Defining and Using Functions

An extension for Mission 3, 4 and 6

What is a function?

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

What is a function?

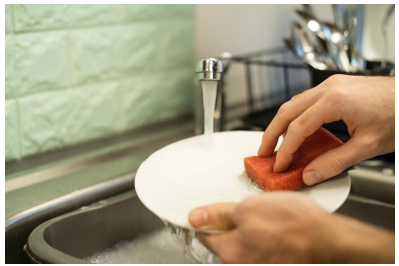
- **Function:** a named set of instructions that accomplishes a task

A function is a form of abstraction.

- **Abstraction:** the process of taking away or removing characteristics from something in order to reduce it to a set of essential characteristics
- What this means is that you can take a task and give it a name. Then you don't have to worry about the details of the task, but on the task itself and when to use it.

FIRIA LABS

Examples of abstraction



28	36	7	18	73	21
81	14	65	37	60	52
13	4	17	1	26	44
76	99	80	10	16	40
58	33	91	49	62	23
71	5	25	11	88	42
7	23	39	67	76	84
15	30	42	96	87	100

Wash the dishes

- You don’t have to be told every step for this task, which is the underlying details. The task has a name and you know what to do.

Turn left when driving

- You don’t need to know how the car is turned when you use the steering wheel. The details are hidden but you can still drive the car

Select the highest number

- There are many ways to find the highest number in a list. You don’t need to see the details of how it is done, just that when you complete the task the highest number will be found.

FIRIA LABS

A function is abstraction

```
def find_max(high, num):
    if num > high:
        high = num
    return high
```

- A section or block of code accomplishes a specific task
- Give the block of code a name
  - This “hides” the details
  - Focus on when to use the block of code
- Instead of writing all the code, a programmer calls the function by name

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

Try defining and calling functions in your Mission 3 code!

- Open your Pixels1 code.
- It may look like this. It will probably be similar but may be different.
- Save the code with a new name, so you keep the original code (use Save-As)

Pixels1-1

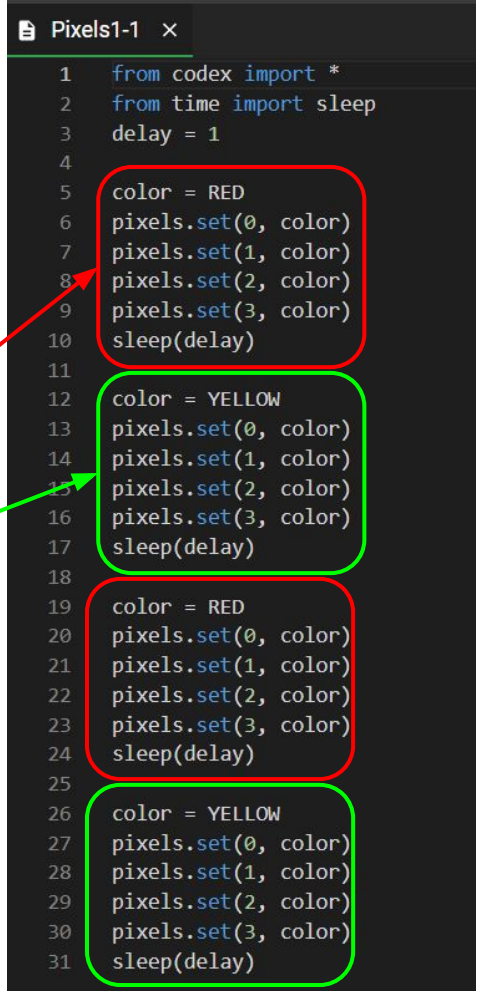
```
from codex import *
from time import sleep
delay = 1

color = RED
pixels.set(0, color)
pixels.set(1, color)
pixels.set(2, color)
pixels.set(3, color)
sleep(delay)

color = YELLOW
pixels.set(0, color)
pixels.set(1, color)
pixels.set(2, color)
pixels.set(3, color)
sleep(delay)

color = RED
pixels.set(0, color)
pixels.set(1, color)
pixels.set(2, color)
pixels.set(3, color)
sleep(delay)

color = YELLOW
pixels.set(0, color)
pixels.set(1, color)
pixels.set(2, color)
pixels.set(3, color)
sleep(delay)
```

FIRIA LABS

Identify sections of code

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!



