Functions and Global Variables

A supplemental lesson after Mission 9

FIRIA LABS

Warm-up

Functions, parameters and local variables - Part 1

Remember when …

- Answer the warm-up questions on the assignment.

Review

Functions, parameters and local variables

Using Functions

For a long time now, since early in the missions, you have been encouraged to use functions in your code.

- When you first started using functions, you identified places in your code that were repeated.
- You created a function for the repeated code
    - Gave it a name
    - Coded the function
    - Called the function

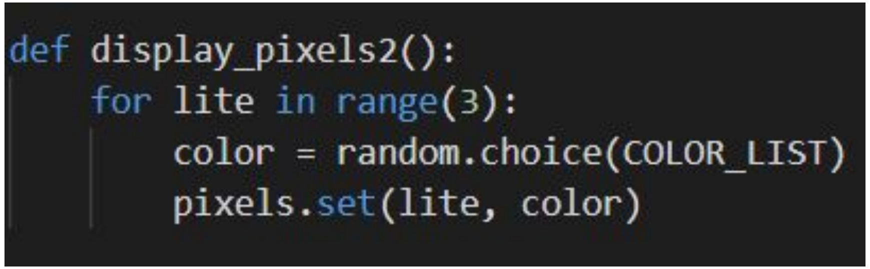
Functions and parameters

More recently, you created and used functions with parameters

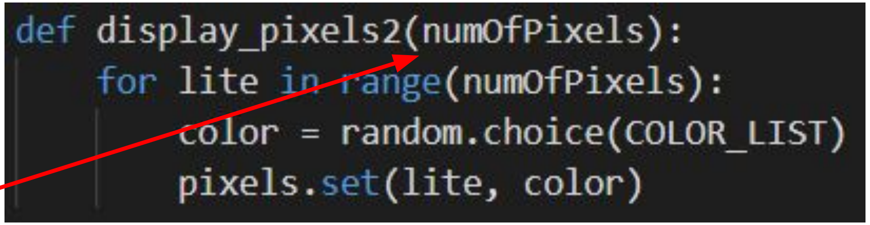
- **Parameter:** information the function needs to complete the task.
- Parameters make functions more flexible – you can use them for many different values and variables, instead of creating a function for every instance.

Functions and parameters

- For example, the first function turns on all four pixels.
- Instead of creating three more functions, from turning on one pixel to turning on three pixels, you can create a function.
- The parameter is the number of pixels to turn on.
- The information you give the function is called a **parameter.**



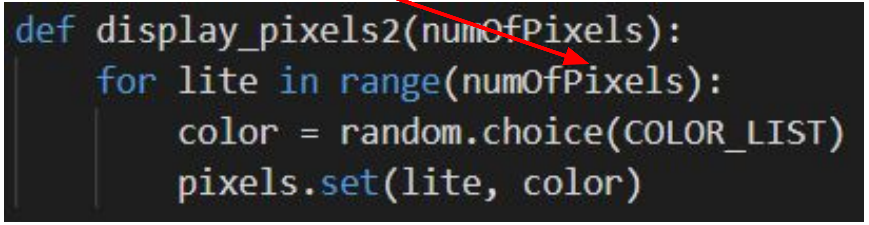
```
def display_pixels2():
    for lite in range(3):
        color = random.choice(COLOR_LIST)
        pixels.set(lite, color)
```



```
def display_pixels2(numOfPixels):
    for lite in range(numOfPixels):
        color = random.choice(COLOR_LIST)
        pixels.set(lite, color)
```

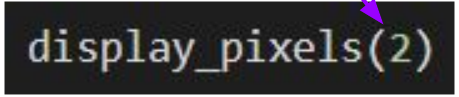
Functions and parameters

- You use the parameter in the function code to complete the task



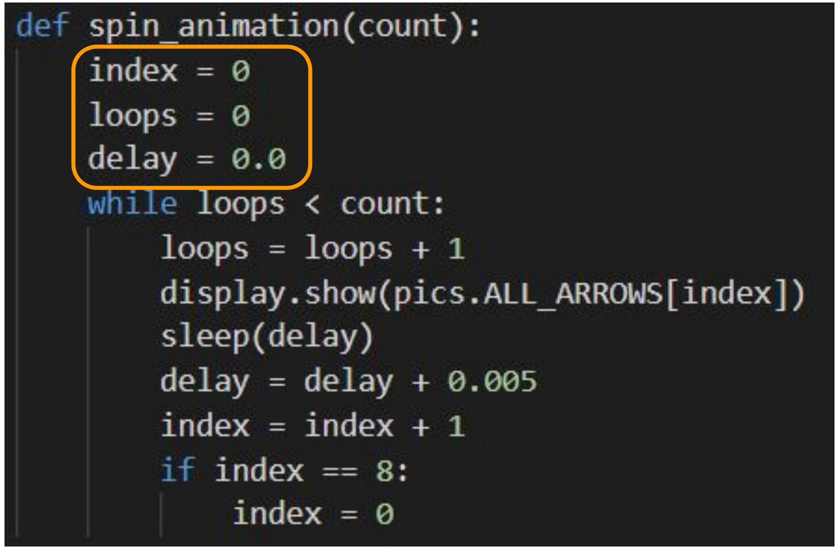
```
def display_pixels2(numOfPixels):
    for lite in range(numOfPixels):
        color = random.choice(COLOR_LIST)
        pixels.set(lite, color)
```

- When you call a function with a parameter, you must give the value for the parameter
- This is called an **argument**



```
display_pixels(2)
```

Functions with local variables



```
def spin_animation(count):
    index = 0
    loops = 0
    delay = 0.0
    while loops < count:
        loops = loops + 1
        display.show(pics.ALL_ARROWS[index])
        sleep(delay)
        delay = delay + 0.005
        index = index + 1
        if index == 8:
            index = 0
```

Sometimes you create and use local variables:

- They do not exist and cannot be used outside the function
- They are not used anywhere else in the code
- They are only used in the function

FIRIA LABS

Review: parameters and local variables

So how do you determine what variable is a parameter and what is a local variable?

