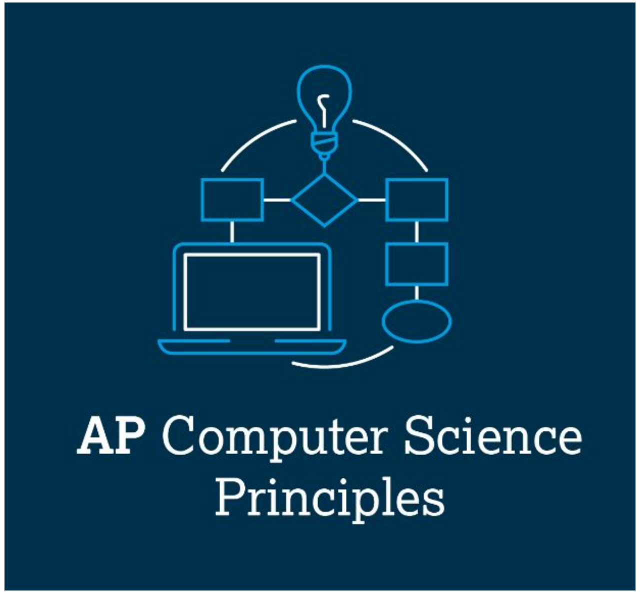
Practice #3

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

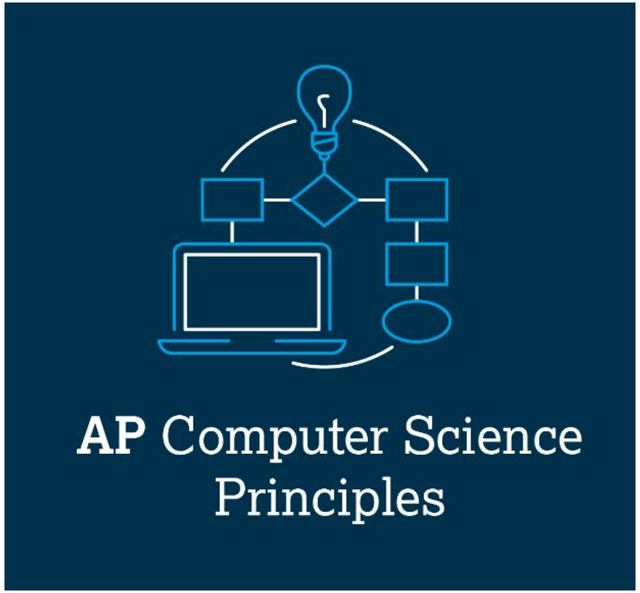


AP CSP Create Performance Task

For this project, you will:

- Start with a program that doesn't yet meet the requirements
- Modify it
- Add functions to it

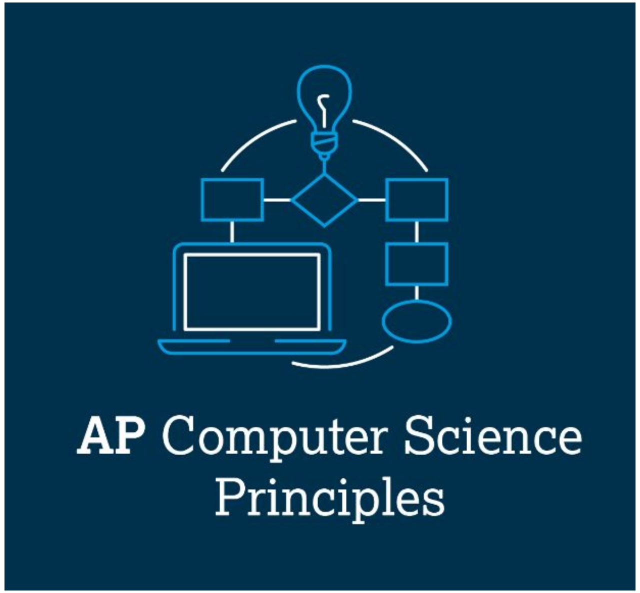
So when you are finished, you will have a program that meets all the requirements for the Create PT



