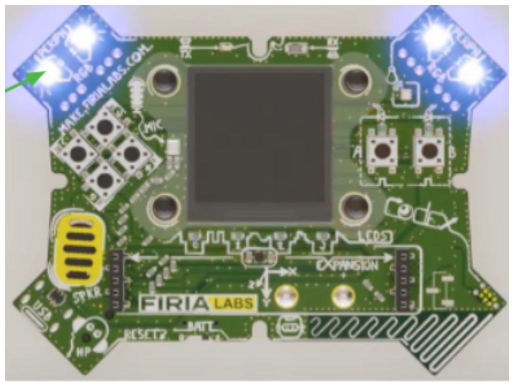
- BLACK (same as off)
- BROWN
- RED
- ORANGE
- YELLOW
- GREEN
- BLUE
- PURPLE
- GRAY
- WHITE
- CYAN
- MAGENTA
- PINK
- LIGHT_GRAY
- DARK_GREEN
- DARK_BLUE

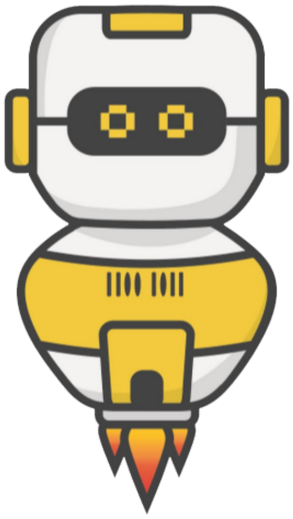
Objective #1: Find the pixels



DO THIS:

- Close the instruction panel
- Use the camera controls to rotate CodeX
- Click on pixel 0 (the first pixel LED)





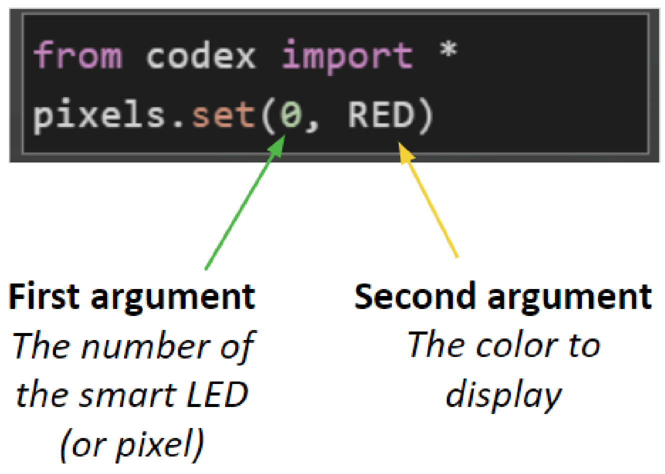
DO THIS:

- Create a new file named **Pixels1**
- Click the **File** menu button
- Select “New File…”
- Name the file **Pixels1**
  - *no spaces in a file name*
- Click **Create**

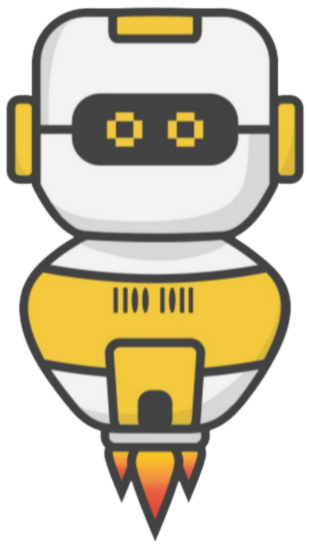
Objective #2: Turn on the red light

In Python, use the function:
pixels.set(number, color)
to turn on a pixel LED

- The function takes two inputs, called **arguments**

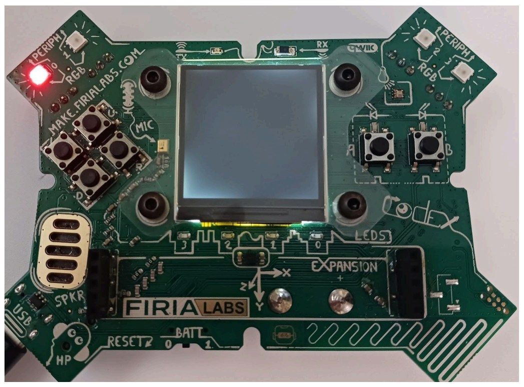


```
from codex import *
pixels.set(0, RED)
```



DO THIS:

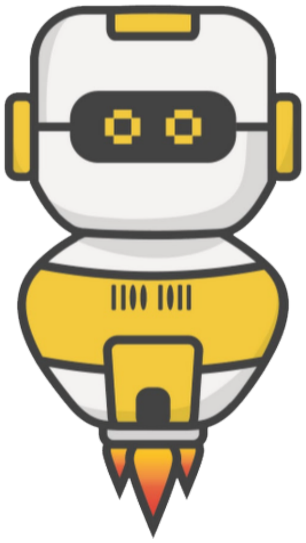
- Click on CodeTrek (same code as above)
- Type the code into your new file
- Run the code
- Change the code to display a different color (use from the list above)



Objective #3: Two in a row

Now display two colors in sequence.