Here are some standard rules for parameters:

- If a variable is used in a calculation (right side of =)
- If a variable is used in a condition (if statement)
- If a variable is used in a condition (loop)

Here are some standard rules for local variables:

- If the variable is being calculated (left side of =)
- If the variable is the counter in a loop

Global Variables

Functions, parameters and local variables

Global Variables

But every now and then …. Not very often …. You use a variable that needs to be accessed in several parts of the code, and needs to keep its value throughout.

- **Global Variable:** a variable declared outside a function (like in the main program) and can be accessed throughout the program

Global Variables

So far you have used a few global variables

- color
- delay
- choice
- index

You might have changed the value of the variable, but that was in the main program and not in a function.

Global Variables

You really want to keep it that way – use as few global variables as needed.

- But SOMETIMES you need a global variable whose value will change inside a function
- Examples:
    - A variable that **count**s something (like the number correct)
    - An **index** of a list, if scrolling left or right
    - The amount of a **delay** that changes during the program code

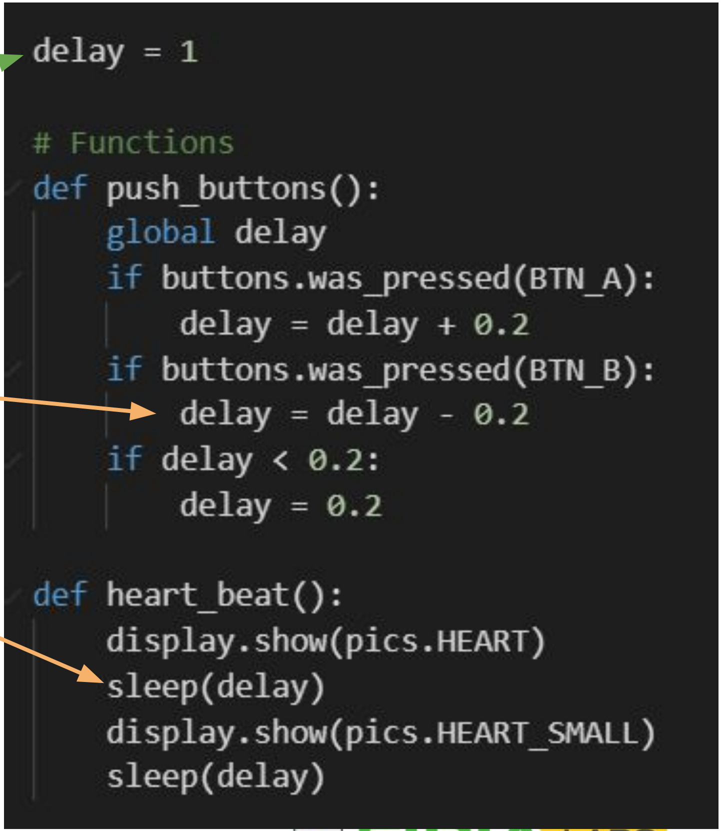
Functions and global variables

Examples

Example #1: Function with global variable

This is most of the code to the Heart Beat program

- A global variable **delay** is declared
- The **delay** is used in two functions

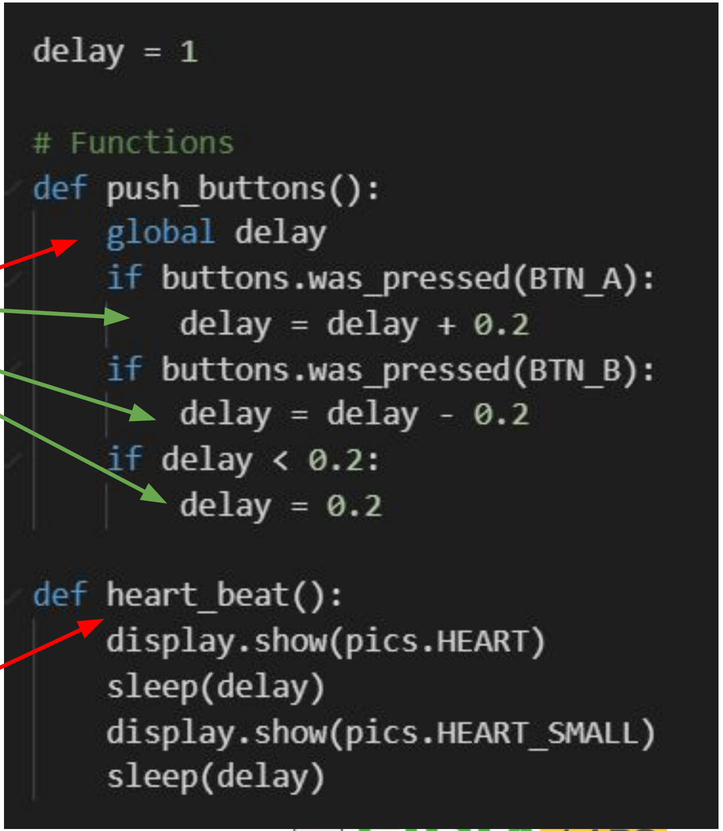


```
delay = 1

# Functions
def push_buttons():
    global delay
    if buttons.was_pressed(BTN_A):
        delay = delay + 0.2
    if buttons.was_pressed(BTN_B):
        delay = delay - 0.2
    if delay < 0.2:
        delay = 0.2

def heart_beat():
    display.show(pics.HEART)
    sleep(delay)
    display.show(pics.HEART_SMALL)
    sleep(delay)
```