AP CSP Create Performance Task

In the Display2 (from part 2 - Functions, Parameters and Local Variables), you created a cool program with parameters, 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



Step #1

Open your project "Display2"

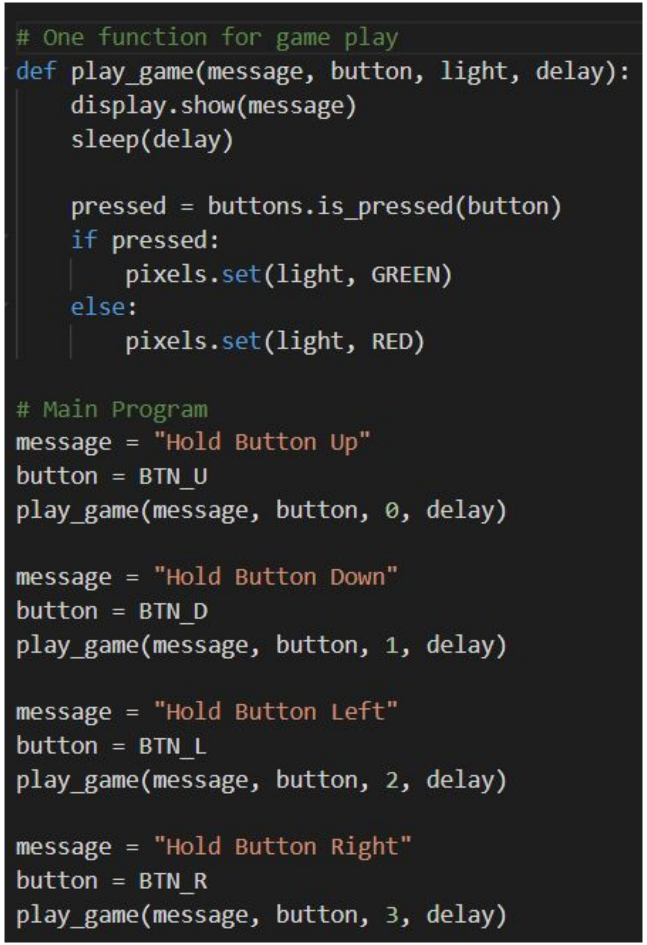
- Do a "Save As" to give your project a descriptive name
  - For today's project, you can call it Practice_PT_3
- Use a comment block at the top to include your name, date, partner, description of project, etc.

Step #2

Create two lists

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

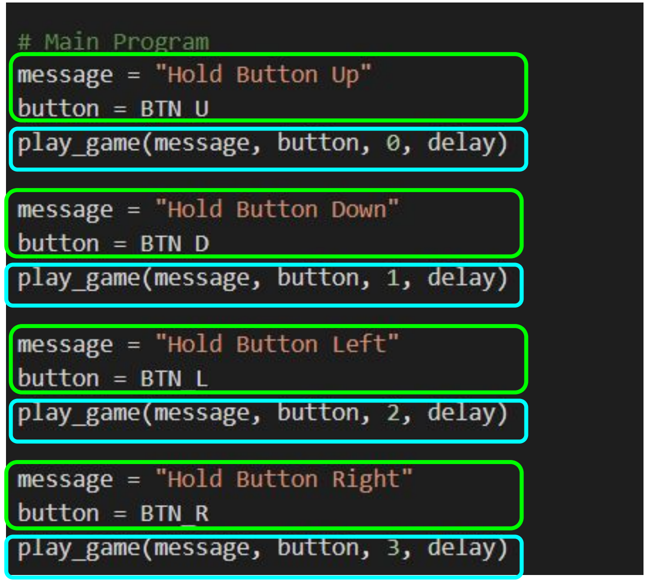
- When you traversed a list, you learned that you can use an index to access the data from two lists.
- You will do the same thing here.



```python
# 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

Create two lists



