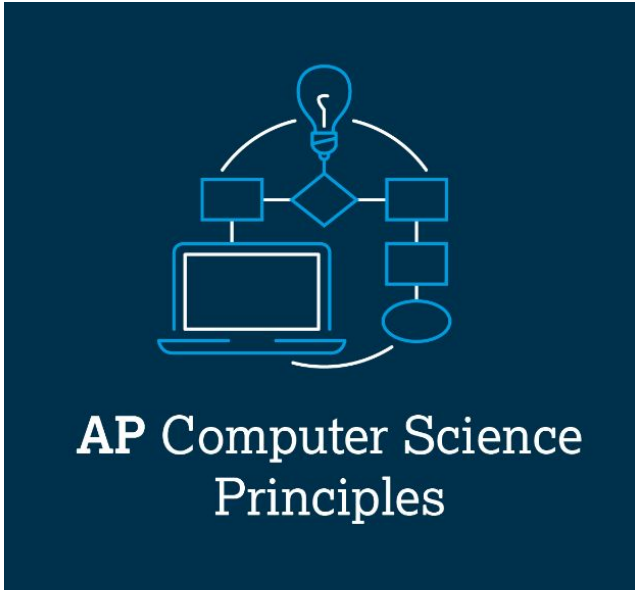
Practice #6

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

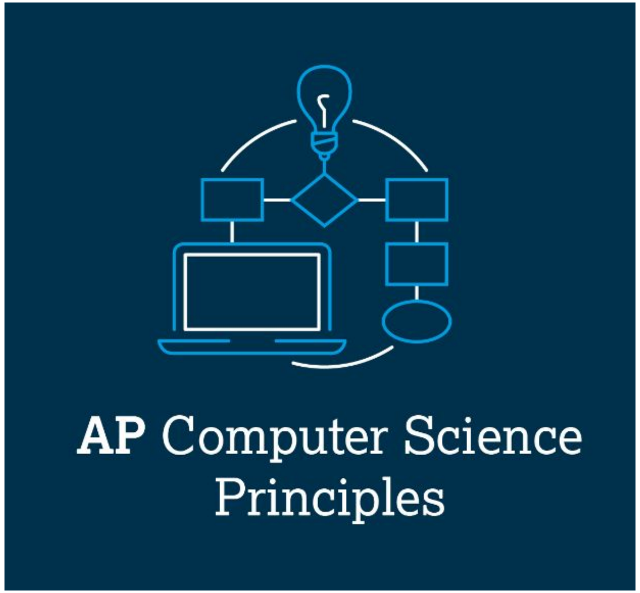


AP CSP Create Performance Task

For this project, you will:

- Start a new project that will create and use lists
- Be a fun memory game

But when you are finished, the program won't meet all the requirements for the Create PT. We will then discuss how you can modify it to meet all the requirements.



Simon Memory Game

Are you familiar with the Simon Memory game?

- A color shows and a sound is made
- With every round, a new color and sound is added to the sequence
- The player has to copy the pattern
- One wrong move and you lose

For this Practice PT, you will create this game, simplified, with just color pixels and four buttons.

Simon Memory Game

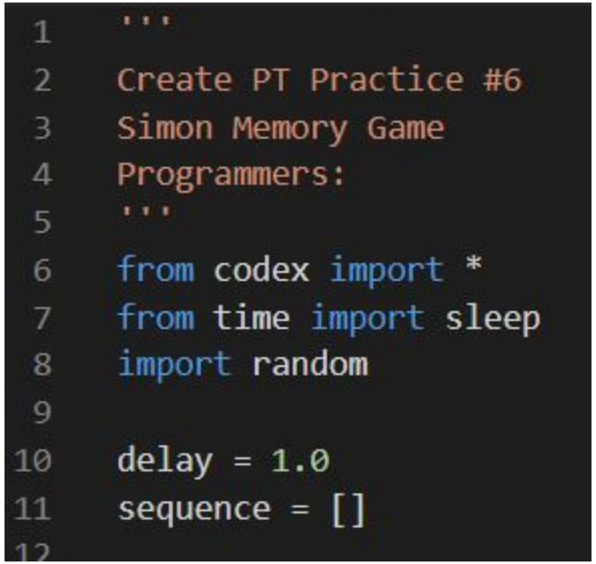
Think about the steps needed for the game

- Generate a random pixel (0-3)
- Use a list to store the pixel
- Traverse the list to display the sequence
- Player’s turn: compare each button press with the list
- If each button press is correct, add to the sequence
- Otherwise, end the game

Let’s get started!