Define a function

Define function for the color RED

- Functions typically are coded near the top of the program, just under imports and variables
- A function definition ends with a colon (:) – you are creating a block of code
- Don’t forget to indent! – the shortcut for this is to highlight the text and press TAB

```
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 color = YELLOW
14 pixels.set(0, color)
15 pixels.set(1, color)
16 pixels.set(2, color)
17 pixels.set(3, color)
18 sleep(delay)
19
20 color = RED
21 pixels.set(0, color)
22 pixels.set(1, color)
23 pixels.set(2, color)
24 pixels.set(3, color)
25 sleep(delay)
```

FIRIA LABS

Define a function

Define another function for the YELLOW (or whatever colors you have)

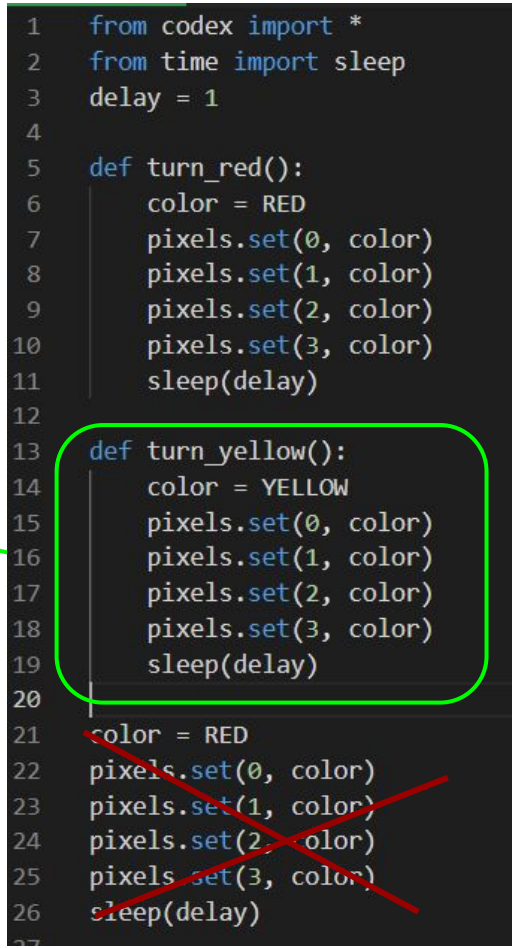
- A function definition ends with a colon (:) – you are creating a block of code
- Don’t forget to indent! – the shortcut for this is to highlight the text and press TAB

```
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  color = YELLOW
14  pixels.set(0, color)
15  pixels.set(1, color)
16  pixels.set(2, color)
17  pixels.set(3, color)
18  sleep(delay)
19
20  color = RED
21  pixels.set(0, color)
22  pixels.set(1, color)
23  pixels.set(2, color)
24  pixels.set(3, color)
25  sleep(delay)
```

FIRIA LABS

Define a function

- Define another function for the YELLOW (or whatever colors you have)
- If you have more colors, define a function for each color
- If your code is repeated, delete the repeated code



```
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 color = RED
22 pixels.set(0, color)
23 pixels.set(1, color)
24 pixels.set(2, color)
25 pixels.set(3, color)
26 sleep(delay)
```

Call a function

- Now you have functions for each task (color)
- If you have code for more than two colors, you will have more than two functions
- If you run your code, nothing will happen
- WHY??

```
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
```

FIRIA LABS

Call a function

- 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()
```

FIRIA LABS

Call a function

```
turn_red()
turn_red()
turn_yellow()
turn_red()
turn_red()
turn_yellow()
```

```
turn_yellow()
turn_yellow()
turn_yellow()
turn_red()
turn_red()
turn_red()
```

Here are more examples with just two functions. If you have more than two functions, there are many possibilities!

Extension

- Put your code in a loop
- Add a “kill switch” to break out of the loop

```
while True:
    turn_yellow()
    turn_yellow()
    turn_red()

    # TO DO -- kill switch
```

FIRIA LABS

Defining Functions

Mission 4 – Display Game

Try functions in your Mission 4 code!

Open your Display code.

It may look like similar to this, but it could be different.

Display-1

```
from codex import *
from time import sleep


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)
```

FIRIA LABS

Identify sections of code

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)
```

