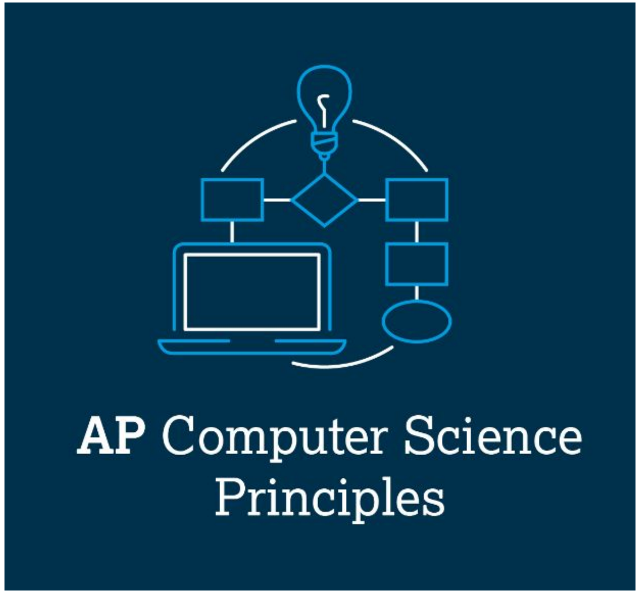
Practice #4

Create Performance Task

AP CSP Create Performance Task

Part of the AP Exam is to create a program that meets specific requirements:

- Creates a list
- Uses a list in a meaningful way
- Has a function with a parameter
  - Parameter is used in an if statement
- Function has:
  - If statement
  - Loop

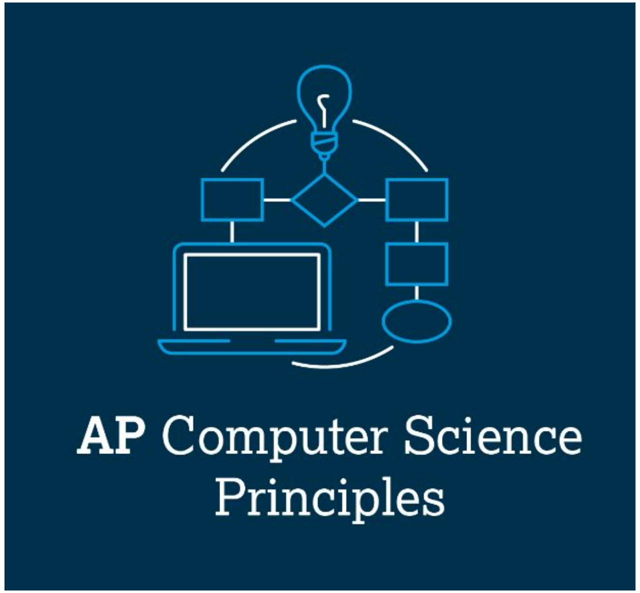


AP CSP Create Performance Task

For this project, you will:

- Start with a program that doesn't yet meet the requirements
- Modify it by adding a loop
- Modify it by adding a list

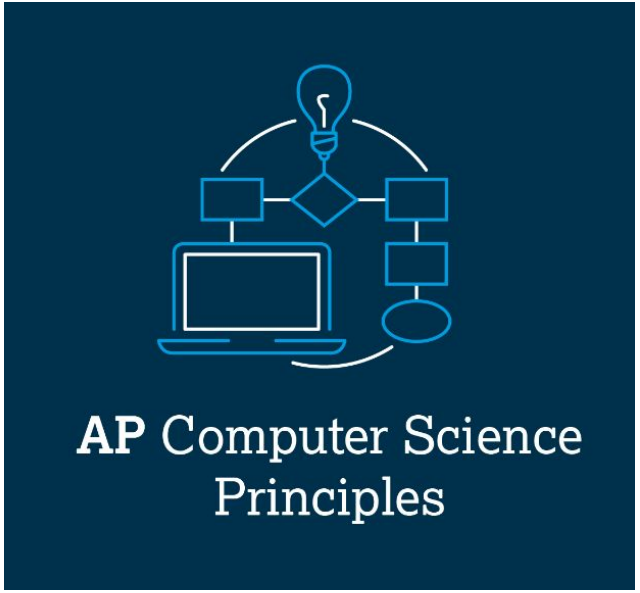
So when you are finished, you will have a program that meets all the requirements for the Create PT



AP CSP Create Performance Task

In **Display3** (from Functions and Global Variables), you created a cool program with a global variable, a parameter, and a function with selection and iteration. But it doesn't meet all the requirements for the Create PT.

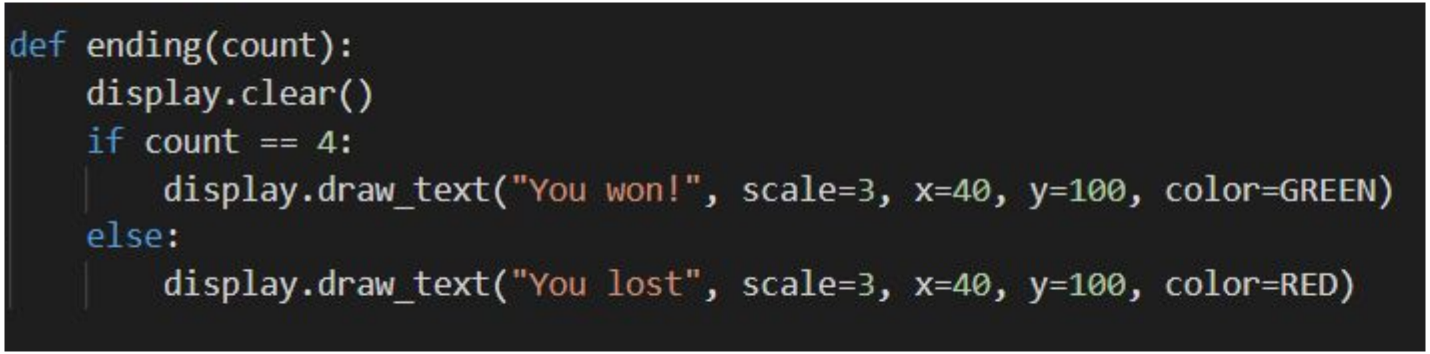
- Creates a list
- Uses a list in a meaningful way
- Has a function with a parameter
    - Parameter is used in an if statement
- Function has:
    - If statement
    - Loop



Create PT Requirements

You will modify this program to meet all the requirements of the Create PT.