```
# 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)
```

The main program should look similar to this example

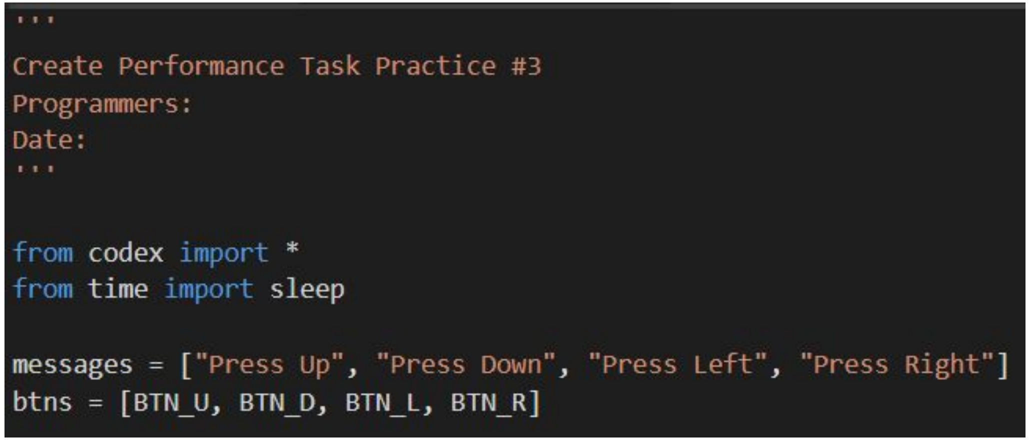
- The information is assigned to variables and then passed as arguments to the function.
- This is repeated four times
- You can put the information into two parallel lists.
- Then, using a loop, you can traverse the list to play the game.

Step #2

Create two lists

At the top of your code, create two lists

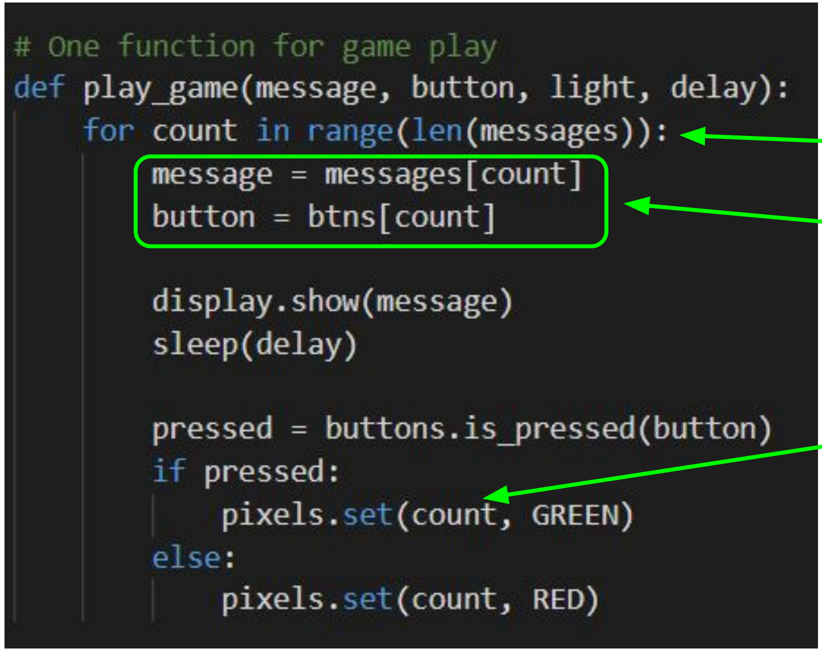
- A list for the message
- A list for the buttons
- Use your order for the game
- Be careful with your spelling and punctuation



```
'''
Create Performance Task Practice #3
Programmers:
Date:
'''

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]
```

Your two lists should look similar to this

Step #2



```
# One function for game play
def play_game(message, button, light, delay):
    for count in range(len(messages)):
        message = messages[count]
        button = btns[count]

        display.show(message)
        sleep(delay)