Example #1: Function with global variable

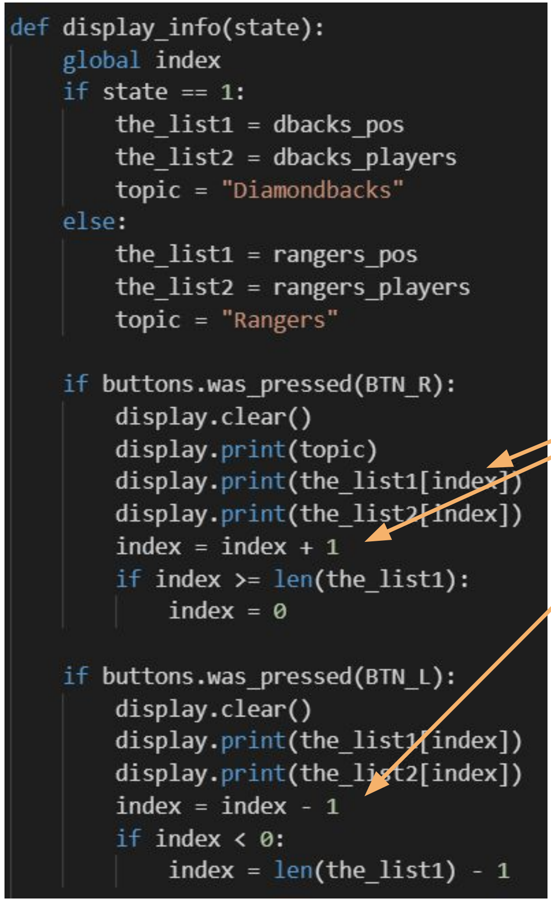
This is most of the code to the Heart Beat program

- The **delay** is used and changed in the **push_buttons()** function
- You want the value to remain, even after the function ends
- Use a **global** declaration to keep the changed value of a global variable in a function
- The delay is not changed in the second function (no global needed)



```
delay = 1

# Functions
def push_buttons():
    global delay
    if buttons.was_pressed(BTN_A):
        delay = delay + 0.2
    if buttons.was_pressed(BTN_B):
        delay = delay - 0.2
    if delay < 0.2:
        delay = 0.2

def heart_beat():
    display.show(pics.HEART)
    sleep(delay)
    display.show(pics.HEART_SMALL)
    sleep(delay)
```

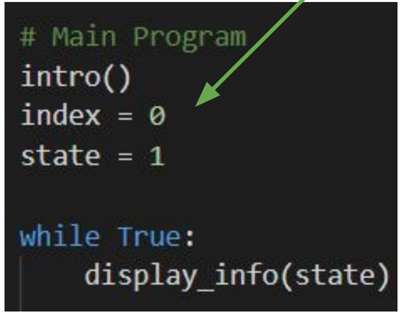
Example #2: Function with global variable



```
def display_info(state):
    global index
    if state == 1:
        the_list1 = dbacks_pos
        the_list2 = dbacks_players
        topic = "Diamondbacks"
    else:
        the_list1 = rangers_pos
        the_list2 = rangers_players
        topic = "Rangers"

    if buttons.was_pressed(BTN_R):
        display.clear()
        display.print(topic)
        display.print(the_list1[index])
        display.print(the_list2[index])
        index = index + 1
        if index >= len(the_list1):
            index = 0

    if buttons.was_pressed(BTN_L):
        display.clear()
        display.print(the_list1[index])
        display.print(the_list2[index])
        index = index - 1
        if index < 0:
            index = len(the_list1) - 1
```

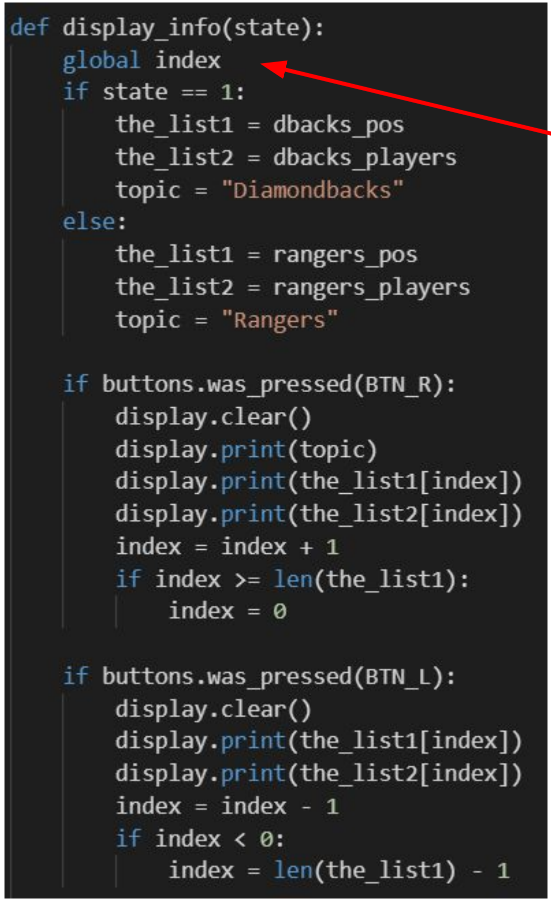
This is most of the code to the Create PT Practice #1 (with two lists)

- A global variable **index** is declared
- The **index** is used AND changed in the display_info() function



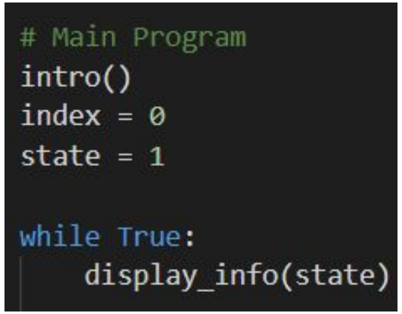
```
# Main Program
intro()
index = 0
state = 1

while True:
    display_info(state)
```