Step #1

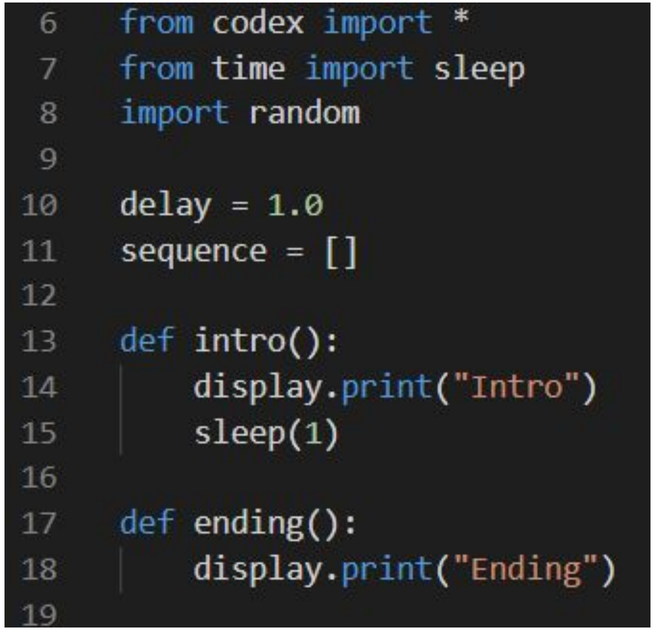
Start a new project



```
'''
Create PT Practice #6
Simon Memory Game
Programmers:
'''
from codex import *
from time import sleep
import random

delay = 1.0
sequence = []
```

- In CodeSpace go to Mission 9, Objective 8
- Start a new file
- Name the file "Create_PT_Practice6
- Add a comment block at the top
- Import the modules you will need
- Create a variable for the delay (you can start with 1 as the value, and adjust as you work on the code)
- Create a list for the sequence of pixels

Step #1

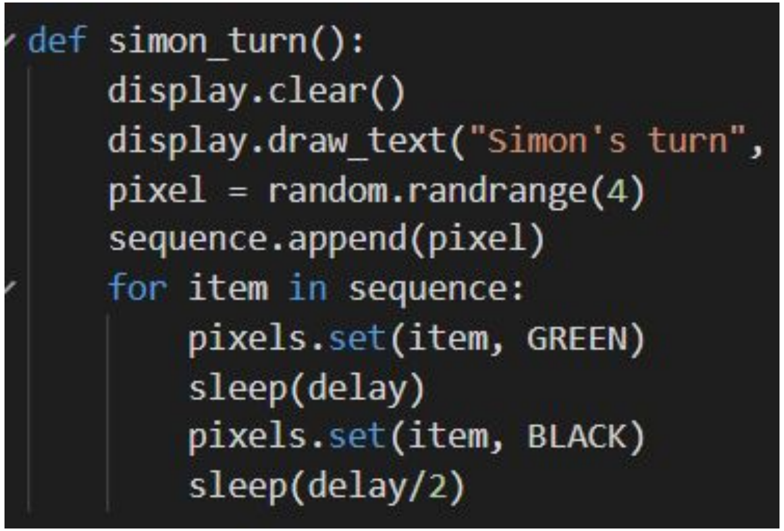


```
from codex import *
from time import sleep
import random

delay = 1.0
sequence = []

def intro():
    display.print("Intro")
    sleep(1)

def ending():
    display.print("Ending")

```

Start a new project

- Create an intro() function
- Create an ending() function
- These can be really simple right now, and you can improve them later
- Right now, you just want them to show so you know when the program begins and ends

Step #2



```
def simon_turn():
    display.clear()
    display.draw_text("Simon's turn",
    pixel = random.randrange(4)
    sequence.append(pixel)
    for item in sequence:
        pixels.set(item, GREEN)
        sleep(delay)
        pixels.set(item, BLACK)
        sleep(delay/2)
```

Create a function for Simon’s turn

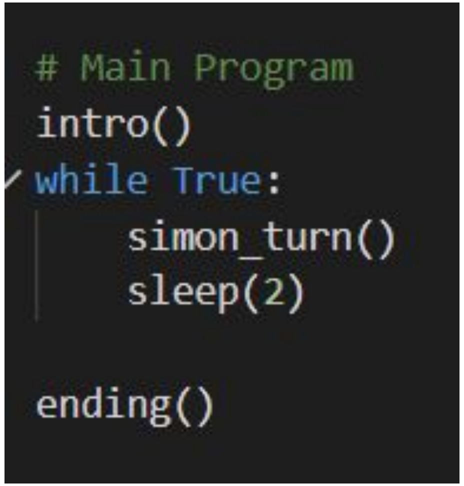
- Create a function that will:
  - Clear the screen
  - Display “Simon’s turn”
  - Get a random pixel (0-3)
  - Append to the list
  - Traverse the list and light up each pixel
- You should be able to do most of this on your own – use the code as needed