- One requirements is a function with a parameter that is used in an if statement
- Does this meet the requirement?



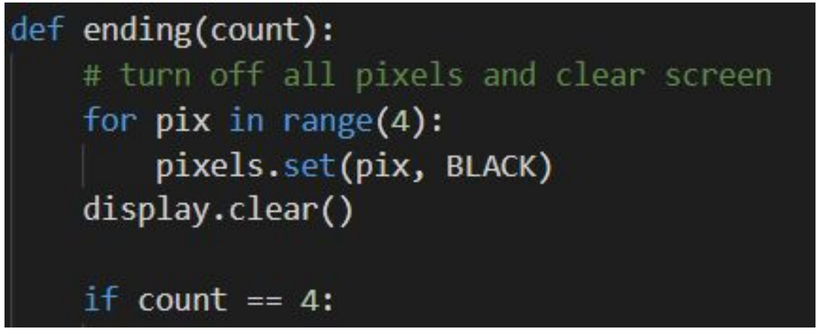
```python
def ending(count):
    display.clear()
    if count == 4:
        display.draw_text("You won!", scale=3, x=40, y=100, color=GREEN)
    else:
        display.draw_text("You lost", scale=3, x=40, y=100, color=RED)
```

Step #1

Open your project "Display3"

- Do a "Save As" to give your project a descriptive name
  - For today's project, you can call it **Practice_PT_4**
- Use a comment block at the top to include your name, date, partner, description of project, etc.

Step #1

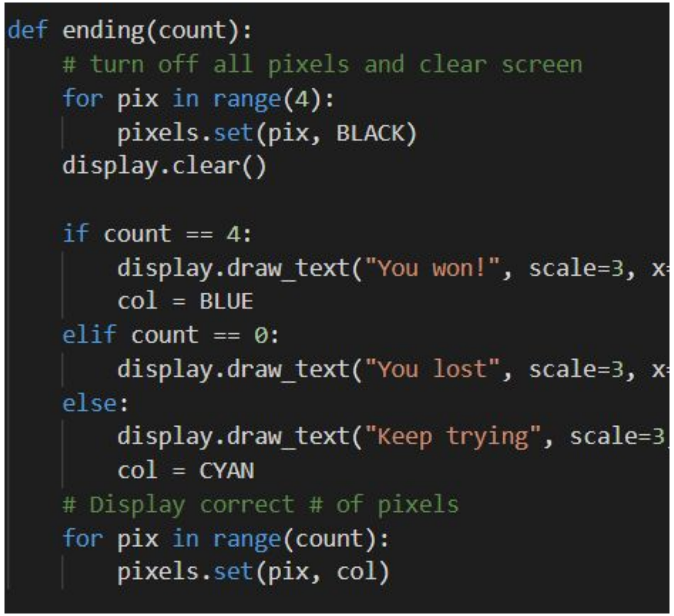


```
def ending(count):
    # turn off all pixels and clear screen
    for pix in range(4):
        pixels.set(pix, BLACK)
    display.clear()

    if count == 4:
```

Add iteration

- Another requirement of the Create PT is that the function have a loop in it.
- Add a loop at the beginning of the **ending()** function that turns all pixels BLACK.
- You have seen this code before. Try it on your own.

Step #1



```
def ending(count):
    # turn off all pixels and clear screen
    for pix in range(4):
        pixels.set(pix, BLACK)
    display.clear()

    if count == 4:
        display.draw_text("You won!", scale=3, x
        col = BLUE
    elif count == 0:
        display.draw_text("You lost", scale=3, x
    else:
        display.draw_text("Keep trying", scale=3
        col = CYAN
    # Display correct # of pixels
    for pix in range(count):
        pixels.set(pix, col)
```

Can you do more?

- Add more selection (branches of the if statement)
  - Add an elif count == 0
  - Other branches for other possibilities
- Use count in a for loop to indicate with pixels how many correct button presses
- Anything else you can think of

Step #2

Does this program meet ALL the requirements?

- Creates a list
- Uses a list in a meaningful way
- Has a function with a parameter
  - Parameter is used in an if statement
- Function has:
  - If statement
  - Loop

Step #2

Does this program meet ALL the requirements?

- If you add a list to the program, and use it in a meaningful way, this project can also meet all the requirements for the Create PT
- There are several possibilities. Let's discuss ONE way to add a list:
  - Use a list for the instructions and buttons (like Practice #3)

