Functions, Parameters and Local Variables

A supplemental lesson after Mission 9

Part 1

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 - Part 1

Review: What is a function?

- **Function:** a named set of instructions that accomplishes a task

Reusable chunks of code

A function is a named chunk of code you can run anytime just by calling its name!

In other programming languages functions are sometimes called **procedures**. Functions can also be bundled with *objects*, where they're referred to as **methods**. Whatever you call them, they are a good way to package up useful sections of code you can use over and over again!

FIRIA LABS

When to use a function?

- 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

FIRIA LABS

A function in Python

In Python you can **define** a new function like this:

```
def flashLEDs():
    leds.user(0b11111111)
    sleep(0.5)
    leds.user(0b00000000)
    sleep(0.5)
```

Once that's defined, you can call the function whenever you like:

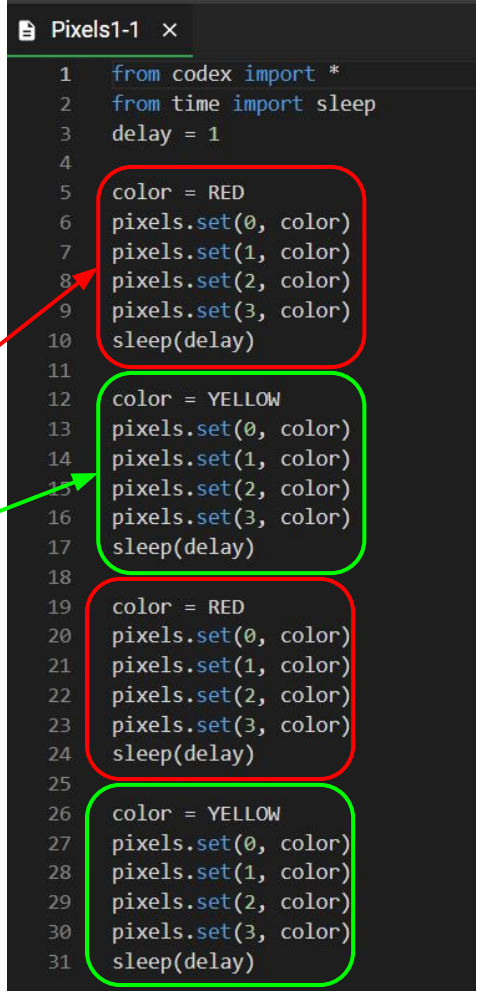
```
while True:
    flashLEDs()
```

FIRIA LABS

Example: Mission 3

Look through your code and find sections that could be a function.

- In this sample, some code is repeated
- This is a perfect place to make a function!



Example: Mission 3

- All of the code is in functions
- Functions have to be called in order for their instructions to run
- The great thing about functions is you can call them multiple times and in any order

Here is one example of calling functions

```
1  from codex import *
2  from time import sleep
3  delay = 1
4
5  def turn_red():
6      color = RED
7      pixels.set(0, color)
8      pixels.set(1, color)
9      pixels.set(2, color)
10     pixels.set(3, color)
11     sleep(delay)
12
13 def turn_yellow():
14     color = YELLOW
15     pixels.set(0, color)
16     pixels.set(1, color)
17     pixels.set(2, color)
18     pixels.set(3, color)
19     sleep(delay)
20
21 # Main program
22 turn_red()
23 turn_yellow()
24 turn_red()
25 turn_yellow()
```

Function calls

FIRIA LABS

Example: Mission 4

Look through your code and find sections that could be functions.

- You probably have four sections in your code.
- Each section is similar but asks for a different button push and a lights a different pixel

```
display.show("Hold Button A")
sleep(1)
pressed = buttons.is_pressed(BTN_A)
if pressed:
    pixels.set(0, GREEN)
else:
    pixels.set(0, RED)
sleep(1)
```

```
display.show("Hold Button B")
sleep(1)
pressed = buttons.is_pressed(BTN_B)
if pressed:
    pixels.set(1, GREEN)
else:
    pixels.set(1, RED)
sleep(1)
```

```
display.show("Hold Button L")
sleep(1)
pressed = buttons.is_pressed(BTN_L)
if pressed:
    pixels.set(2, GREEN)
else:
    pixels.set(2, RED)
sleep(1)
```

```
display.show("Hold Button R")
sleep(1)
pressed = buttons.is_pressed(BTN_R)
if pressed:
    pixels.set(3, GREEN)
else:
    pixels.set(3, RED)
sleep(1)
```