FIRIA LABS

Step #2

Create a function for Simon's turn

- Test the code before you move on
- Add a main program
  - Call the intro
  - Start a while loop
  - Call the function
  - After the loop, call the ending
- Do you see the pattern growing?
  - Each time the loop starts over, the original pixels should light, with a new one at the end

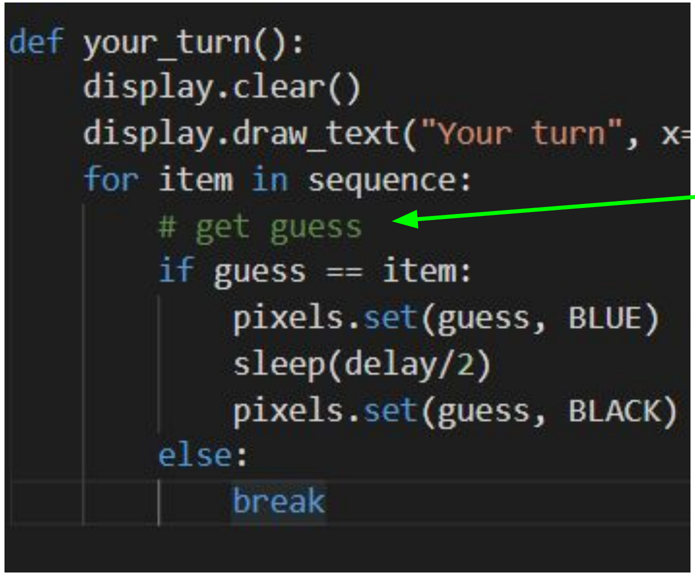


```
# Main Program
intro()
while True:
    simon_turn()
    sleep(2)

ending()
```

FIRIA LABS

Step #3

Create a function for your turn



```
def your_turn():
    display.clear()
    display.draw_text("Your turn", x=
    for item in sequence:
        # get guess
        if guess == item:
            pixels.set(guess, BLUE)
            sleep(delay/2)
            pixels.set(guess, BLACK)
        else:
            break
```

- Create a function that will:
  - Clear the display
  - Display “Your turn”
  - Traverse the list
    - Get a button press from the user
    - Compare the press with the item in the list
    - If correct, light the pixel and continue
    - Otherwise, end

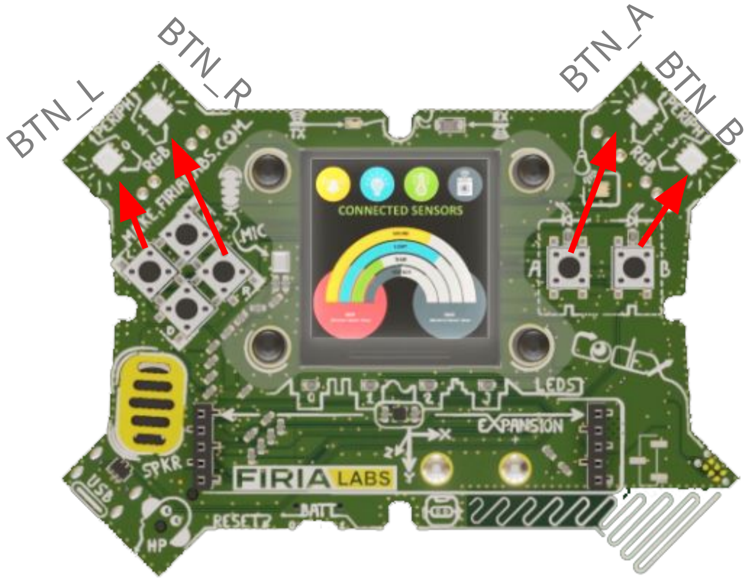
Step #4

Get the guess (button press)

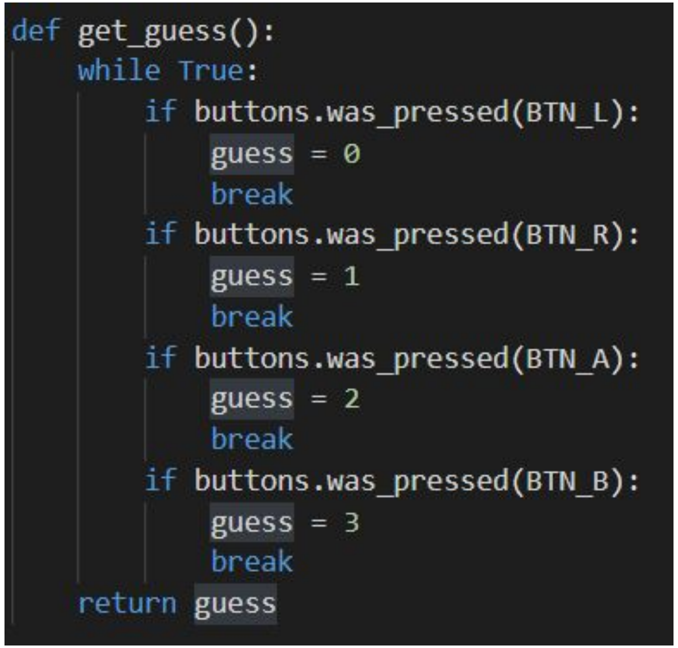
The pixel is a number, but a button press is not. You need to change a button press to a number.