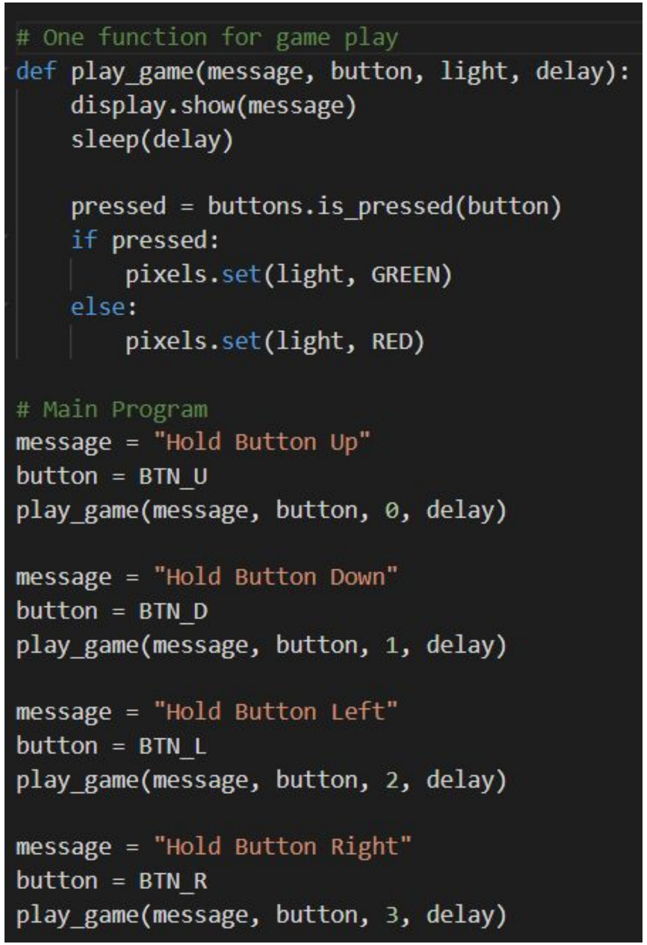
FIRIA LABS

Step #2

Create two lists

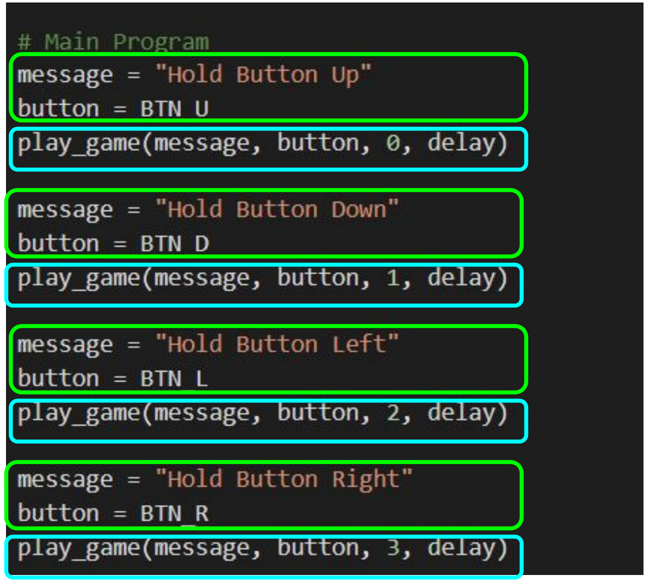
A requirement for the Create PT is to use a list in a meaningful way.

- When you traversed a list, you learned that you can use an index to access the data from two lists.
- You will do the same thing here.



```
# One function for game play
def play_game(message, button, light, delay):
    display.show(message)
    sleep(delay)

    pressed = buttons.is_pressed(button)
    if pressed:
        pixels.set(light, GREEN)
    else:
        pixels.set(light, RED)

# Main Program
message = "Hold Button Up"
button = BTN_U
play_game(message, button, 0, delay)

message = "Hold Button Down"
button = BTN_D
play_game(message, button, 1, delay)

message = "Hold Button Left"
button = BTN_L
play_game(message, button, 2, delay)

message = "Hold Button Right"
button = BTN_R
play_game(message, button, 3, delay)
```

Step #2



```
# Main Program
message = "Hold Button Up"
button = BTN_U
play_game(message, button, 0, delay)

message = "Hold Button Down"
button = BTN_D
play_game(message, button, 1, delay)

message = "Hold Button Left"
button = BTN_L
play_game(message, button, 2, delay)

message = "Hold Button Right"
button = BTN_R
play_game(message, button, 3, delay)
```

Create two lists

The main program should look similar to this example

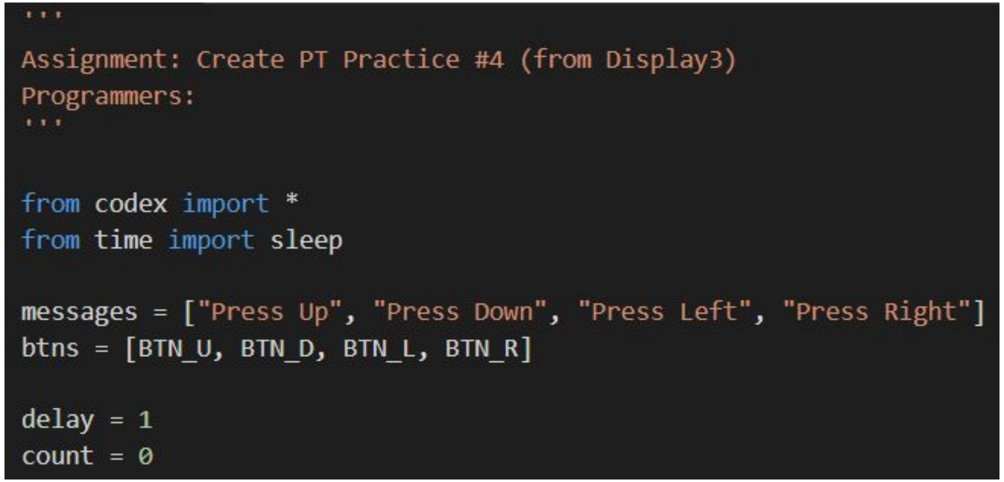
- The information is assigned to variables and then passed as arguments to the function.
- This is repeated four times
- You can put the information into two parallel lists.
- Then, using a loop, you can traverse the list to play the game.

Step #2

Create two lists

At the top of your code, create two lists

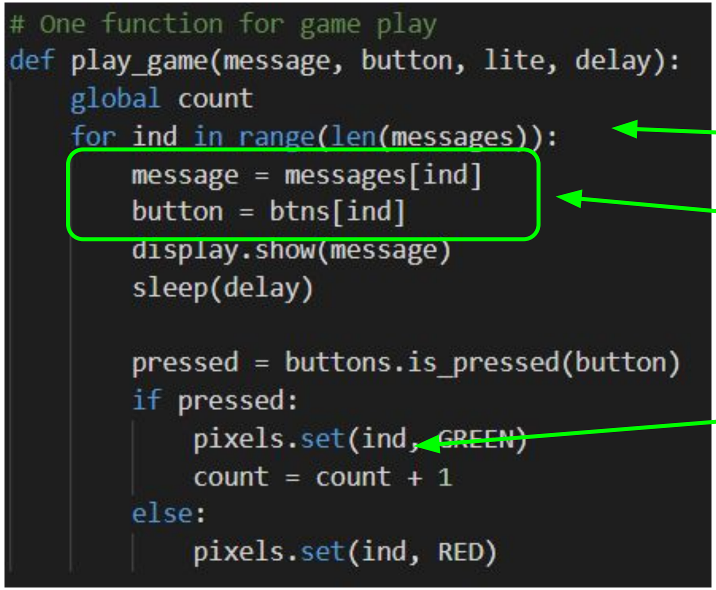
- A list for the message
- A list for the buttons
- Use your order for the game
- Be careful with your spelling and punctuation



```
'''
Assignment: Create PT Practice #4 (from Display3)
Programmers:
'''

from codex import *
from time import sleep

messages = ["Press Up", "Press Down", "Press Left", "Press Right"]
btns = [BTN_U, BTN_D, BTN_L, BTN_R]

delay = 1
count = 0
```