Example #2: Function with global variable



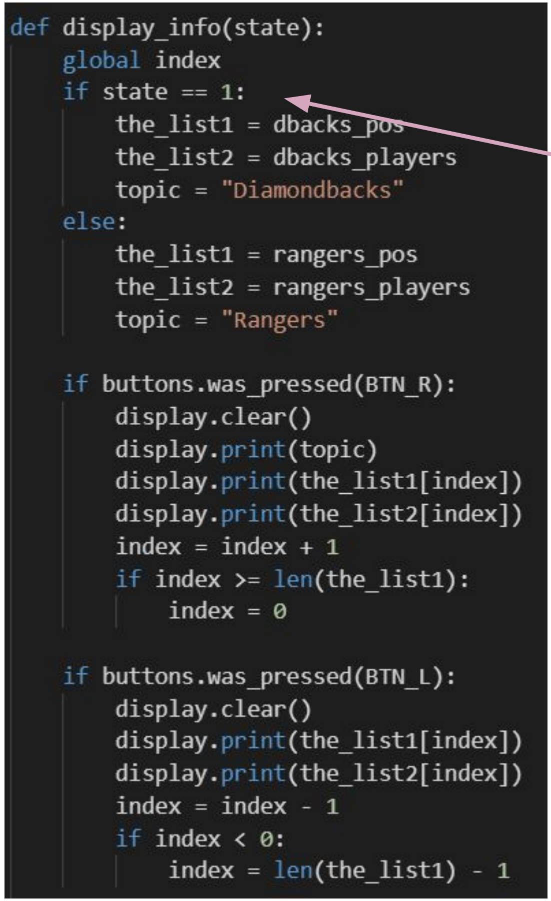
```python
def display_info(state):
    global index
    if state == 1:
        the_list1 = dbacks_pos
        the_list2 = dbacks_players
        topic = "Diamondbacks"
    else:
        the_list1 = rangers_pos
        the_list2 = rangers_players
        topic = "Rangers"

    if buttons.was_pressed(BTN_R):
        display.clear()
        display.print(topic)
        display.print(the_list1[index])
        display.print(the_list2[index])
        index = index + 1
        if index >= len(the_list1):
            index = 0

    if buttons.was_pressed(BTN_L):
        display.clear()
        display.print(the_list1[index])
        display.print(the_list2[index])
        index = index - 1
        if index < 0:
            index = len(the_list1) - 1
```

- You want to keep the most recent value of index, a global variable
- Use a **global** declaration to keep the changed value of a global variable in a function



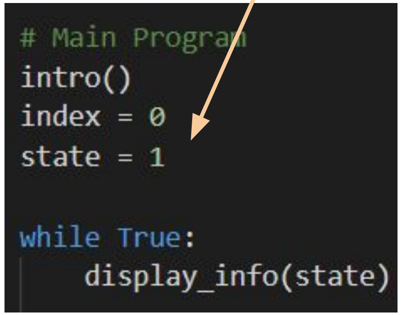
```python
# Main Program
intro()
index = 0
state = 1

while True:
    display_info(state)
```

Example #2: Function with global variable



```
def display_info(state):
    global index
    if state == 1:
        the_list1 = dbacks_pos
        the_list2 = dbacks_players
        topic = "Diamondbacks"
    else:
        the_list1 = rangers_pos
        the_list2 = rangers_players
        topic = "Rangers"

    if buttons.was_pressed(BTN_R):
        display.clear()
        display.print(topic)
        display.print(the_list1[index])
        display.print(the_list2[index])
        index = index + 1
        if index >= len(the_list1):
            index = 0

    if buttons.was_pressed(BTN_L):
        display.clear()
        display.print(the_list1[index])
        display.print(the_list2[index])
        index = index - 1
        if index < 0:
            index = len(the_list1) - 1
```

- Notice that **state** is also a global variable
- It is also used in **display_info()**
- However, it is NOT changed
- No need to use a **global** declaration for **state**
- ***NOTE:*** *if you did use a global declaration for state, it wouldn't affect the program at all*

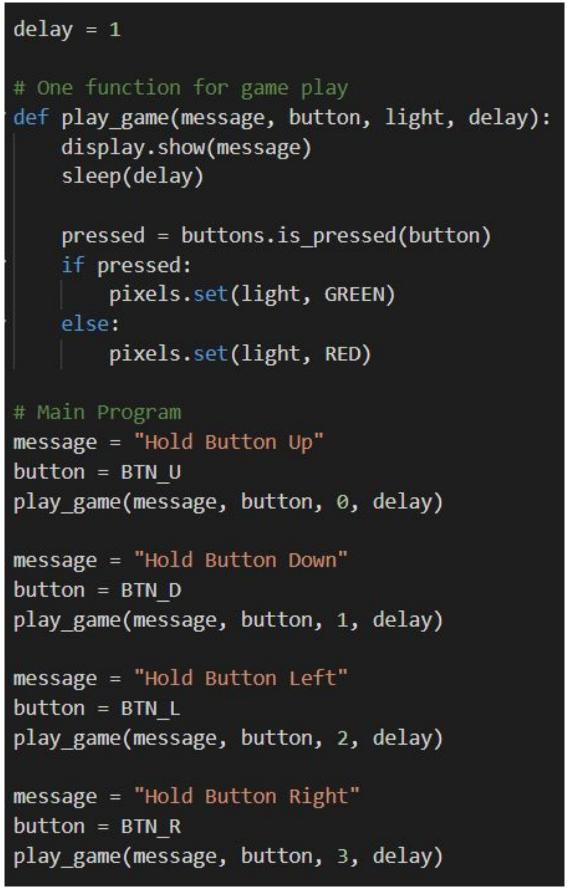


```
# Main Program
intro()
index = 0
state = 1

while True:
    display_info(state)
```

Functions and global variables

Activity

Add a global variable to a program

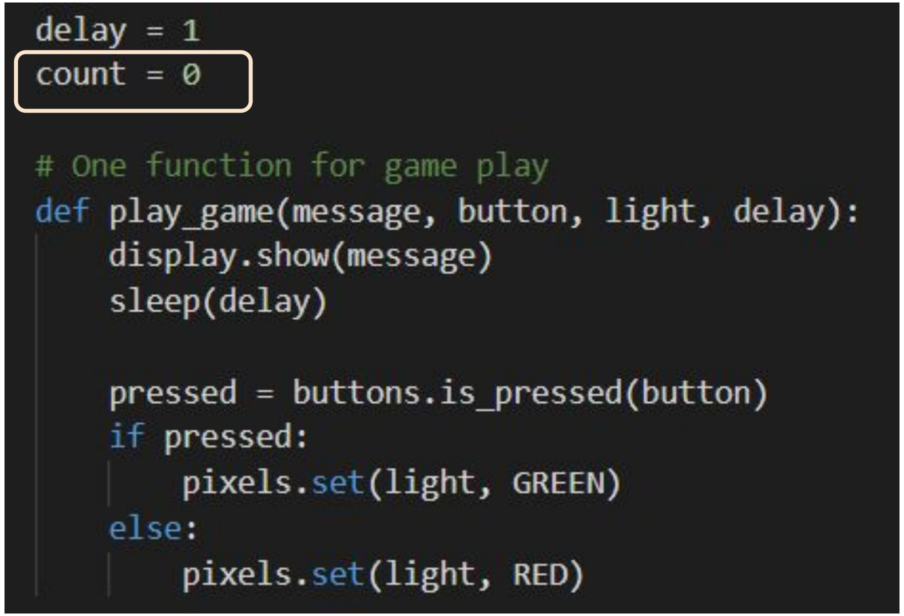


```
delay = 1

# 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)
```