- One way to do this is to use an if statement
- Decide what buttons you want the user to press

Here is what I am using, but you can use your own button configuration:



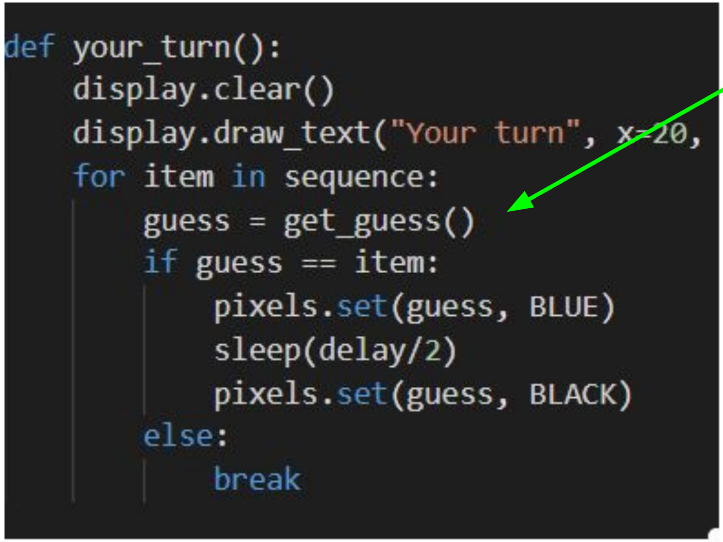
Step #4 Get the guess (button press)



```
def get_guess():
    while True:
        if buttons.was_pressed(BTN_L):
            guess = 0
            break
        if buttons.was_pressed(BTN_R):
            guess = 1
            break
        if buttons.was_pressed(BTN_A):
            guess = 2
            break
        if buttons.was_pressed(BTN_B):
            guess = 3
            break
    return guess
```

- Create a function for this
- Use a while loop
- Assign the pixel number to the button press
- Break after each possible button press
- At the end of the function
    - return guess

Step #4 Get the guess (button press)

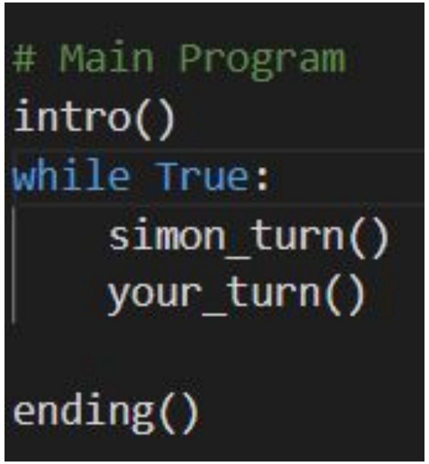


```
def your_turn():
    display.clear()
    display.draw_text("Your turn", x=20,
    for item in sequence:
        guess = get_guess()
        if guess == item:
            pixels.set(guess, BLUE)
            sleep(delay/2)
            pixels.set(guess, BLACK)
        else:
            break
```

- Call this new function in **your_turn()** function
- Remember – it returns a value, so it must be part of an assignment statement

Step #4

Get the guess (button press)



```
# Main Program
intro()
while True:
    simon_turn()
    your_turn()

ending()
```

- Call **your_turn()** in the main program
- How does the program work?
- If you guess all the buttons correctly, does it keep going?
- If you press a wrong button, does it stop?

Step #5

Stop the game

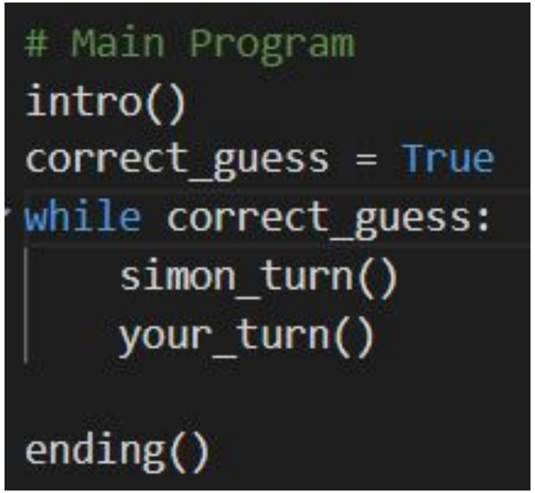
Hopefully you noticed that the program works correctly as long as the correct buttons are pressed. But… when the wrong button is pressed, simon’s turn keeps going.