Example: Mission 4

- Now you have functions for each task (or button press)
- Four functions for four tasks
- Is your indenting correct?
- Will your code work properly now? Why or why not?

```
def option_A():
    display.show("Hold Button A")
    sleep(1)
    pressed = buttons.is_pressed(BTN_A)
    if pressed:
        pixels.set(0, GREEN)
    else:
        pixels.set(0, RED)
    sleep(1)
```

```
def option_B():
    display.show("Hold Button B")
    sleep(1)
    pressed = buttons.is_pressed(BTN_B)
    if pressed:
        pixels.set(1, GREEN)
    else:
        pixels.set(1, RED)
    sleep(1)
```

```
def option_L():
    display.show("Hold Button L")
    sleep(1)
    pressed = buttons.is_pressed(BTN_L)
    if pressed:
        pixels.set(2, GREEN)
    else:
        pixels.set(2, RED)
    sleep(1)
```

```
def option_R():
    display.show("Hold Button R")
    sleep(1)
    pressed = buttons.is_pressed(BTN_R)
    if pressed:
        pixels.set(3, GREEN)
    else:
        pixels.set(3, RED)
    sleep(1)
```

Function calls

```
44  # Main Program
45  option_A()
46  option_B()
47  option_L()
48  option_R()
```

FIRIA LABS

Example: Mission 6

Function definition

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

Function call

```
heart_beat()
```

Example: Mission 8

Function definition

```
def display_pixels():
    color = random.choice(COLOR_LIST)
    pixels.set(0, color)
    color = random.choice(COLOR_LIST)
    pixels.set(1, color)
    color = random.choice(COLOR_LIST)
    pixels.set(2, color)
    color = random.choice(COLOR_LIST)
    pixels.set(3, color)
```

Function call

```
display_pixels()
```

FIRIA LABS

Example: Mission 9

```
def show_random_arrow():
    arrow = random.randrange(8)
    display.show(pics.ALL_ARROWS[arrow])
```

Function calls

```
show_random_arrow()
wait_button()
draw_centerlines()
```

Example: Mission 10

```
def wait_button():
    display.print("Press A to start")
    while True:
        if buttons.was_pressed(BTN_A):
            break

def draw_centerlines():
    display.fill(BLACK)
    display.draw_line(CENTER, 0, CENTER, 105,WHITE)
    display.draw_line(CENTER, 135, CENTER, 239, WHITE)
    display.draw_line(0, CENTER, 105, CENTER, WHITE)
    display.draw_line(135, CENTER, 239, CENTER, WHITE)
```

FIRIA LABS

Example: Intro & Ending

```
def intro():
    display.print("Welcome to the ")
    display.print("World Series")
    display.print("A = Diamondbacks")
    display.print("B = Rangers")
    display.print("")
    display.print("R = Scroll forward")
    display.print("L = Slideshow")
    display.print("U = Random player")
    display.print("D = Quit")

def ending():
    display.clear()
    display.print("Thank you!")
    display.print("Have a good day!")
```

Example: with for loop

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

Function calls

```
intro()
display_pixels2()
ending()
```

FIRIA LABS

Functions, parameters and local variables

Functions and parameters

- All of the examples are functions
- None of the functions in the examples required a **parameter**
- But you will often create a function that needs information in order to complete its task.
- **Parameter:** information the function needs to complete the task.

FIRIA LABS

Functions and parameters

- Look at this example:

```
def display_pixels2():
    for lite in range(4):
        color = random.choice(COLOR_LIST)
        pixels.set(lite, color)
```

- This code will always turn on four pixel LEDs (0,1, 2, 3)
- What if you didn’t always want to turn on all four pixel LEDs?
- Sometimes you want to turn on one, or sometimes two, or sometimes three, or sometimes all four
- You could write four different functions, but that kind of defeats the purpose of using a function

Functions and parameters

- Instead, you can give the function the information it needs to complete the task
- In this case, it would be 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)
```

FIRIA LABS

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)
```

FIRIA LABS

Functions and parameters

- An **argument** can be a literal value

```
display_pixels(2)

num_pixels = 2
display_pixels2(num_pixels)
```

In both these cases, the value passed to the parameter is 2, so pixels 0 and 1 will turn on.