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

Your code should look similar to this

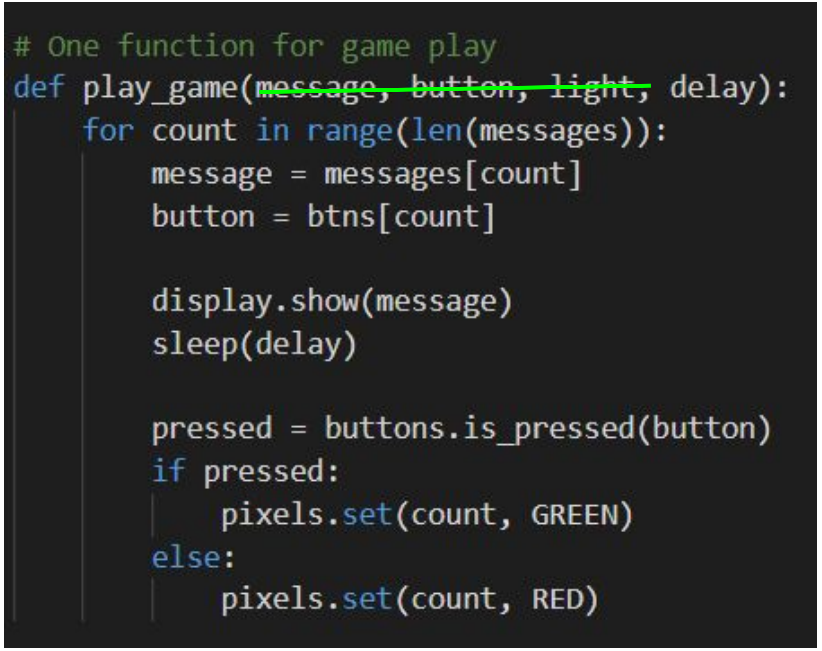
Use the lists

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

FIRIA LABS

Step #2



```
# One function for game play
def play_game(~~message, button, light,~~ delay):
    for count in range(len(messages)):
        message = messages[count]
        button = btns[count]

        display.show(message)
        sleep(delay)

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

Use the lists

Now that you are accessing information from lists, you no longer need them as parameters.

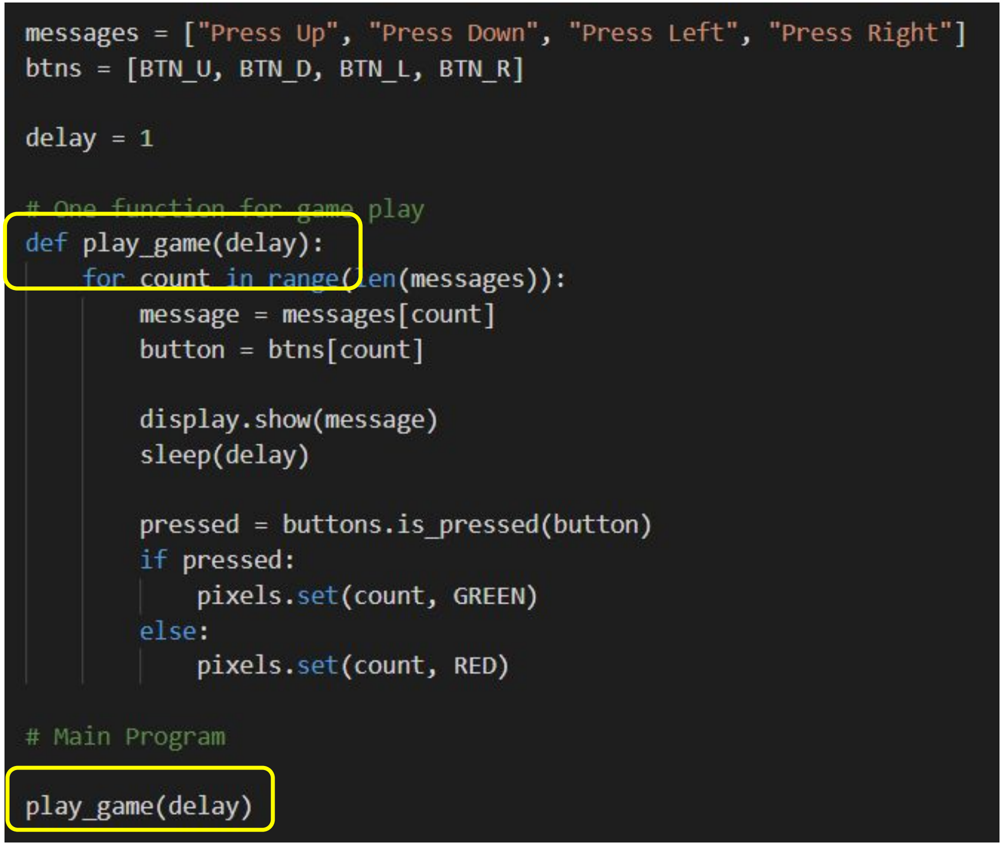
Since you are using the loop's count for the pixel, you no longer need that as a parameter either.

- Modify the function by deleting the first three parameters

Step #2

Use the lists

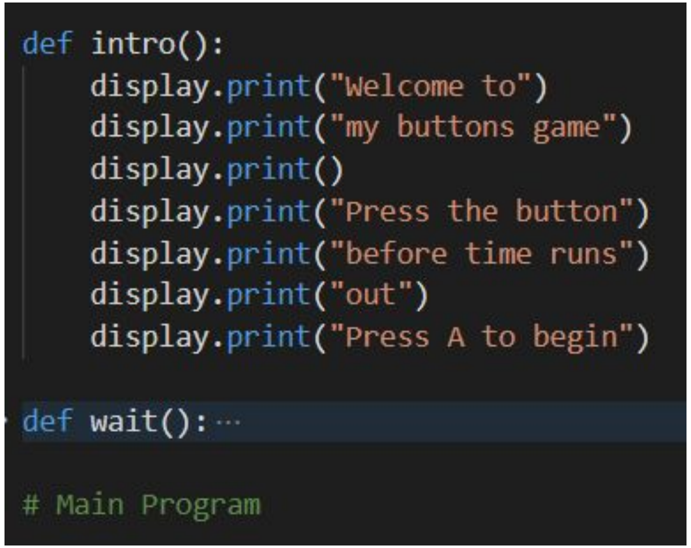
- Modify the main program to call the function ONE time, with ONE parameter
- Run the code and make sure it is error-free



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

delay = 1

# One function for game play
def play_game(delay):
    for count in range(len(messages)):
        message = messages[count]
        button = btns[count]

        display.show(message)
        sleep(delay)

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

# Main Program

play_game(delay)
```