- The while True loop needs to change so that it ends when the wrong button is pressed
- You can use the same technique that you used in the Practice Extra lesson

Step #5

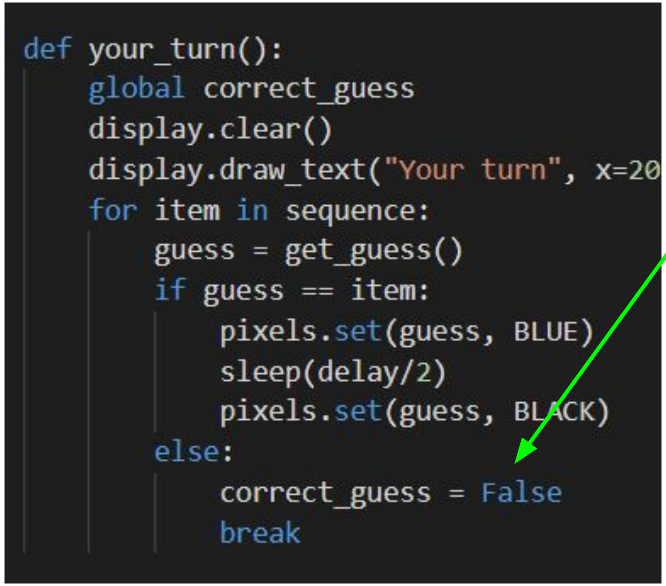
Stop the game

- Change the while loop in the main program to use a Boolean variable



```
# Main Program
intro()
correct_guess = True
while correct_guess:
    simon_turn()
    your_turn()

ending()
```

Step #5



```
def your_turn():
    global correct_guess
    display.clear()
    display.draw_text("Your turn", x=20
    for item in sequence:
        guess = get_guess()
        if guess == item:
            pixels.set(guess, BLUE)
            sleep(delay/2)
            pixels.set(guess, BLACK)
        else:
            correct_guess = False
            break
```

Stop the game

- In **your_turn()**, change the value of **correct_guess** to **False** if the button pressed is incorrect
- It is a global variable – remember what you need to do?
- Test the program again
- Does the game
  - keep going while guesses are correct?
  - stop working when guess is incorrect?

Step #6

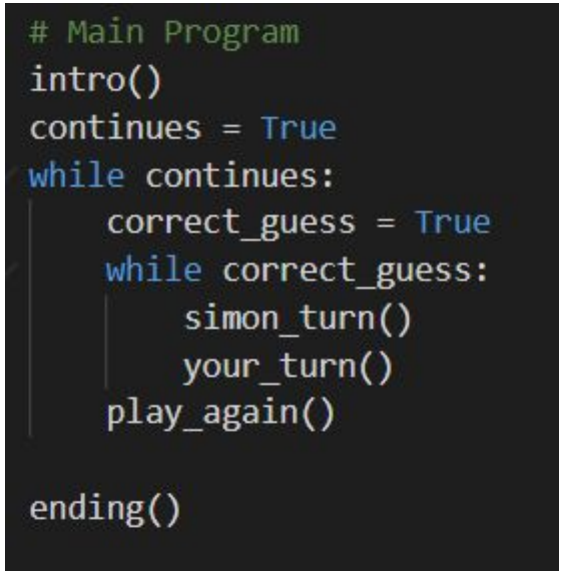
Play again

If you finished the Practice Extra lesson, you learned about using a Boolean variable and while loop to play the game again.

- Can you use that technique to play the game again without restarting the code?
- You already have one Boolean variable and while loop, but you can have another