Your two lists should look similar to this

Step #2



```
# One function for game play
def play_game(message, button, lite, delay):
    global count
    for ind in range(len(messages)):
        message = messages[ind]
        button = btns[ind]
        display.show(message)
        sleep(delay)

        pressed = buttons.is_pressed(button)
        if pressed:
            pixels.set(ind, GREEN)
            count = count + 1
        else:
            pixels.set(ind, RED)
```

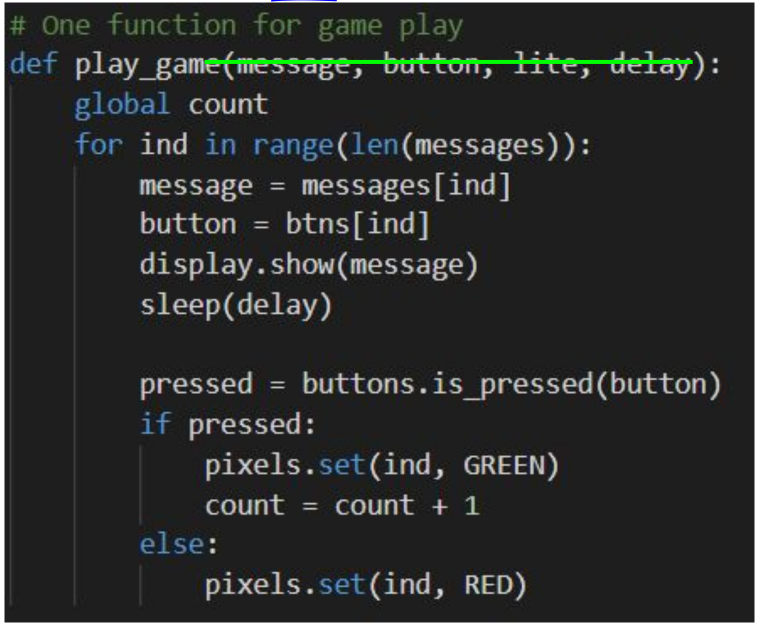
Use the lists

Modify your function to use a for loop and traverse the lists

- All code inside the function can now be inside a for loop
- Add code to get values from the lists
- The counter for the loop can be used for the pixel to turn on

Your code should look similar to this

Step #2



```
# One function for game play
def play_game(~~message, button, lite, delay~~):
    global count
    for ind in range(len(messages)):
        message = messages[ind]
        button = btns[ind]
        display.show(message)
        sleep(delay)

        pressed = buttons.is_pressed(button)
        if pressed:
            pixels.set(ind, GREEN)
            count = count + 1
        else:
            pixels.set(ind, RED)
```

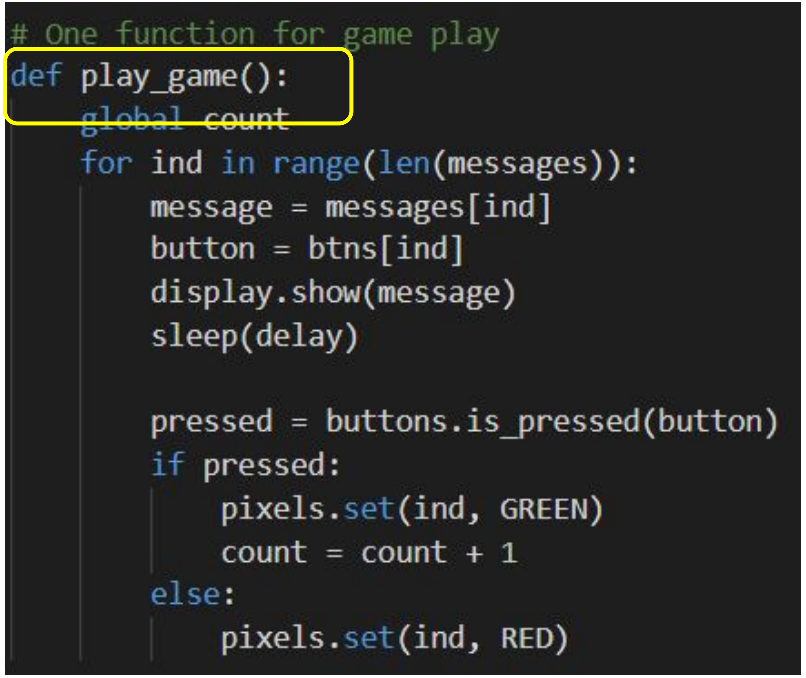
Use the lists

- Now that you are accessing information from lists, you no longer need them as parameters.
- Since you are using the loop's **ind** for the pixel, you no longer need that as a parameter either.
- And delay is a global variable that doesn't change, so you don't need it either.
- Modify the function by deleting all the parameters

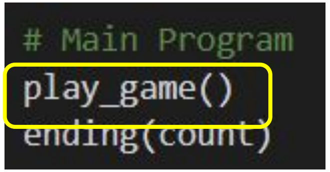
Step #2

Use the lists

- Modify the main program to call the function ONE time, without a parameter
- Run the code and make sure it is error-free



```python
# One function for game play
def play_game():
    global count
    for ind in range(len(messages)):
        message = messages[ind]
        button = btns[ind]
        display.show(message)
        sleep(delay)

        pressed = buttons.is_pressed(button)
        if pressed:
            pixels.set(ind, GREEN)
            count = count + 1
        else:
            pixels.set(ind, RED)
```



```python
# Main Program
play_game()
ending(count)
```