Open your **Display2** program

- Completed at the end of “Functions, parameters and local variables Part 2”
- “Save As” and name the new program **Display3**
- You will add a global **count** variable to the code

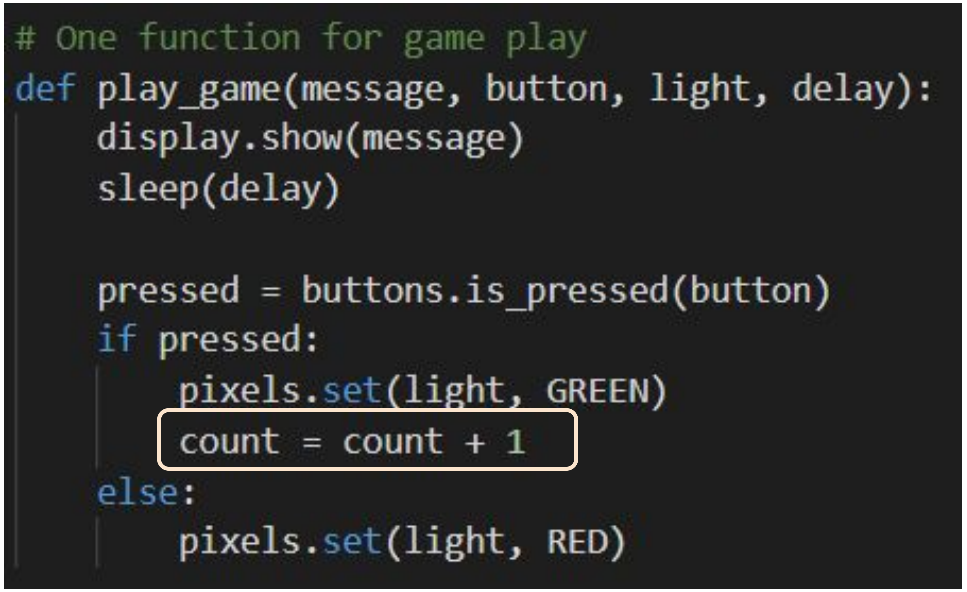
Add a global variable to a program



```
delay = 1
count = 0

# 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)
```

- Declare a global variable for count, and assign it the value of 0

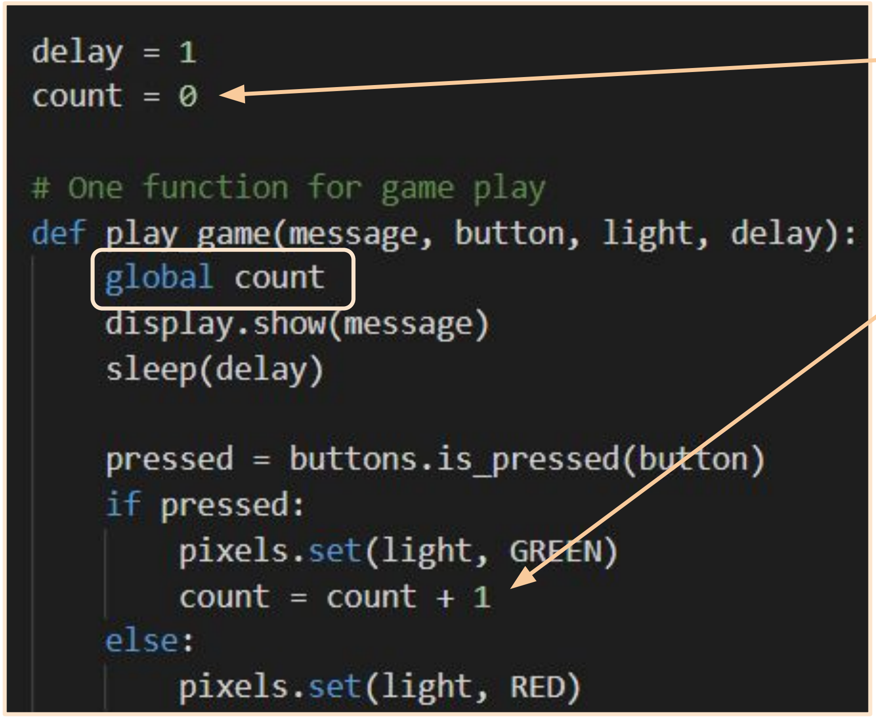
Add a global variable to a program



```
# 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)
        count = count + 1
    else:
        pixels.set(light, RED)
```

- You will count the number of times you press the correct button
- Increment **count** in the if statement