- An **argument** can be a variable
- The name of the **argument** **variable** does not have to be the same as the **parameter**
- The **value** of the **variable** is passed to the **parameter**

FIRIA LABS

Functions with parameters

```
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
```

The argument **count** is used to stop the loop.

```
def get_degrees(tilt):
    scaled = (tilt/16384) * 90
    degrees = int(scaled)
    if degrees < -90:
        degrees = -90
    if degrees > 90:
        degrees = 90
    return degrees
```

The argument **tilt** is used to calculate the scale for the degrees.

Functions with parameters

```
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
```

Possible function calls with argument:

```
spin_animation(3)
spin_animation(num_spins)
```

```
def get_degrees(tilt):
    scaled = (tilt/16384) * 90
    degrees = int(scaled)
    if degrees < -90:
        degrees = -90
    if degrees > 90:
        degrees = 90
    return degrees
```

Possible function calls with argument:

```
val = accel.read()
degrees_x = get_degrees(val[0])
degrees_y = get_degrees(val[1])
```

FIRIA LABS

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
```

You may notice that there are some other variables used in the function.

- These are local variables
- They do not exist and cannot be used outside the function
- But that is okay because:
- They are not used anywhere else in the code
- They are only used in the function

FIRIA LABS

Functions with local variables

- This function also has local variables.
- They are only used in this function and nowhere else

```
def get_degrees(tilt):
    scaled = (tilt/16384) * 90
    degrees = int(scaled)
    if degrees < -90:
        degrees = -90
    if degrees > 90:
        degrees = 90
    return degrees
```

FIRIA LABS

Functions with 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

FIRIA LABS

Functions with parameters

```
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
```

Assignment statements: local variables

Local variables used in other places in the function

There is only one variable that isn't local here: **count**

- count is used in a loop condition, so it needs to be a parameter

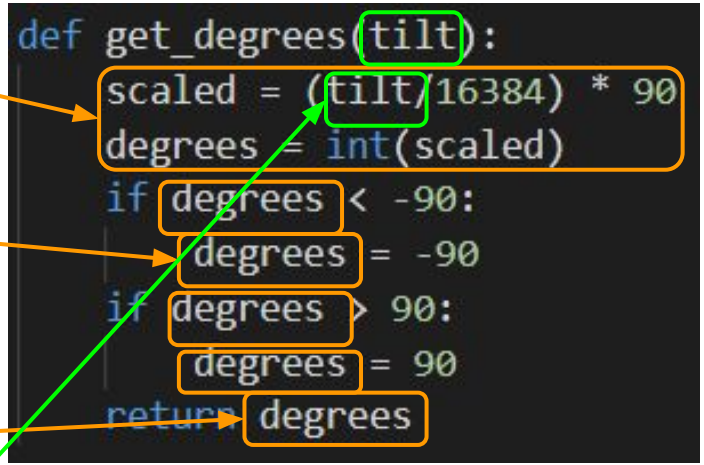
FIRIA LABS

Functions with parameters

Assignment statements: local variables

Local variables used in other places in the function

One of the local variables is being changed, and it will be needed outside the function. It's value will be returned



There is only one variable that isn't local here: **tilt**, which is used in a calculation (right side of =).

- So it needs to be a parameter

FIRIA LABS

Functions, parameters and local variables

Activity

Example: Functions with parameters

- Look at a function you created and used recently
- Can you identify the parameter?
- Why does it need to be a parameter?
- Can you identify the local variables?
  - Hint: there are 4
- Why are they local variables?
- What will a function call look like?

```
def slideshow(topic):
    if topic == 1:
        the_list1 = dbacks_pos
        the_list2 = dbacks_players
        team = "Diamondbacks"
    else:
        the_list1 = rangers_pos
        the_list2 = rangers_players
        team = "Rangers"
    for index in range(len(the_list1)):
        display.clear()
        display.print(team)
        display.print(the_list1[index])
        display.print(the_list2[index])
        sleep(2)
    display.clear()
    display.print("End of list")
```

FIRIA LABS

A. Functions with parameters

- Look at the code:
- You decide to make the code into a function.
- What would you call the function?
- What are the variables it needs?
- What are the parameters?
- What are the local variables?
- Does it need a return?
- What will a function call look like?