Step #6

Play again



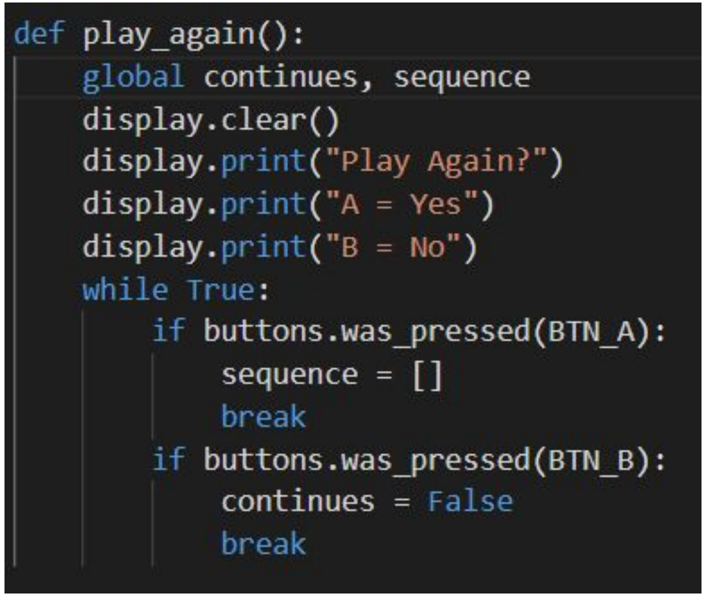
```
# Main Program
intro()
continues = True
while continues:
    correct_guess = True
    while correct_guess:
        simon_turn()
        your_turn()
    play_again()

ending()
```

- Create another Boolean variable in the main program
- Use another while loop in the main program
- Be careful with your indenting!!

Step #6

Play again

- Create the **play_again()** function, just like the ones from the Practice Extra lesson
  - If button A is pressed, you want to start over, so initialize the **sequence** list to empty
  - If button B is pressed, change the value of **continues** to False
  - Both **sequence** and **continues** are global



```
def play_again():
    global continues, sequence
    display.clear()
    display.print("Play Again?")
    display.print("A = Yes")
    display.print("B = No")
    while True:
        if buttons.was_pressed(BTN_A):
            sequence = []
            break
        if buttons.was_pressed(BTN_B):
            continues = False
            break
```

Step #7

Create PT Requirements

- Do you remember what the requirements are 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
- Which ones are met in this program?

FIRIA LABS

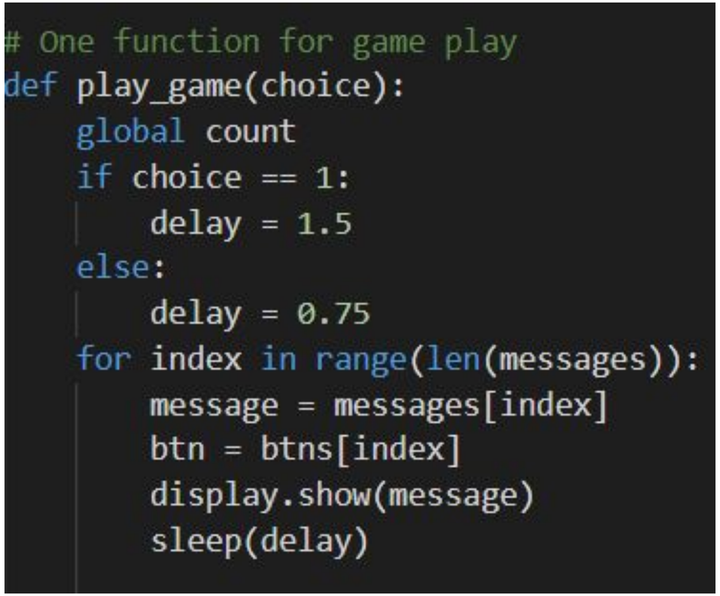
Step #7

- Create PT 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

Create PT Requirements

- The game has a list that is created and used in a meaningful way, but it still needs a function with a parameter, loop and if statement.
- What are some possibilities?
- Look at Create_PT_Practice3 and Create_PT_Practice4

Step #7



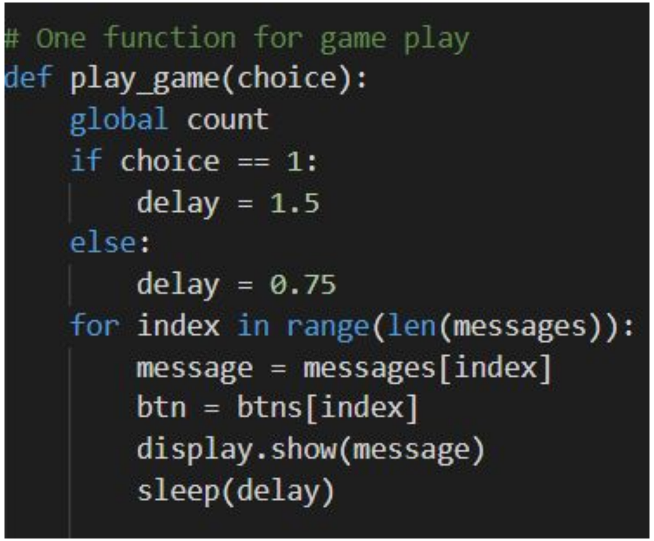
```
# One function for game play
def play_game(choice):
    global count
    if choice == 1:
        delay = 1.5
    else:
        delay = 0.75
    for index in range(len(messages)):
        message = messages[index]
        btn = btns[index]
        display.show(message)
        sleep(delay)
```