RIA LABS

Define a function

Create a function for the first section of code

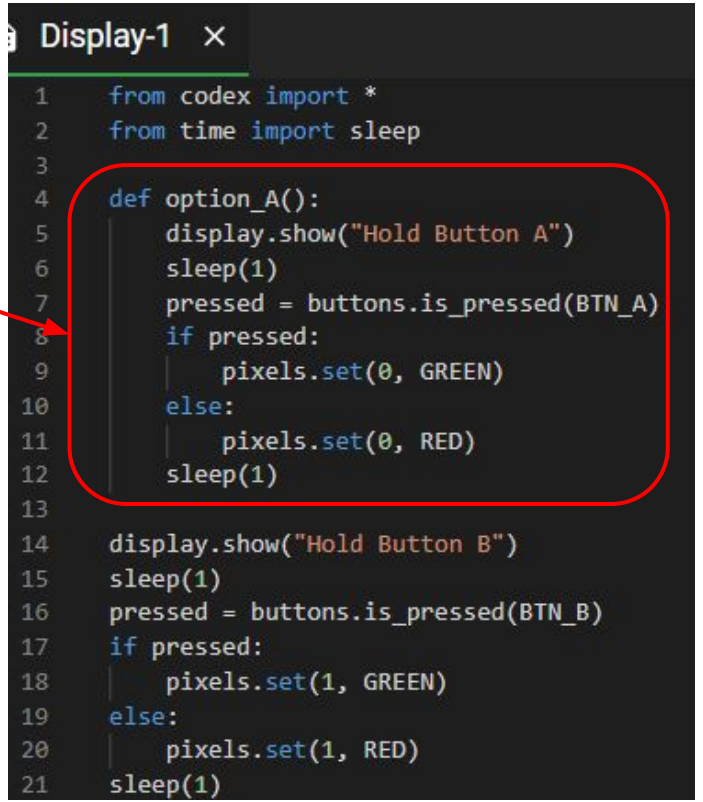
- Functions typically are coded near the top of the program, under imports and variables
- Use a descriptive name for the function
- A function definition ends with a colon (:) – you are creating a block of code
- Don’t forget to indent! – the shortcut for this is to highlight the text and press TAB

```
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)
```

Define a function

Your function may look like this.

- Create functions for the other three sections of code



Call a function

- 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)
```

FIRIA LABS

Call a function

- All of the code is in functions
- Functions have to be called 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

```
34  def option_R():
35      display.show("Hold Button R")
36      sleep(1)
37      pressed = buttons.is_pressed(BTN_R)
38      if pressed:
39          pixels.set(3, GREEN)
40      else:
41          pixels.set(3, RED)
42      sleep(1)
43
44  # Main Program
45  option_A()
46  option_B()
47  option_L()
48  option_R()
```

FIRIA LABS

Call a function

Here are more examples. There are many possibilities!

- Function calls go BELOW function definitions
- A function call does NOT end with a colon (:)
- The functions will be run sequentially, in the order you call them

```
# Main Program
option_R()
option_L()
option_B()
option_A()
```

```
# Main Program
option_B()
option_A()
option_L()
option_R()
```

FIRIA LABS

Challenges

Three challenges are suggested, with different levels of difficulty.
Try one at your choice level, or two challenges, or all three.

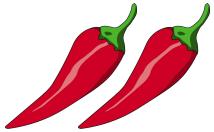
Mild Challenge

Improve Mission 4 with these extensions:

- Create a function that turns off all pixels (set each to black)
- Put your code in a loop to play forever
- Add a “kill switch” to end the loop

```
# Main Program
while True:
    option_A()
    option_B()
    option_L()
    option_R()
    clear_pixels()

    # TO DO kill switch
```

FIRIA LABS



Medium Challenge

Improve Mission 6 with this extension:

- Open the heartbeat function (Heart2)
- Create at least one function from the code in the loop
  - Possibilities:
    - Heart beat code
    - The speed up and slow down button pushes
- Add a kill switch to stop the program

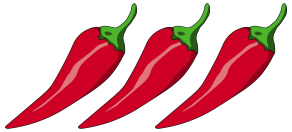
```
# Main Program
while True:
    heart_beat()
    push_buttons()

    # Kill Switch
```

Programming tip:

*The button A and button B conditions change the value of the variable **delay**. If you put the code in a function, you have to include **global delay** right after the function definition.*

```
def push_buttons():
    global delay
    if ...
```

FIRIA LABS



Spicy Challenge

Improve your Remix Program by:

- Creating at least one function from the code
- Adding a while loop (if not already there)
- Add a kill switch to stop the program (if not already there)
- Anything else you can think of – be creative!

FIRIA LABS

Wrap-up

Abstraction and Functions

What is a function?

- Now you have had some experience with abstraction and functions.
- Open your assignment document and answer the questions in the reflection.

FIRIA LABS