Step #3

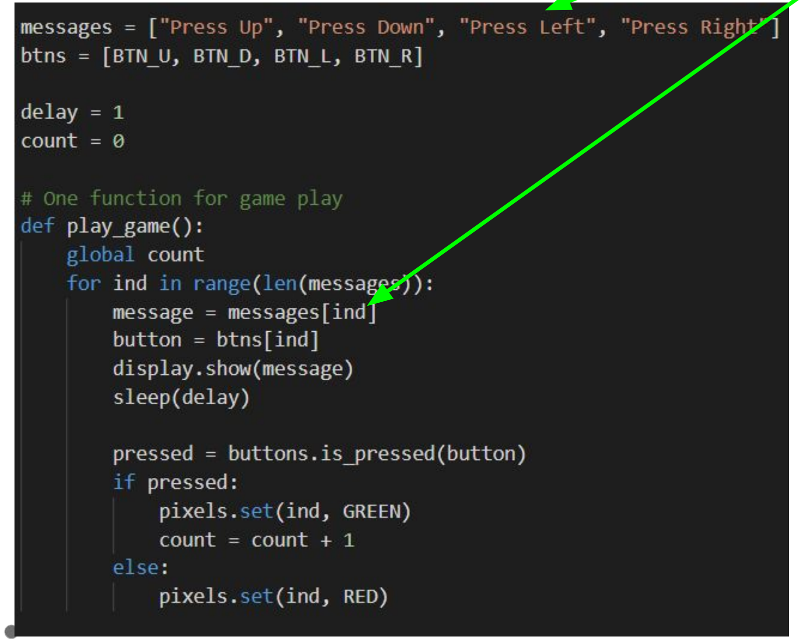
Does this program meet ALL the requirements?

- Creates a list
- Uses a list in a meaningful way
- Has a function with a parameter
  - Parameter is used in an if statement
- Function has:
  - If statement
  - Loop

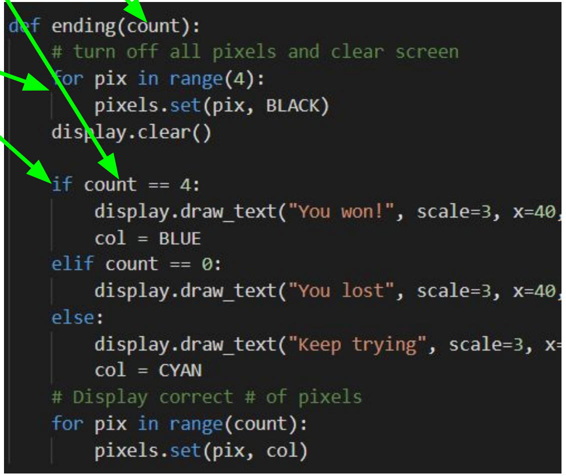
Step #3

Does this program meet ALL the requirements?

- Creates a list
- Uses a list in a meaningful way
- Has a function with a parameter
    - Parameter is used in an if statement
- Function has:
    - If statement
    - Loop



```
messages = ["Press Up", "Press Down", "Press Left", "Press Right"]
btns = [BTN_U, BTN_D, BTN_L, BTN_R]

delay = 1
count = 0

# One function for game play
def play_game():
    global count
    for ind in range(len(messages)):
        message = messages[ind]
        button = btns[ind]
        display.show(message)
        sleep(delay)

        pressed = buttons.is_pressed(button)
        if pressed:
            pixels.set(ind, GREEN)
            count = count + 1
        else:
            pixels.set(ind, RED)
```



```
def ending(count):
    # turn off all pixels and clear screen
    for pix in range(4):
        pixels.set(pix, BLACK)
    display.clear()

    if count == 4:
        display.draw_text("You won!", scale=3, x=40,
        col = BLUE
    elif count == 0:
        display.draw_text("You lost", scale=3, x=40,
    else:
        display.draw_text("Keep trying", scale=3, x=
        col = CYAN
    # Display correct # of pixels
    for pix in range(count):
        pixels.set(pix, col)
```

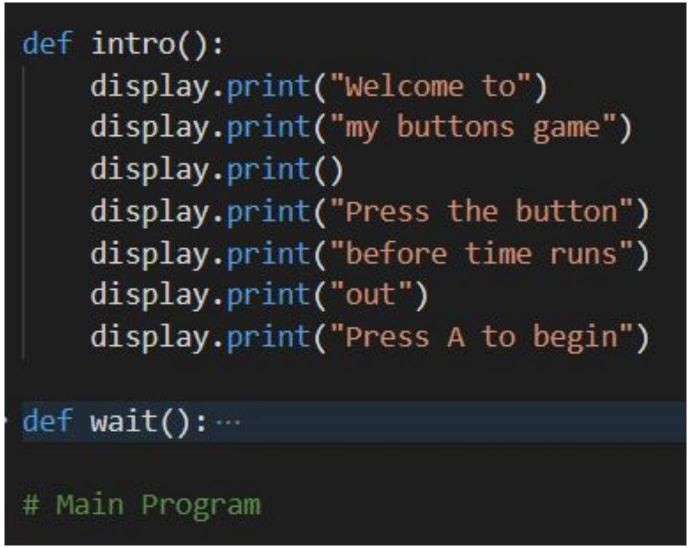
Step #3

Does this program meet ALL the requirements?

- YES!
- Even though the list is used in a different function than the one with a parameter – that is okay.
- All requirements are met!
- BUT — Can you do more???

Step #4

Create an intro function



```
def intro():
    display.print("Welcome to")
    display.print("my buttons game")
    display.print()
    display.print("Press the button")
    display.print("before time runs")
    display.print("out")
    display.print("Press A to begin")

def wait():...

# Main Program
```

It is always nice to include instructions with your code, and a message when the program ends.