Your code should look similar to this

Step #3



```
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
```

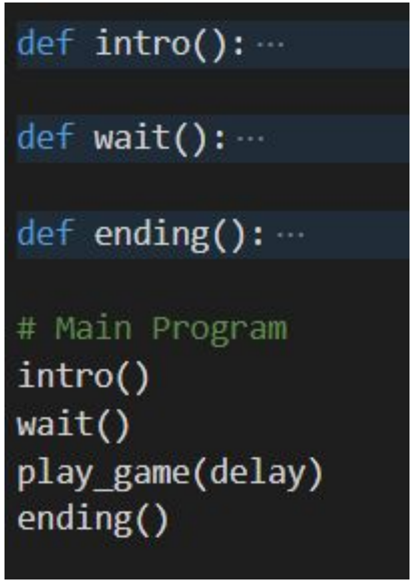
Create an intro function

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

Step #3

Create an ending function



```
def intro(): ···

def wait(): ···

def ending(): ···

# Main Program
intro()
wait()
play_game(delay)
ending()
```

It is also nice to include a message when the program ends, so the user knows it has ended.

- Create an ending() function that gives lets the user know the program has ended
- Use display.print statements

Call the functions in the main program

Step #4

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

Review the requirements:

You included the requirement of creating and using lists.

You included the requirement of an if statement and a loop.