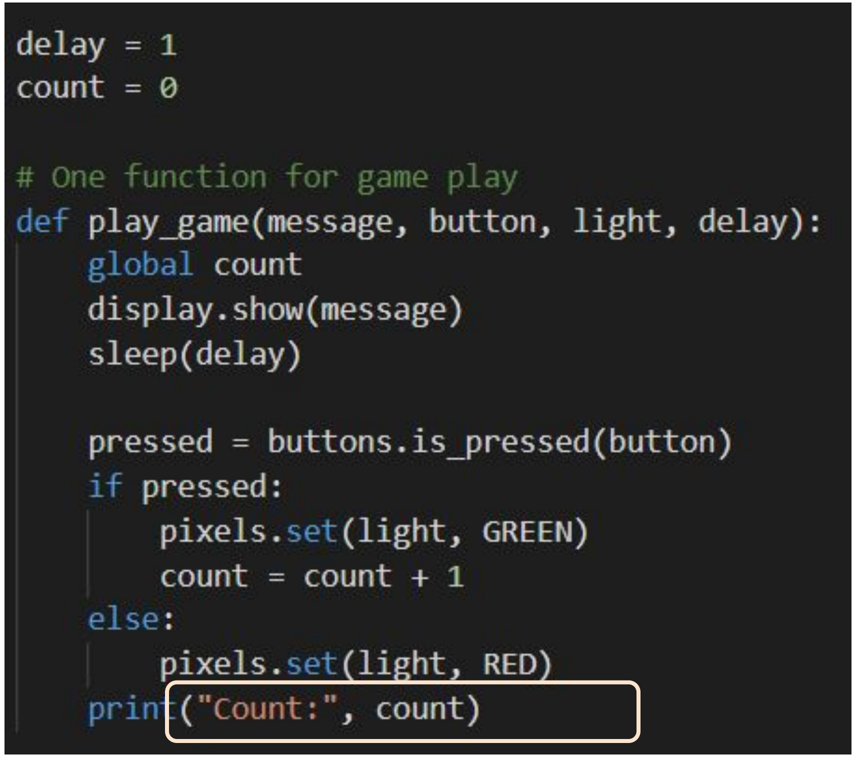
Add a global variable to a program



```
delay = 1
count = 0

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

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

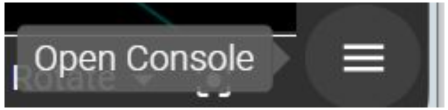
- **count** is a global variable
- You are changing the value of **count** in the function
- Use a **global** declaration for **count**

Add a global variable to a program



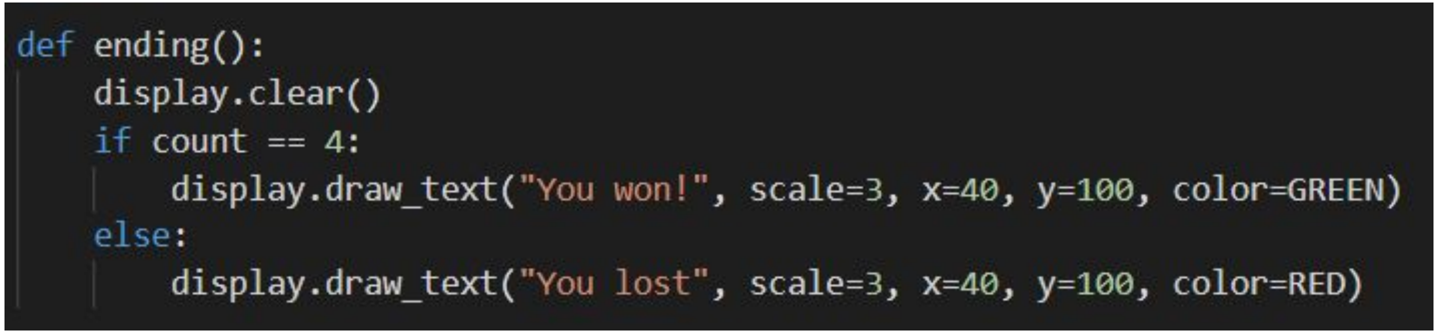
```
delay = 1
count = 0

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

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

- Just to watch how it works, add a print statement in the function
- Open the console and watch the value of count change during program execution



Add a global variable to a program

- Now use the count variable in an ending function
- You will NOT change the value of count, so you do not need a global declaration
- Use an if statement to display a message

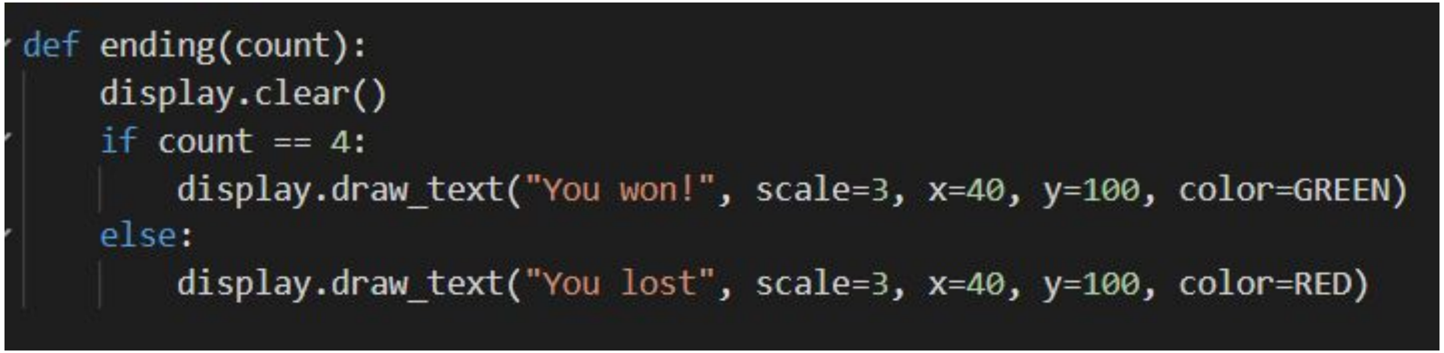


```python
def ending():
    display.clear()
    if count == 4:
        display.draw_text("You won!", scale=3, x=40, y=100, color=GREEN)
    else:
        display.draw_text("You lost", scale=3, x=40, y=100, color=RED)
```

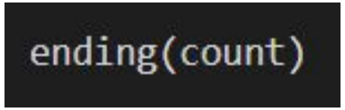
Your messages can be whatever you want. Also, you can use display.show() or display.print() if you want to.

What about a parameter?