- Create an intro() function that gives general instructions about the program
- Use display.print statements
- Suggestion:
    - You can also create a “wait” function that will show the instructions until a button is pressed

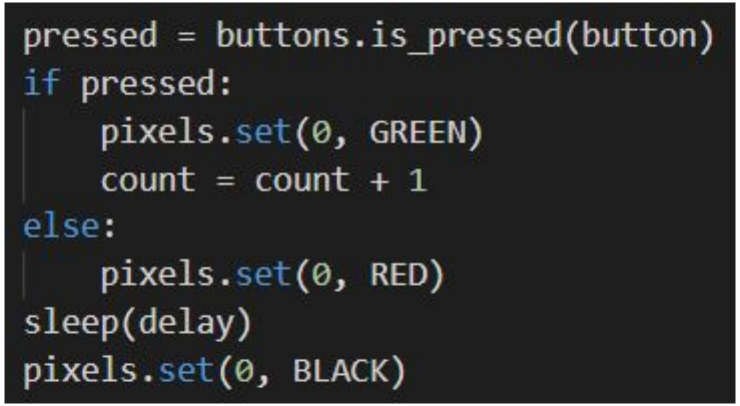
FIRIA LABS

Step #5

Do more with the list

You only have 4 pixels to light up, so it looks like you can only play the game four times.

- What if you only use one pixel for right and wrong, and turn it off after each guess so you can re-use it?
- Then you can ask for button presses several times, not just four.
- Make this modification to your code.
- One solution is shown, but there are many ways to do this.

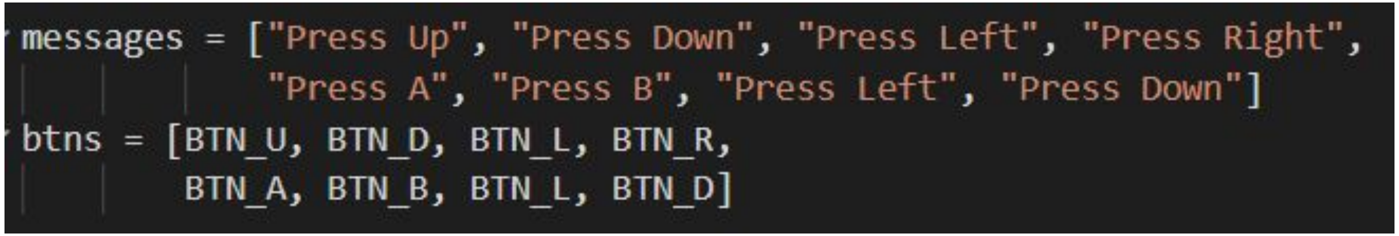


```
pressed = buttons.is_pressed(button)
if pressed:
    pixels.set(0, GREEN)
    count = count + 1
else:
    pixels.set(0, RED)
sleep(delay)
pixels.set(0, BLACK)
```

Step #5

Do more with the list

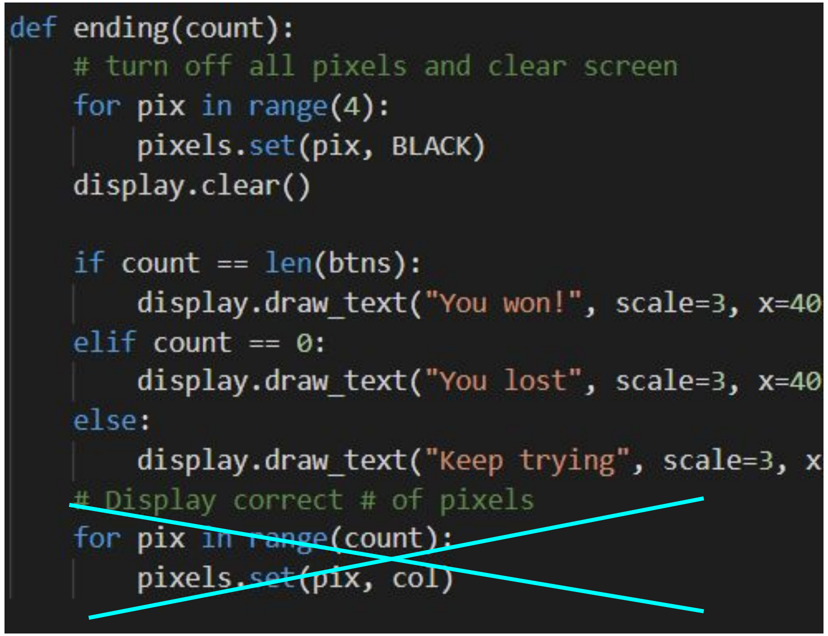
- Now add more data to both your lists to extend the game
- Make sure you match the instruction with the button to press – keep the order correct!



```
messages = ["Press Up", "Press Down", "Press Left", "Press Right",
            "Press A", "Press B", "Press Left", "Press Down"]
btns = [BTN_U, BTN_D, BTN_L, BTN_R,
        BTN_A, BTN_B, BTN_L, BTN_D]
```

Here is an example

Step #5



```
def ending(count):
    # turn off all pixels and clear screen
    for pix in range(4):
        pixels.set(pix, BLACK)
    display.clear()

    if count == len(btns):
        display.draw_text("You won!", scale=3, x=40
    elif count == 0:
        display.draw_text("You lost", scale=3, x=40
    else:
        display.draw_text("Keep trying", scale=3, x
    ~~# Display correct # of pixels~~
    ~~for pix in range(count):~~
    ~~    pixels.set(pix, col)~~
```

Do more with the list

- Modify the ending() function to indicate the correct number for right.
- You can even use the len of a list, so if you modify the list even more, you don't have to come back and change the code in the ending.

If you have code that displays the correct number in pixels, you will want to delete it, since you only have four pixels.

Step #6

Use modulo division

This is completely optional, but wouldn’t it be fun to use modulo division for something?

- You learned about it awhile ago, so let’s use it to turn on a different pixel for each question.

Step #6

Use modulo division

Question –

Given: **x % 4**

What are the possible answers?