Create PT Requirements

- Look at Create_PT_Practice3
- This program gave the user a choice between easy and hard
- The choice was passed to a parameter
- The parameter was used in an if statement to set the amount of delay
- The function has a loop to traverse the list

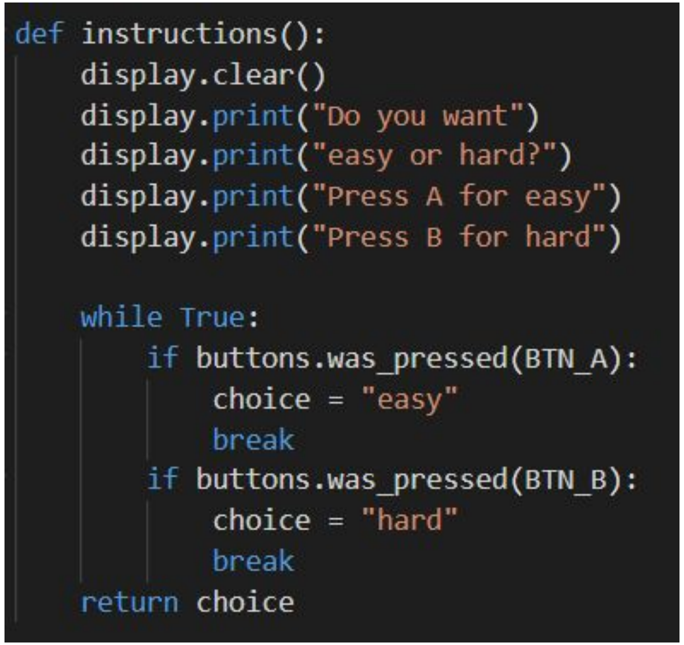
Step #7

Create_PT_Practice3

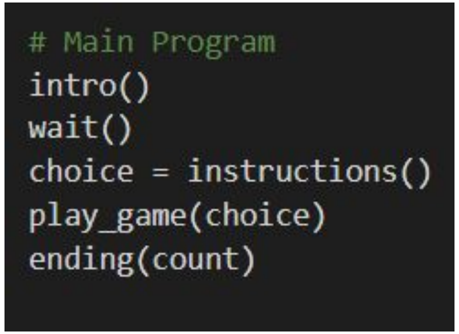


```python
# One function for game play
def play_game(choice):
    global count
    if choice == 1:
        delay = 1.5
    else:
        delay = 0.75
    for index in range(len(messages)):
        message = messages[index]
        btn = btns[index]
        display.show(message)
        sleep(delay)
```

Function with parameter, if and loop



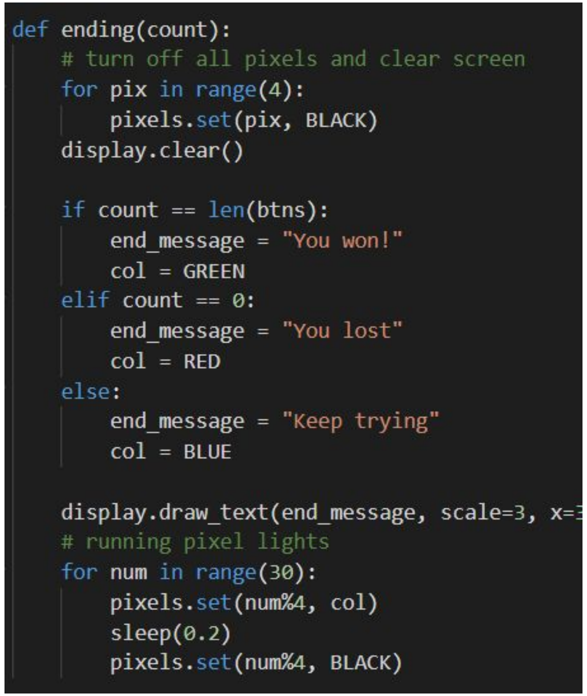
```python
def instructions():
    display.clear()
    display.print("Do you want")
    display.print("easy or hard?")
    display.print("Press A for easy")
    display.print("Press B for hard")

    while True:
        if buttons.was_pressed(BTN_A):
            choice = "easy"
            break
        if buttons.was_pressed(BTN_B):
            choice = "hard"
            break
    return choice
```



```python
# Main Program
intro()
wait()
choice = instructions()
play_game(choice)
ending(count)
```