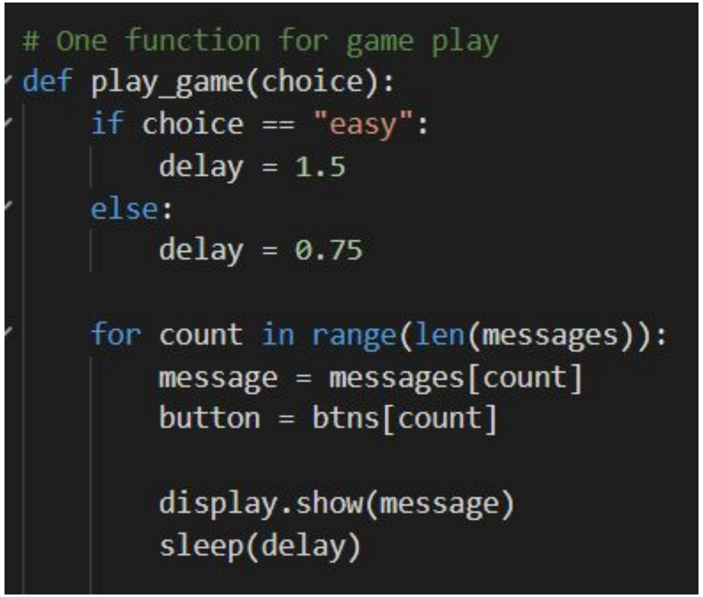
However, even though the function has a parameter, it isn't used in a condition.

Step #4

Include an if statement using the parameter

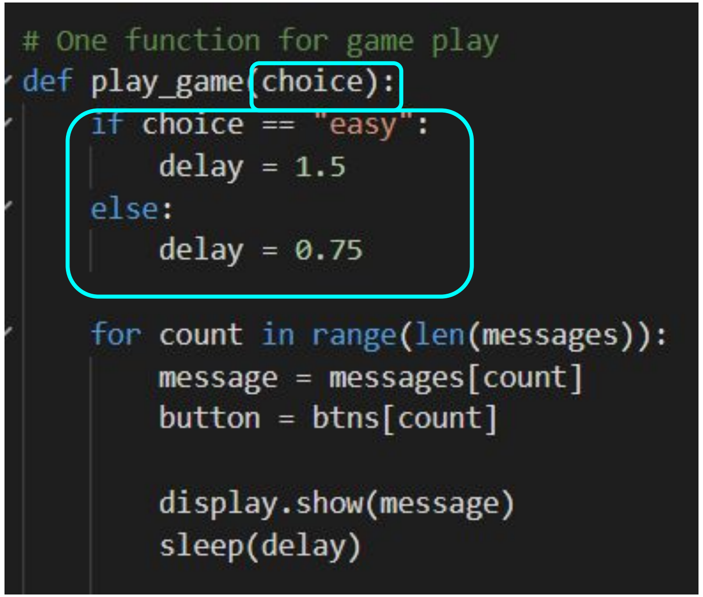
The easiest way to do this is to give the user a choice of easy or hard game.

- An easy game can have a longer delay
- A hard game can have a shorter delay
- You can use an if statement in the function to set the value of delay



```python
# One function for game play
def play_game(choice):
    if choice == "easy":
        delay = 1.5
    else:
        delay = 0.75

    for count in range(len(messages)):
        message = messages[count]
        button = btns[count]

        display.show(message)
        sleep(delay)
```

Step #4



```
# One function for game play
def play_game(choice):
    if choice == "easy":
        delay = 1.5
    else:
        delay = 0.75

    for count in range(len(messages)):
        message = messages[count]
        button = btns[count]

        display.show(message)
        sleep(delay)
```

Include an if statement using the parameter

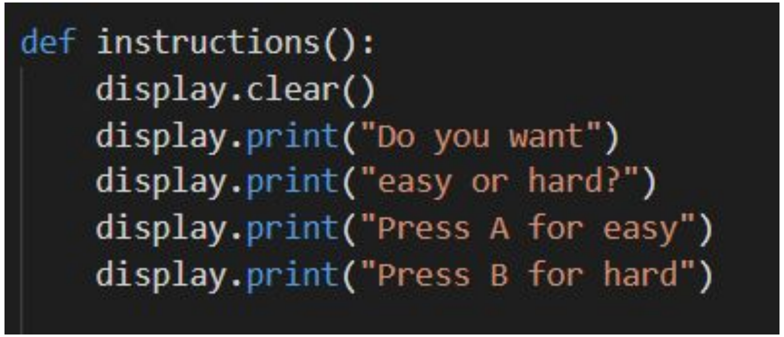
Modify the play_game function to use "choice" as the parameter

- Use an if statement to assign a value to delay.
- The values shown are examples. You can determine the amount of wait time for your easy and hard game.