Step #6

Use modulo division

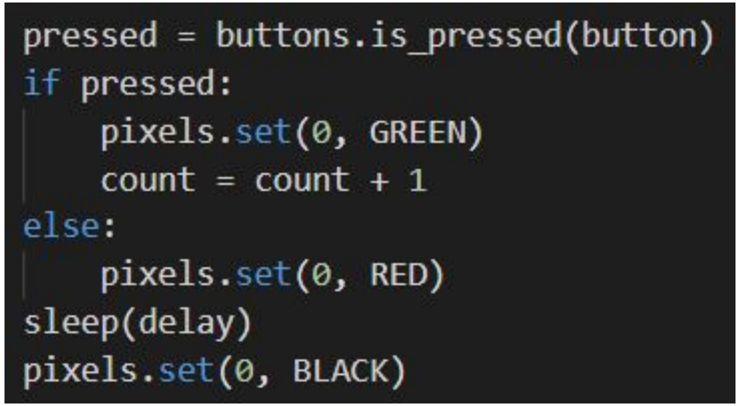
Given: **x % 4**

Did you remember that the possible answers are 0, 1, 2, and 3?

Well, these are the same numbers as the pixels!

You can use modulo division to light up pixels as the user plays the game.

Step #6



```
pressed = buttons.is_pressed(button)
if pressed:
    pixels.set(0, GREEN)
    count = count + 1
else:
    pixels.set(0, RED)
sleep(delay)
pixels.set(0, BLACK)
```

Use modulo division

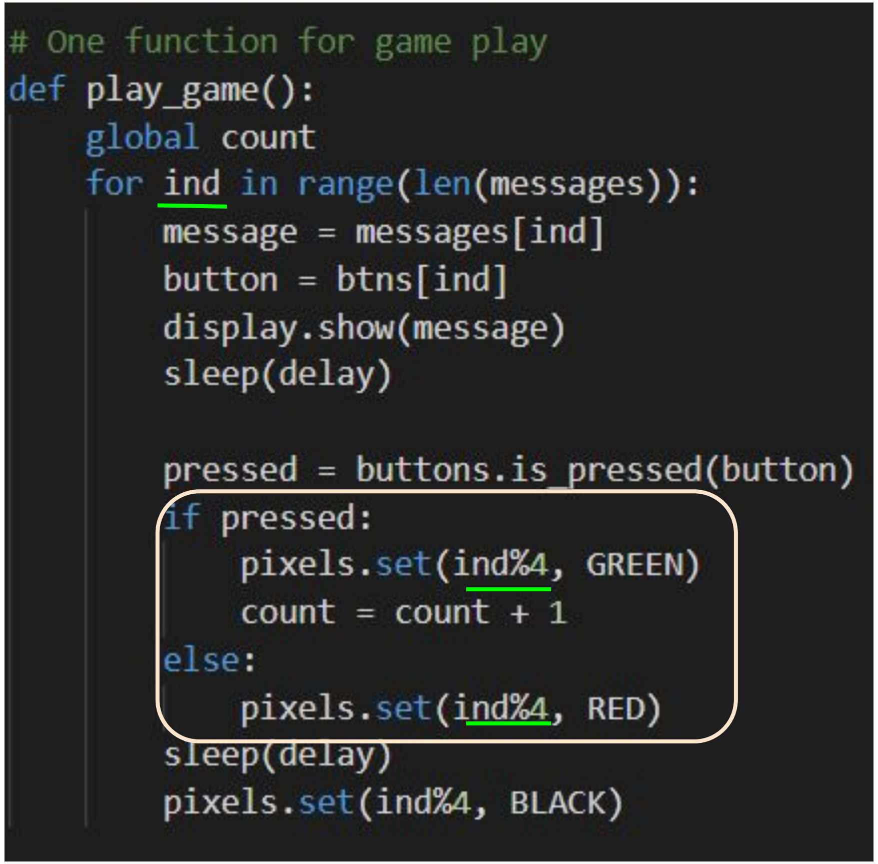
Right now you have this code to light up only one pixel for every question.

- Use the ind (or counter in the for loop) and modulo division to light up a pixel (0-4) as the questions are answered
- pixels.set(ind%4, GREEN)
- pixels.set(ind%4, RED)

Step #6

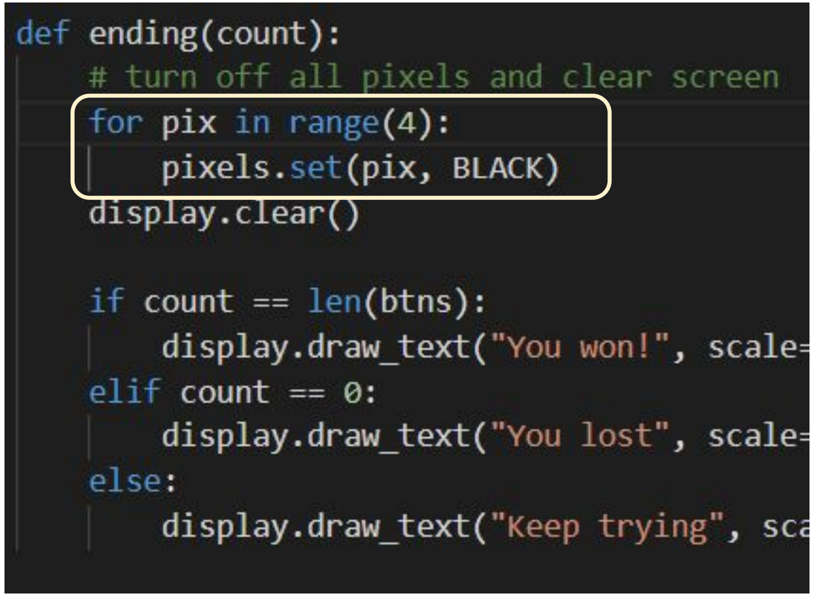
Use modulo division

With this code the user gets a different pixel to light up with each question, with the lights scrolling left to right.



```
# One function for game play
def play_game():
    global count
    for ind in range(len(messages)):
        message = messages[ind]
        button = btns[ind]
        display.show(message)
        sleep(delay)

        pressed = buttons.is_pressed(button)
        if pressed:
            pixels.set(ind%4, GREEN)
            count = count + 1
        else:
            pixels.set(ind%4, RED)
        sleep(delay)
        pixels.set(ind%4, BLACK)
```