Function call

Step #7



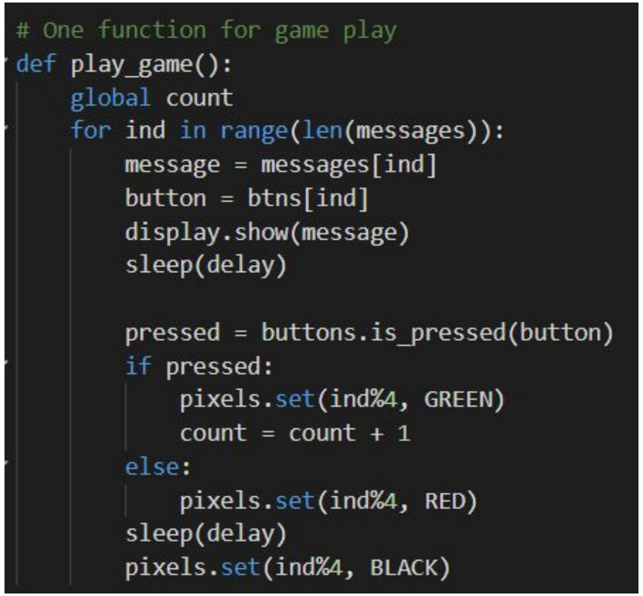
```python
def ending(count):
    # turn off all pixels and clear screen
    for pix in range(4):
        pixels.set(pix, BLACK)
    display.clear()

    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=
    # running pixel lights
    for num in range(30):
        pixels.set(num%4, col)
        sleep(0.2)
        pixels.set(num%4, BLACK)
```

Create PT Requirements

- Look at Create_PT_Practice4
- This program included a global variable for **count**
- **count** was incremented for every correct "guess"
- **count** was passed to a parameter in ending() or results() and used in an if statement
- The function used a loop for running pixel lights

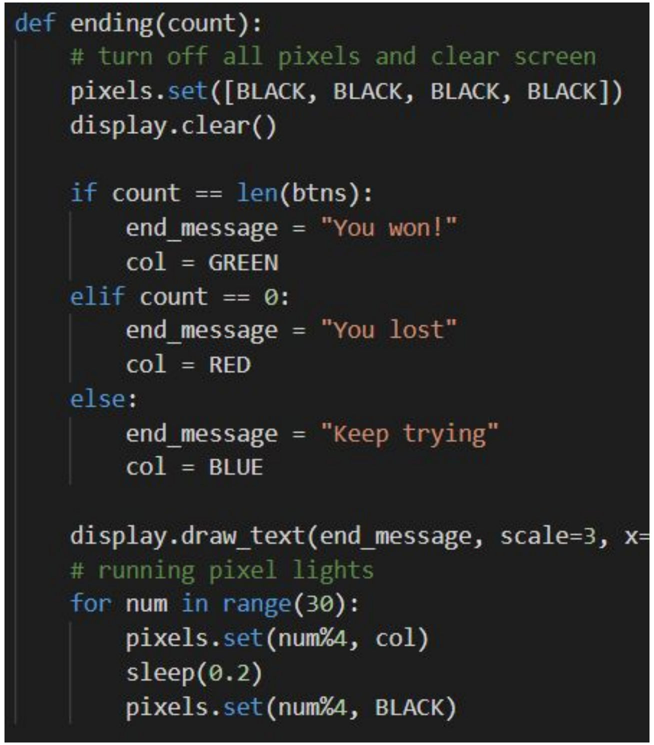
Step #7

Create_PT_Practice4



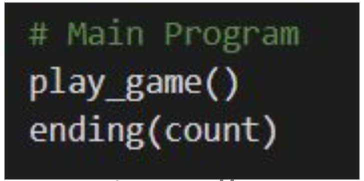
```python
# One function for game play
def play_game():
    global count
    for ind in range(len(messages)):
        message = messages[ind]
        button = btns[ind]
        display.show(message)
        sleep(delay)

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

Count is incremented



```python
def ending(count):
    # turn off all pixels and clear screen
    pixels.set([BLACK, BLACK, BLACK, 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=
    # running pixel lights
    for num in range(30):
        pixels.set(num%4, col)
        sleep(0.2)
        pixels.set(num%4, BLACK)
```

Function with parameter, if and loop



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

Function call

Step #7

- Create PT 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

Create PT Requirements

- You could add either of these coding elements to your game to meet the requirements:
  - Add a choice of easy or hard
  - Add a counter and results() function that uses the counter as a parameter in an if statement, and add a loop for the lights
- You do not need to add this to your code now. It is an option for the actual Create 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 (Mission 7)
- Using the list in a meaningful way
- Creating a function with a parameter
- Calling the function
- Using the parameter in an if statement (my_choice)
- Using sequence and selection in the function

FIRIA LABS