Defining and Using Functions

An extension for Mission 3

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?

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 it out in your Mission 3 code!

Open your Pixels1 code
(if not already open).

It may look like this. It will
probably be similar but
may be different.

Pixels1-1

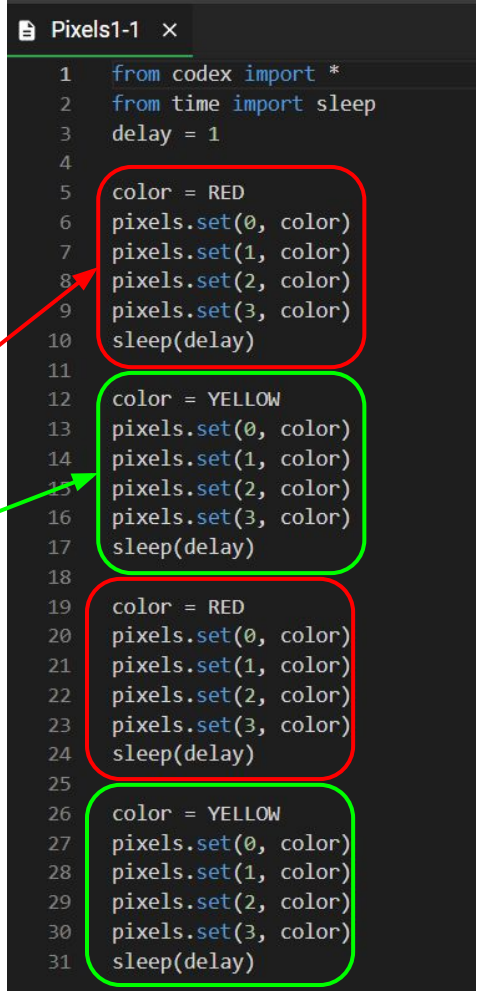
```
from codex import *
from time import sleep
delay = 1

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, 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)

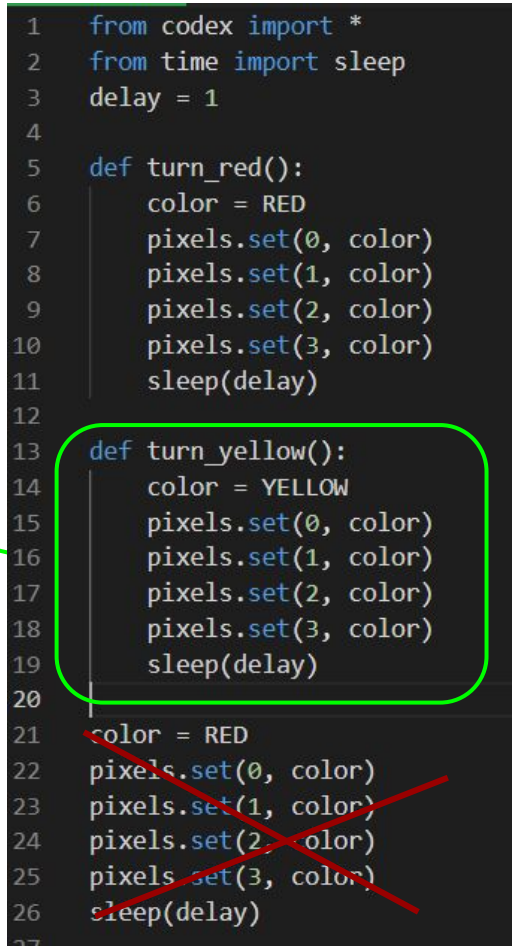
- A function definition ends with a colon (:) – you are creating a block of code
- Don’t forget to indent! – the shortcut for this is to highlight the text and press TAB

```
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 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 turn_red()
22 turn_yellow()
23 turn_red()
24 turn_yellow()
25
```

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