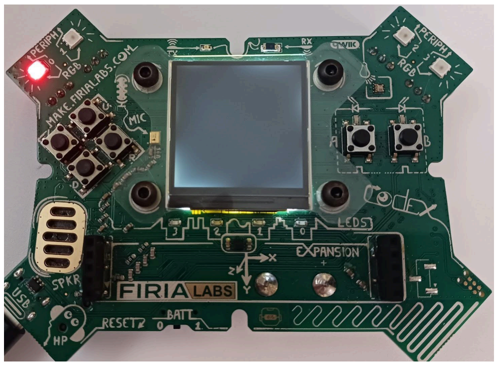
- **Sequence** means "go in order, one line at a time"
- But computers are very fast!

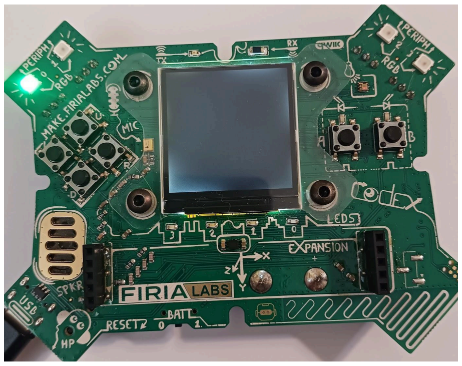


DO THIS:

Using only pixel 0, turn it RED and then GREEN

- The **Pixels1** code turns pixel 0 to RED
- Add another line of code (line 3) that will turn pixel 0 to GREEN
- Did you see both colors?

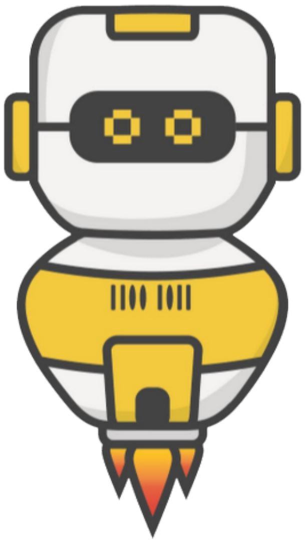




Objective #4: What's going on?

Why is only the last color showing?

- Did you notice the program ends very quickly?
- It doesn't wait for you to see the first color before it displays the second color
- And the last color stays on even after the program ends



DO THIS:

Add more colors to your code. Using only pixel 0, change it to four different colors.

- Add two more lines of code to **Pixels1**
- Line 4 -- change to BLUE
- Line 5 -- change to WHITE
- Before you run your code, predict what will happen.
- Write your prediction in the Mission Log
- Run your code

Mission Quiz

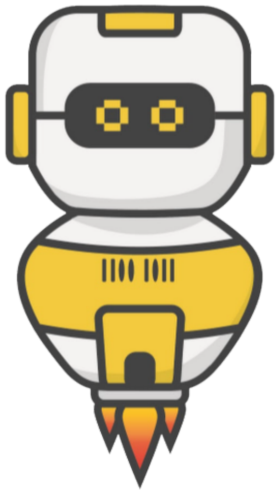
Test your skills by **taking the quiz**.

Objective #5: Find the bug

Inside the mind of a computer

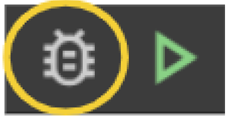
Computers are fast. Even a small computer like the CodeX can execute *millions* of operations per second!

CodeSpace includes an important tool called the “debugger”. It slows down the program execution and lets you see the code running sequentially -- one line at a time.