Step #7



```
def ending(count):
    # turn off all pixels and clear screen
    for pix in range(4):
        pixels.set(pix, BLACK)
    display.clear()

    if count == len(btns):
        display.draw_text("You won!", scale=
    elif count == 0:
        display.draw_text("You lost", scale=
    else:
        display.draw_text("Keep trying", sca
```

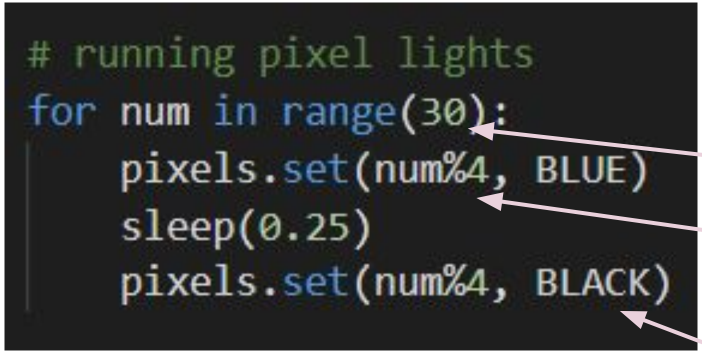
Use modulo division (again)

This is also completely optional, but wouldn't it be fun to use modulo division in the ending as well?

- Maybe add the effect of running lights as the message is shown.
- Look at parts of the ending() function
    - Code to turn all pixels BLACK

Step #7

Use modulo division (again)



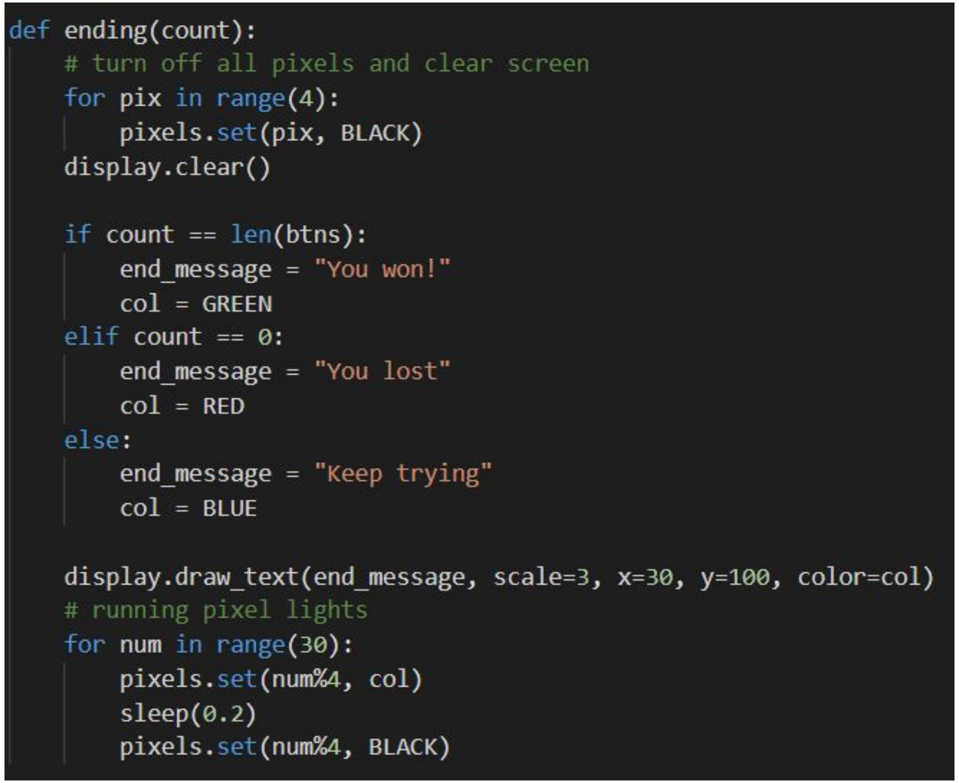
```
# running pixel lights
for num in range(30):
    pixels.set(num%4, BLUE)
    sleep(0.25)
    pixels.set(num%4, BLACK)
```

- You can do something similar to turn (and then off) all the pixels for a while to create the effect of running lights
- Choose a number for how many lights you want to "run"
- Use modulo division to scroll left to right, lighting up the pixel
- Remember to also turn off the pixel

Step #7

Variations

- Get a random color for each pixel
- Assign a value to a variable for the color, depending on how many they got correct, and use the variable in the running lights
- Any other variation you can think of – be creative!



```
def ending(count):
    # turn off all pixels and clear screen
    for pix in range(4):
        pixels.set(pix, BLACK)
    display.clear()

    if count == len(btns):
        end_message = "You won!"
        col = GREEN
    elif count == 0:
        end_message = "You lost"
        col = RED
    else:
        end_message = "Keep trying"
        col = BLUE

    display.draw_text(end_message, scale=3, x=30, y=100, color=col)
    # running pixel lights
    for num in range(30):
        pixels.set(num%4, col)
        sleep(0.2)
        pixels.set(num%4, BLACK)
```

Step #8

Test and debug

- Run your code for accuracy and bugs.
    - Run the code getting all questions correct
    - Run the code getting none of the questions correct
    - Run the code with some of the questions correct
- Does the game work correctly for both choices?
- Make any other modifications you want for this Practice PT

And now you have another Create PT practice

Congratulations!

By completing this practice project you have prepared for the PT by:

- Creating a list
- Using the list in a meaningful way
- Creating a function with a parameter
- Calling the function
- Using the parameter in an if statement
- Using sequence and selection in the function

And now you have your own create PT practice

Moving forward

You will continue to prepare for the Create PT by:

- Identifying the requirements in your code
- Pasting an image of the requirements in a document