Step #4

Ask the user for their choice

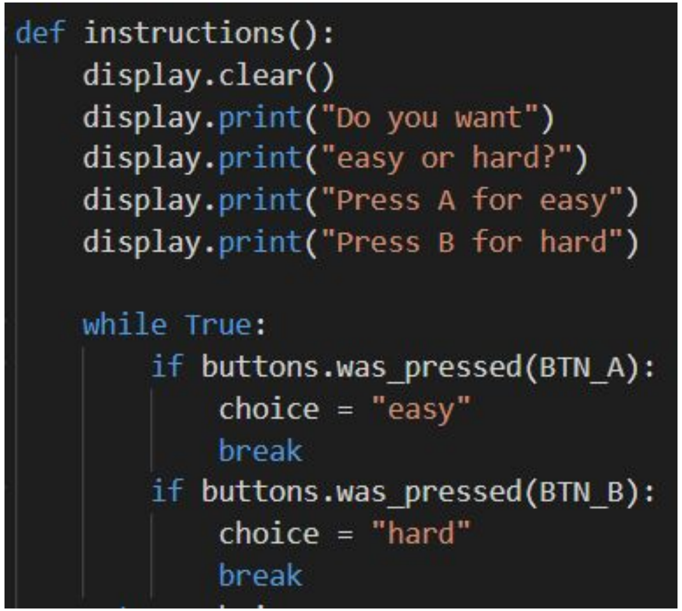
- Now you have to add code in the main program to ask the user if they want an easy or hard game.
- You can create a function for this.
- Use your own words and buttons.



```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")
```

Step #4

Ask the user for their choice

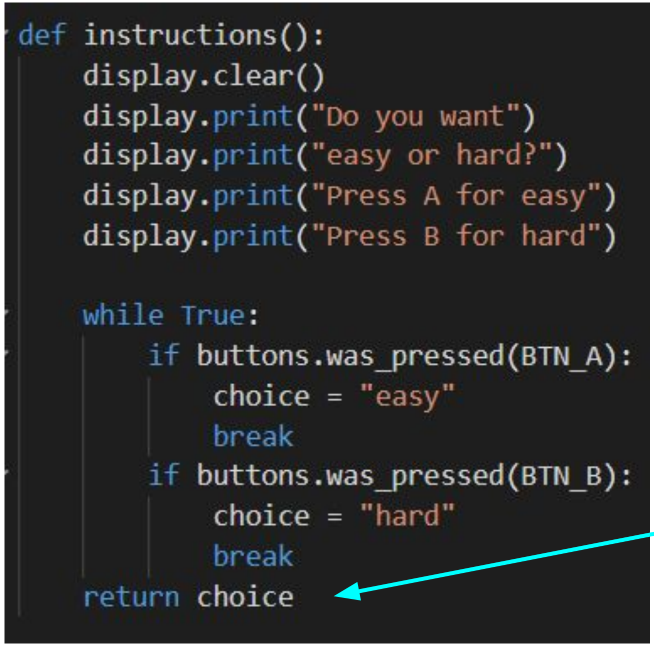


```
def instructions():
    display.clear()
    display.print("Do you want")
    display.print("easy or hard?")
    display.print("Press A for easy")
    display.print("Press B for hard")

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

- But how are you going to get their answer?
- Of course you will use a while True loop and wait for an answer.
- But you also need assign a value to choice that you can pass as an argument to the function.

Step #4

Ask the user for their choice



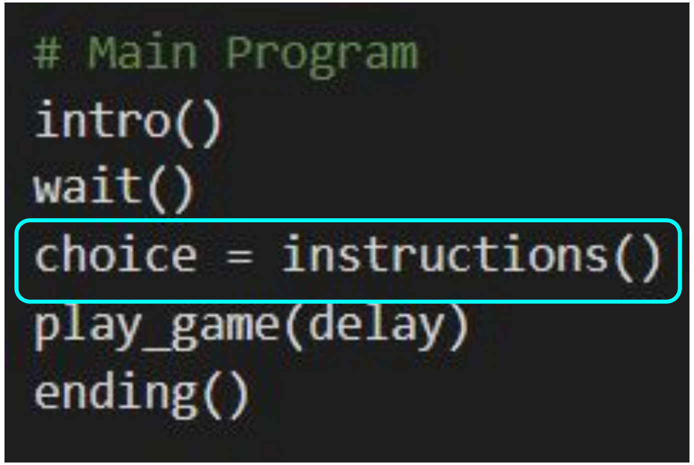
```python
def instructions():
    display.clear()
    display.print("Do you want")
    display.print("easy or hard?")
    display.print("Press A for easy")
    display.print("Press B for hard")