DO THIS:

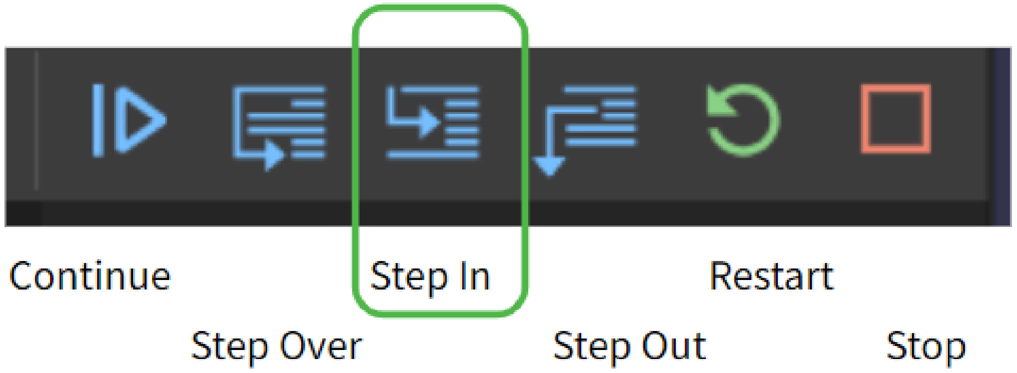
- Click on debug to add it to your toolbox
- Write the definition of ‘bug’ in your Mission Log
- Watch the debug video in the instruction panel
- Answer the question in your Mission Log
- Click on the debug button

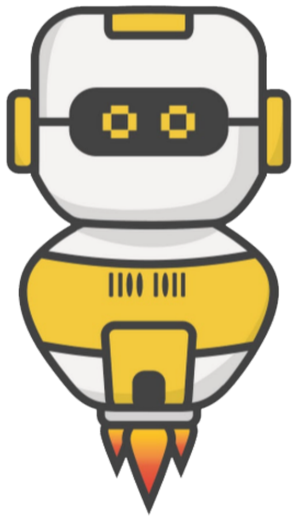


Objective #6: Step by step colors

Your turn to use the debugger

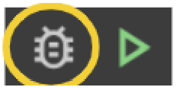
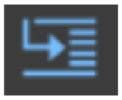
- You can execute, or run, one line at a time using the "Step In" button, which is part of the debugger.
- Remember: The code is run AFTER the line is highlighted.





DO THIS:

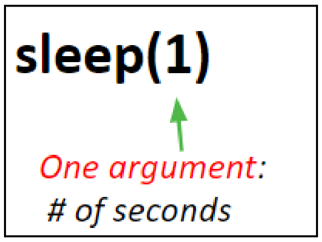
Try stepping through your code

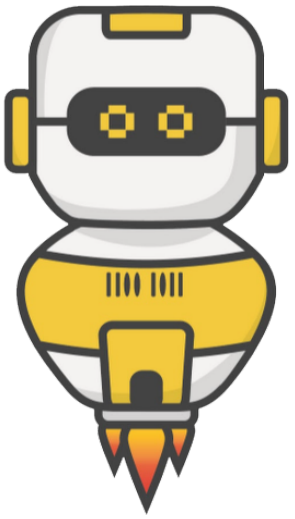
- Click the debug icon 
- Click the **Step In** icon 
- Slowly click the **Step In** icon several more times, observing what happens after each click
- Now do you see all four colors?

Objective #7: Slow it down

When you *step slowly* through the code, all the colors show.

- You just need a way to delay the computer a little after it shows each color
- The **time module** has a **sleep()** function that will pause program execution
- **sleep(1)** will delay (or pause) the program for 1 second





DO THIS:

Update your code

- Click on delay to add it to the toolbox
- Open CodeTrek
- Change your code to look like the code in CodeTrek
- **Remember:** type the actual **sleep(1)** command for line 9, not the **#TODO** comment
- Run your code

Objective #8: Name that number

It would be fun to play with some different delay times.

- Right now the number 1 appears *three* times in the code
- If you want to change the pause shorter or longer, you would need to change 1 three times
- This number is called a **'literal'**

Instead of repeating a ***literal number*** like 1 in your code, you can use a name instead.

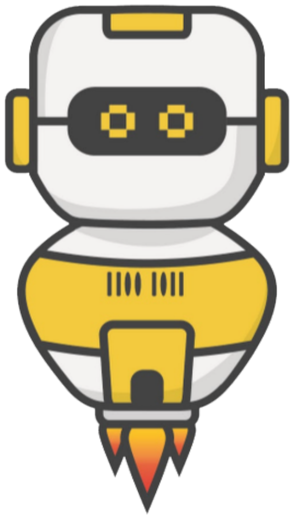
- A **variable** is a **name** you can give to data
- Data can be a number, a color, an image, or even words (called strings)
- The **variable name** describes the data
- A **variable** is defined before it is used in code

Once a **variable** is defined, it can be used instead of a **literal**. Then if you want to change the value, you only have to change it once.