- A parameter wasn't needed, since **count** is global
- But, can you add a parameter anyway?
- For the Create PT, you need a function with a parameter
- Go ahead and try it



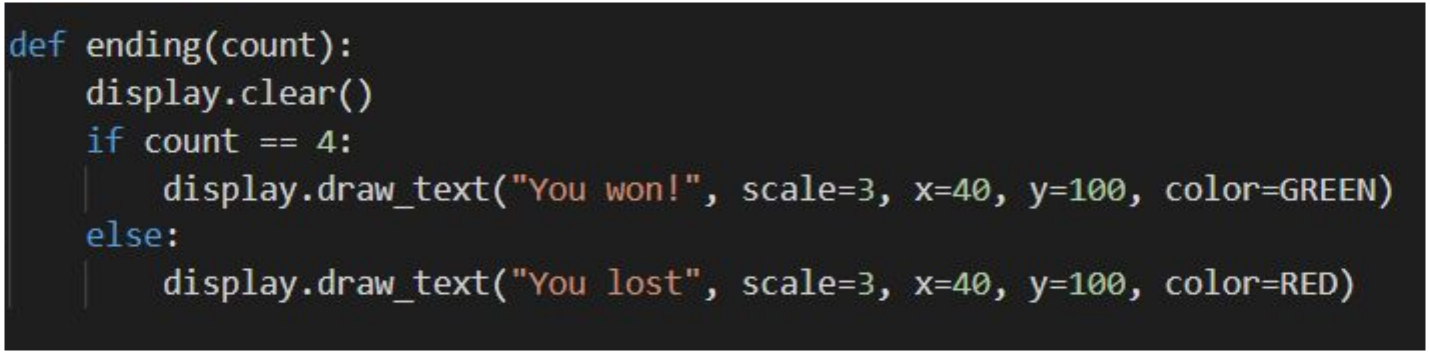
```
def ending(count):
    display.clear()
    if count == 4:
        display.draw_text("You won!", scale=3, x=40, y=100, color=GREEN)
    else:
        display.draw_text("You lost", scale=3, x=40, y=100, color=RED)
```



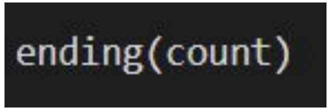
```
ending(count)
```

What about a parameter?

- For the Create PT, you need a function with a parameter that is used in an if statement
- Does this now meet the requirement?



```
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)
```



```
ending(count)
```

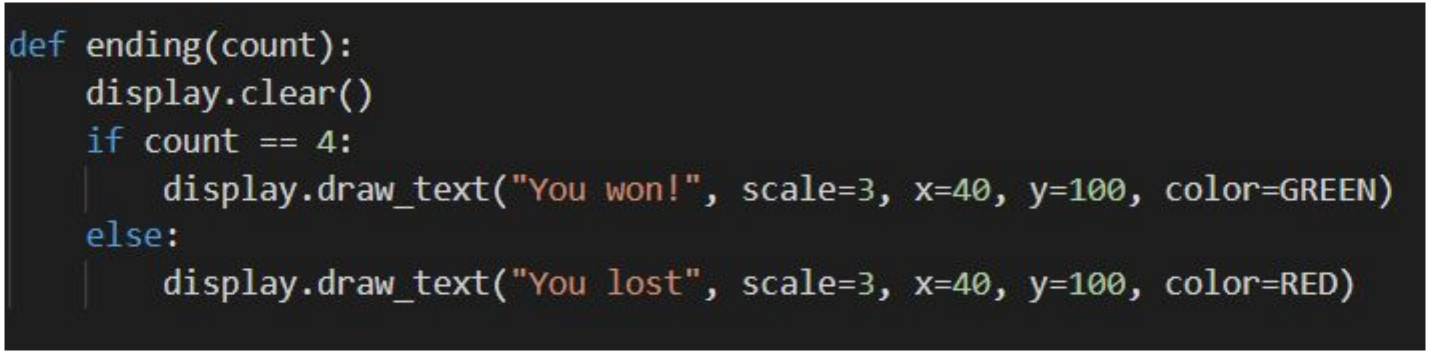
Functions and global variables

Optional Activity

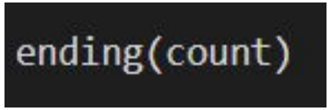
FIRIA LABS

What about a loop?

- Another requirement of the Create PT is that the function have a loop in it.
- Does this function meet that requirement?



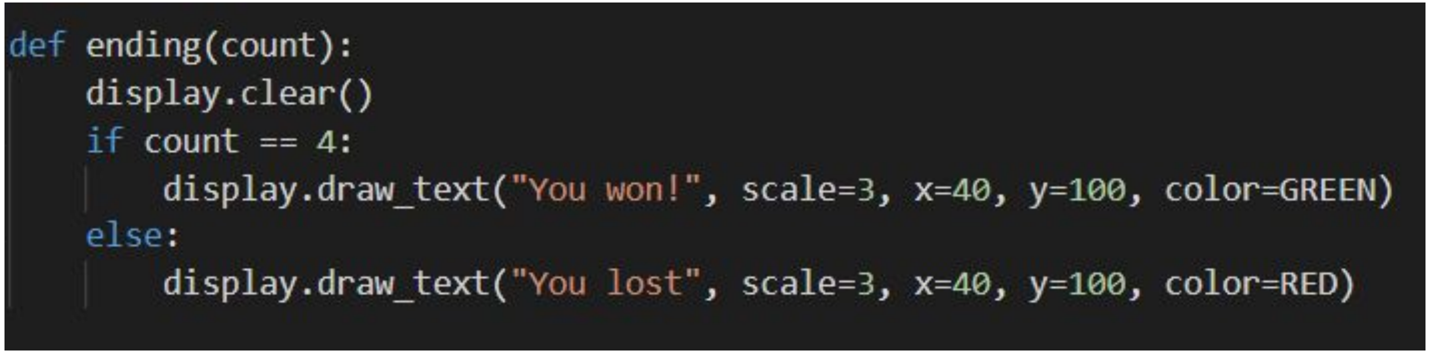
```
def ending(count):
    display.clear()
    if count == 4:
        display.draw_text("You won!", scale=3, x=40, y=100, color=GREEN)
    else:
        display.draw_text("You lost", scale=3, x=40, y=100, color=RED)
```



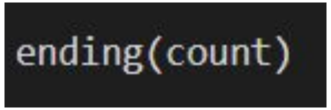
```
ending(count)
```

What about a loop?

- Add a loop at the beginning of the function that turns all pixels BLACK.
- You have seen this code before. Try it on your own.