```
# Ending message
if count == 4:
    display.clear()
    display.draw_text("You WON", scale=4, color=BLUE, x=30, y=80)

else:
    display.clear()
    display.draw_text("You LOST", scale=4, color=RED, x=30, y=80)
```

FIRIA LABS

B. Functions with parameters

- Look at the code:
- You decide to make the circled code into a function.
- What would you call the function?
- What are the variables it needs?
- What are the parameters?
- What are the local variables?
- Does it need a return?
- What will a function call look like?

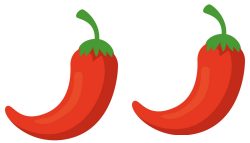
```
while True:
    # Start game with button B
    if buttons.was_pressed(BTN_B):
        # Reset the board for each game
        reset()
        # Select first random number
        num1 = random.randrange(6) + 1
        if num == 1:
            one_roll()
        elif num == 2:
            two_roll()
        elif num == 3:
            three_roll()
        elif num == 4:
            four_roll()
        elif num == 5:
            five_roll()
        else:
            six_roll()
        sleep(delay)
```

C. Functions with parameters

- Look at the code:
- You decide to make the code at the bottom into a function.
- What would you call the function?
- What are the variables it needs?
- What are the parameters?
- What are the local variables?
- Does it need a return?
- What will a function call look like?

```
while True:
    if buttons.was_pressed(BTN_U):
        audio.mp3("sounds/codetrek")
    if buttons.was_pressed(BTN_D):
        display.fill(GREEN)
    if buttons.was_pressed(BTN_L):
        display.show(pics.TIARA)
    if buttons.was_pressed(BTN_R):
        display.fill(BLACK)
        pixels.set(0, RED)
        pixels.set(1, GREEN)
        pixels.set(2, BLUE)
        pixels.set(3, YELLOW)
        sleep(delay)
        pixels.set(0, BLACK)
        pixels.set(1, BLACK)
        pixels.set(2, BLACK)
        pixels.set(3, BLACK)
    if buttons.was_pressed(BTN_A):
        audio.mp3("sounds/welcome")
    if buttons.was_pressed(BTN_B):
        display.show(pics.HAPPY)

    sleep(delay)
    display.fill(BLACK)
    display.show("Press a Button!")
    sleep(delay)
```

D. Functions with parameters



- Look at the code:
- You decide to make the code at the bottom into a function.
- What would you call the function?
- What are the variables it needs?
- What are the parameters?
- What are the local variables?
- Does it need a return?
- What will a function call look like?

```
while True:
    red = random.randrange(0, 255)
    green = random.randrange(0, 255)
    blue = random.randrange(0, 255)
    color = (red, green, blue)

    pixels.set(0, color)

    red = random.randrange(0, 255)
    green = random.randrange(0, 255)
    blue = random.randrange(0, 255)
    color = (red, green, blue)

    pixels.set(1, color)

    how_many = 4
    # turn off pixel LEDs
    for lite in range(how_many):
        pixels.set(lite, BLACK)
```

FIRIA LABS

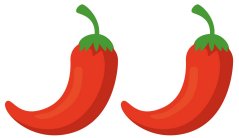
E. Functions with parameters

- Look at the code:
- You decide to make the circled code into a function so it can be used for all the pixel LEDs.
- What would you call the function?
- What are the variables it needs?
- What are the parameters?
- What are the local variables?
- Does it need a return?
- What will a function call look like?

```
while True:
    red = random.randrange(0, 255)
    green = random.randrange(0, 255)
    blue = random.randrange(0, 255)
    color = (red, green, blue)

    pixels.set(0, color)

    red = random.randrange(0, 255)
    green = random.randrange(0, 255)
    blue = random.randrange(0, 255)
    color = (red, green, blue)

    pixels.set(1, color)

    how_many = 4
    # turn off pixel LEDs
    for lite in range(how_many):
        pixels.set(lite, BLACK)
```

FIRIA LABS

F. Functions with parameters



- Look at the code:
- You decide to make the code at the bottom into a function.
- What would you call the function?
- What are the variables it needs?
- What are the parameters?
- What are the local variables?
- Does it need a return?
- What will a function call look like?

```
while True:
    if buttons.was_pressed(BTN_A):
        choice = 0
    if buttons.was_pressed(BTN_B):
        choice = 1
    if buttons.was_pressed(BTN_U):
        choice = 2
    if buttons.was_pressed(BTN_D):
        choice = 3
    if buttons.was_pressed(BTN_L):
        choice = 4
    if buttons.was_pressed(BTN_R):
        choice = 5

    my_image = my_list[choice]

    if type(my_image) == tuple:
        display.fill(my_image)
    else:
        display.show(my_image)
```

Wrap-up

Functions, parameters and local variables - Part 1

When to use parameters and local variables?

- Answer the reflection questions on the assignment.