Examples of **defining variables**:

```
delay = 1
color = RED
picture = pics.HAPPY
name = 'Bob'
```

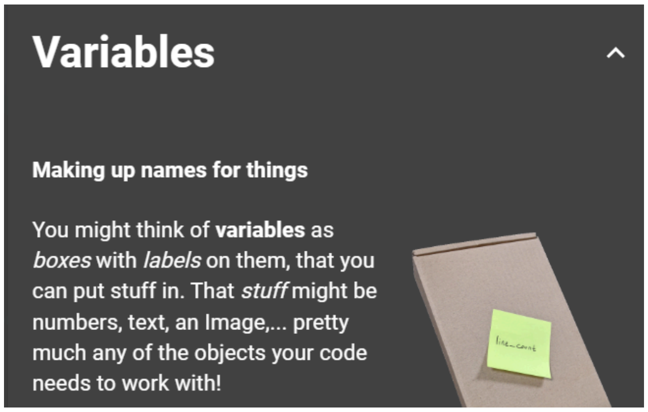
Objective #8: Name that number



DO THIS:

Learn more about variables

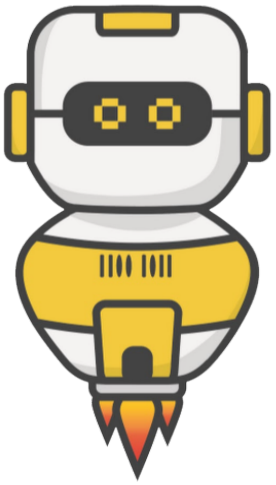
- Click on 🔧variable to add it to the toolbox
- Answer the Mission Log questions



Variables

Making up names for things

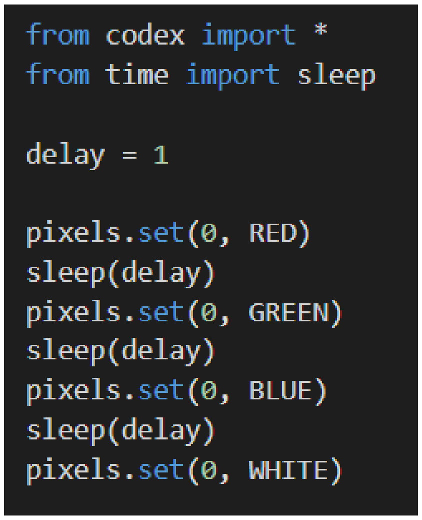
You might think of **variables** as *boxes* with *labels* on them, that you can put stuff in. That *stuff* might be numbers, text, an Image,... pretty much any of the objects your code needs to work with!



DO THIS:

Update your code

- Open CodeTrek and use it to help you modify your code
- Define a **variable** called ***delay***
- Use ***delay*** in the sleep() functions
- Run your code
- Did you see all four colors?



```
from codex import *
from time import sleep

delay = 1

pixels.set(0, RED)
sleep(delay)
pixels.set(0, GREEN)
sleep(delay)
pixels.set(0, BLUE)
sleep(delay)
pixels.set(0, WHITE)
```

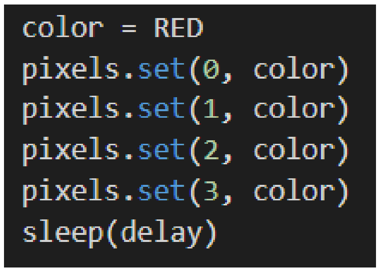
Mission Quiz

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

Objective #9: Warning sign

Time to light up all four pixel LEDs

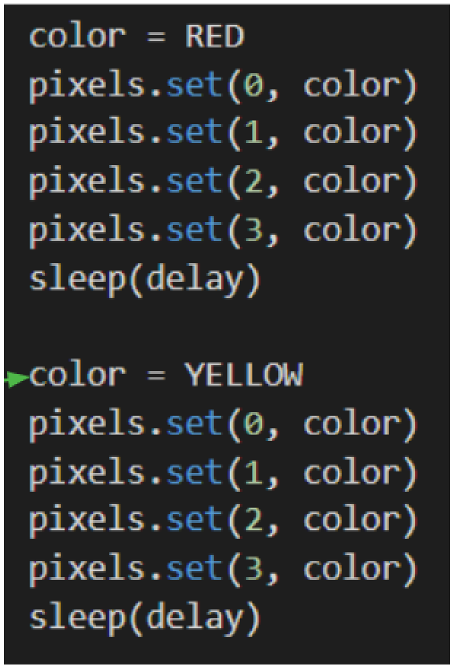
- You can use **pixels.set()** to light up all four pixels
- Just change the argument for the pixel number
- You can use a variable for the color argument



```
color = RED
pixels.set(0, color)
pixels.set(1, color)
pixels.set(2, color)
pixels.set(3, color)
sleep(delay)
```