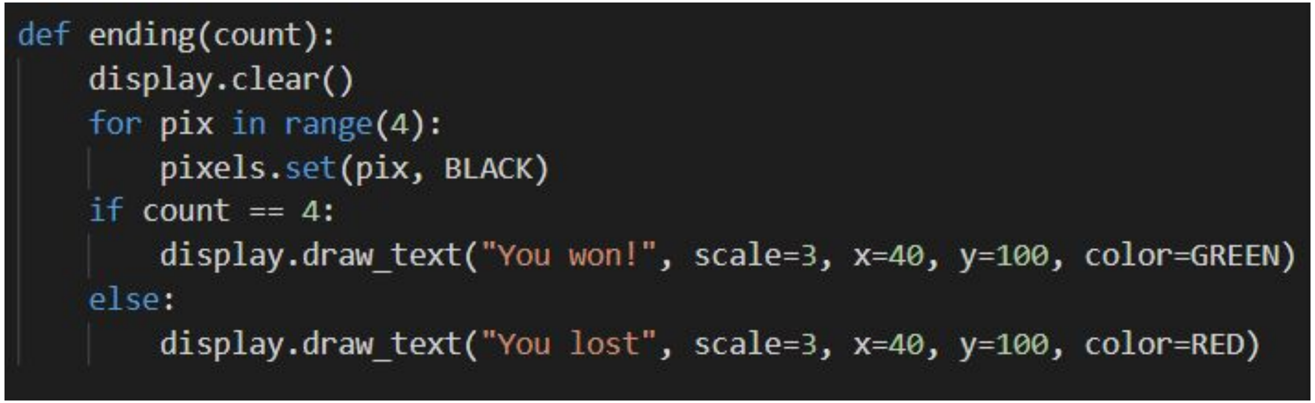
```python
def ending(count):
    display.clear()
    if count == 4:
        display.draw_text("You won!", scale=3, x=40, y=100, color=GREEN)
    else:
        display.draw_text("You lost", scale=3, x=40, y=100, color=RED)
```



```python
ending(count)
```

What about a loop?

- Does this function meet the requirements?
- Has a function with a parameter
    - Parameter is used in an if statement
- Function has:
    - If statement
    - Loop



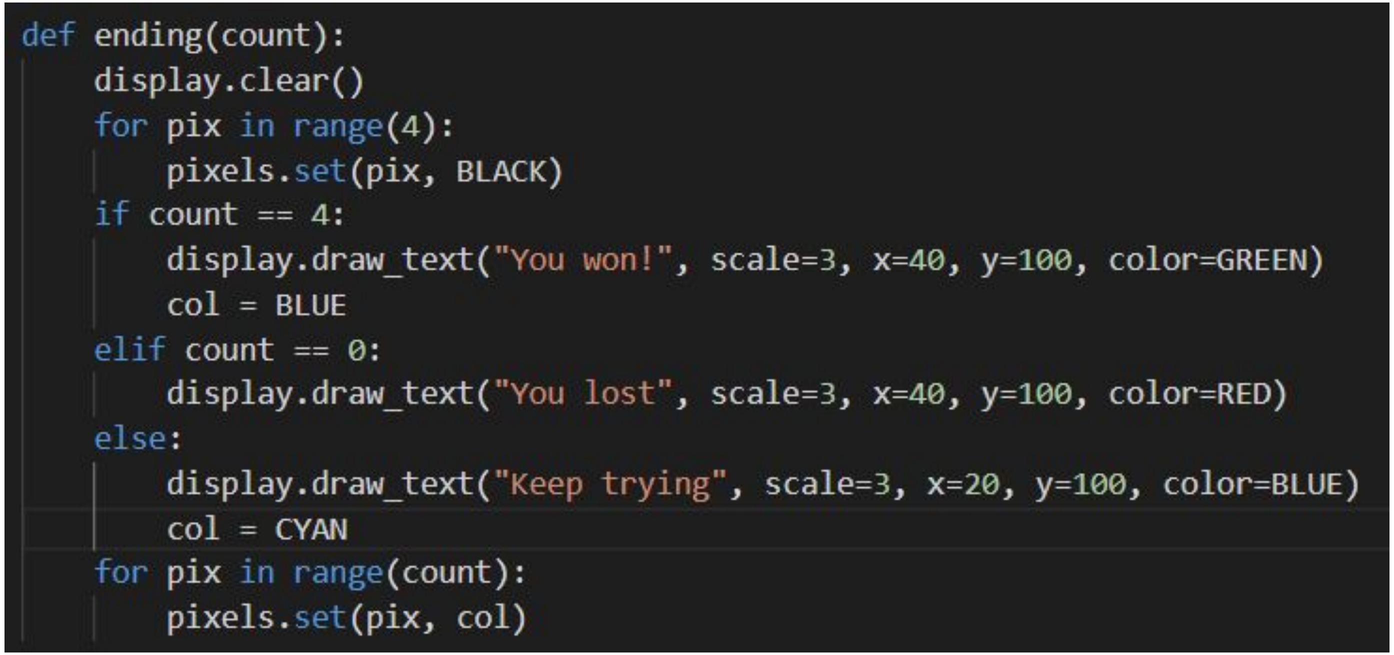
```
def ending(count):
    display.clear()
    for pix in range(4):
        pixels.set(pix, BLACK)
    if count == 4:
        display.draw_text("You won!", scale=3, x=40, y=100, color=GREEN)
    else:
        display.draw_text("You lost", scale=3, x=40, y=100, color=RED)
```

What about a loop?

- Can you do more?
- Add more selection (branches of the if statement)
  - One for 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

Function with parameter, selection & iteration

- How creative can you be?
- You can do something similar to this:



```python
def ending(count):
    display.clear()
    for pix in range(4):
        pixels.set(pix, BLACK)
    if count == 4:
        display.draw_text("You won!", scale=3, x=40, y=100, color=GREEN)
        col = BLUE
    elif count == 0:
        display.draw_text("You lost", scale=3, x=40, y=100, color=RED)
    else:
        display.draw_text("Keep trying", scale=3, x=20, y=100, color=BLUE)
        col = CYAN
    for pix in range(count):
        pixels.set(pix, col)
```

Function with parameter, selection & iteration

- 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

Go to the next Create PT Practice lesson, and make it work!

Wrap-up

Functions and Global Variables

When to use parameters and local variables?

- Answer the reflection questions on the assignment.