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

- Wait!
- You learned in the last lesson that a variable used only in the function is local and not accessible anywhere else.
- Choice needs to be accessed in the main program.
- So … return it!

Step #4

Ask the user for their choice

- When you call the function, remember to assign the returned value to a variable



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

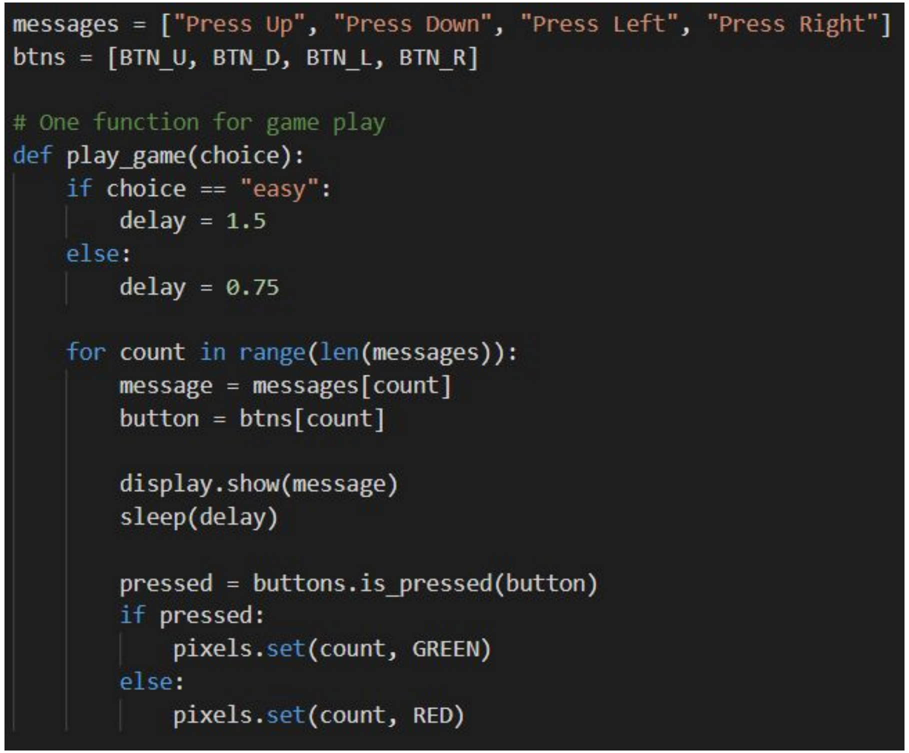
FIRIA LABS

Step #5

Meet the requirements?

Look at this part of your code. Does the code meet all the requirements?

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



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

# One function for game play
def play_game(choice):
    if choice == "easy":
        delay = 1.5
    else:
        delay = 0.75

    for count in range(len(messages)):
        message = messages[count]
        button = btns[count]

        display.show(message)
        sleep(delay)

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

Step #6

Test and debug

- Run your code for accuracy and bugs.
- Run for easy
- Run for hard
- Does the game work correctly for both choices?
- Make sure you test all buttons thoroughly.

And now you have your own 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 sequence and selection in the function
- Using the parameter in an if statement

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