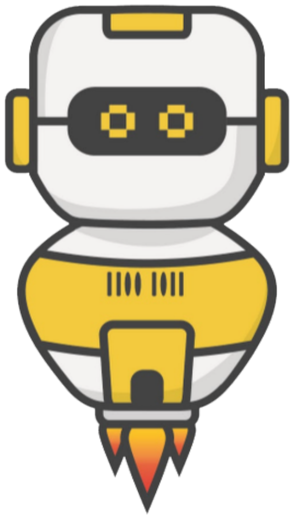
Time to light up all four pixel LEDs

- You can change the value of a variable any time in the code
- This example shows changing the value of color
- You can also change the value of delay if you want to



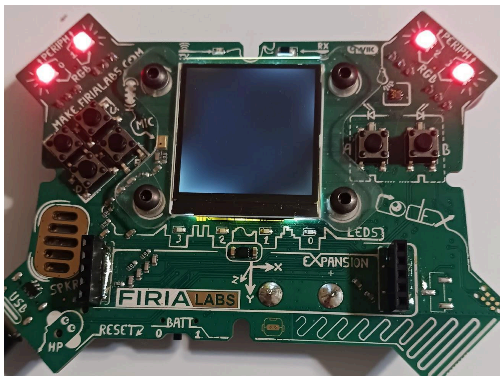
```
color = RED
pixels.set(0, color)
pixels.set(1, color)
pixels.set(2, color)
pixels.set(3, color)
sleep(delay)

color = YELLOW
pixels.set(0, color)
pixels.set(1, color)
pixels.set(2, color)
pixels.set(3, color)
sleep(delay)
```

Objective #9: Warning sign



DO THIS:

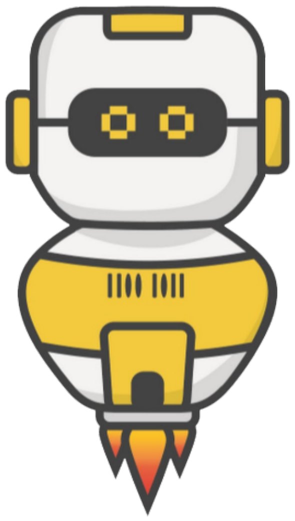
- Create a second variable for **color**
- Set the color to RED
- Use the **color** variable in **pixels.set()** to turn all four pixels the same color
- Change the **color** and turn all four pixels a second color
- Use CodeTrek if you need help
- Run your code
- Optional: repeat the code between RED and YELLOW again to make a flashing warning sign



Objective #9 is complete!

Mission Complete

You have completed the third mission.



Do this:

- Read your “Completed Mission” message
- Complete your Mission 3 Log
    - Post-Mission Reflection
- Get ready for your next mission!

Post-Mission Reflection

You can set the pixel LEDs to any color you want. Lights like these are used in so many applications in the real world.

What are some ways you see smart LEDs (programmable lights) being used?

____

____

____

____

____

____

____

Wait! Before you go ...

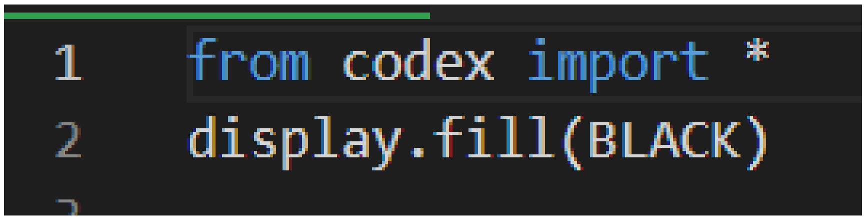
The need for clearing the CodeX

The need for clearing the CodeX

- Every time you run your program, it is loaded onto the CodeX
- The last program run stays on the CodeX, even after it is unplugged from the computer
- So you want the last program run to be something that clears the CodeX and isn't an assignment

Create a "Clear" file

- Create a new file called **Clear**
- Type these two lines of code:



```
from codex import *
display.fill(BLACK)
```

- Run the code
    - The CodeX should be blank, with no pictures or lights on
- Run this code at the end of every class period

Okay. Now you can go.