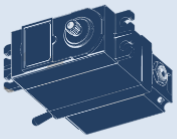


SMART SERVO GUIDES: RE-CODING

This guide will show you how to alter the code on your Smart Servo.

1: Choosing a Text Editor

The code that comes pre-installed on the Smart Servo is written in Circuit Python. This Python coding can be read and edited by nearly anything that con open a text file. We recommend using a text editing program that will recognize the Python code and colorize it for readability. Here are three text editors we recommend:

CODE PAD

- Best option for Chromebooks
- Free Chrome Extension
- Install guide at tinyurl.com/CodePadInstall

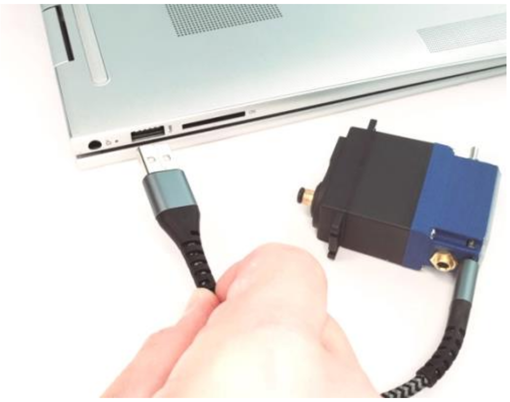
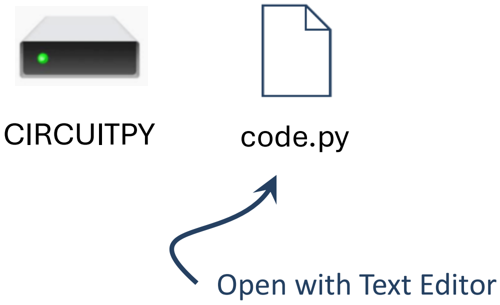
SUBLIME TEXT

- Works on Mac & PC (Used in guides below)
- Free install
- Download at https://www.sublimetext.com/

CIRCUIT PYTHON CODE

- Web-based
- Includes Serial Monitor
- Access at https://code.circuitpython.org/

2: Accessing the Programming that Came with Your Smart Servo

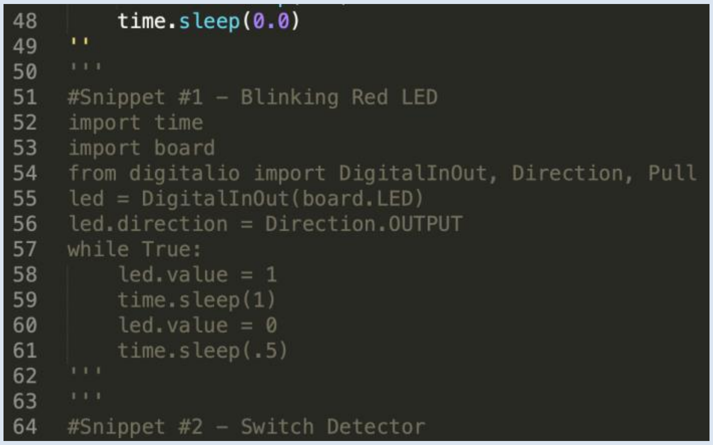
Step 1: Connect one end of the Micro-USB cable to the Smart Servo and the other end to the USB port of the computer you want to use to program. 	**Step 2:** Like an external drive, the Smart Servo will appear with the name **CIRCUITPY**. Open this drive and find the file **code.py**. Open this file with your text editor. 

SMART SERVO GUIDES: RE-CODING

3: Starting with ‘snips’ of code

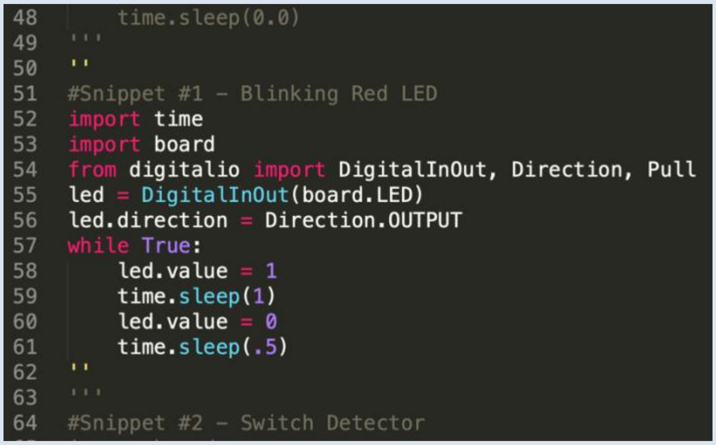
The base code that is used in the Getting Started Guide was developed and tested in Adapted Physical Education equipment created for students with physical disabilities. This default code can be found in the first lines of the code.py file. This code can be edited, saved, and tested.

To start working with “snips” of code, we’re going to use three single apostrophes written as ‘’’ before and after the base code. This will gray-out the code and cause the Smart Servo to ignore this code when it runs. To start working with a snip, simple remove one of the apostrophes before and after the snip (‘’’→’’)



```
48      time.sleep(0.0)
49  ''
50  '''
51  #Snippet #1 - Blinking Red LED
52  import time
53  import board
54  from digitalio import DigitalInOut, Direction, Pull
55  led = DigitalInOut(board.LED)
56  led.direction = Direction.OUTPUT
57  while True:
58      led.value = 1
59      time.sleep(1)
60      led.value = 0
61      time.sleep(.5)
62  '''
63  '''
64  #Snippet #2 - Switch Detector
```

Smart Servo ignores all code after line 49.

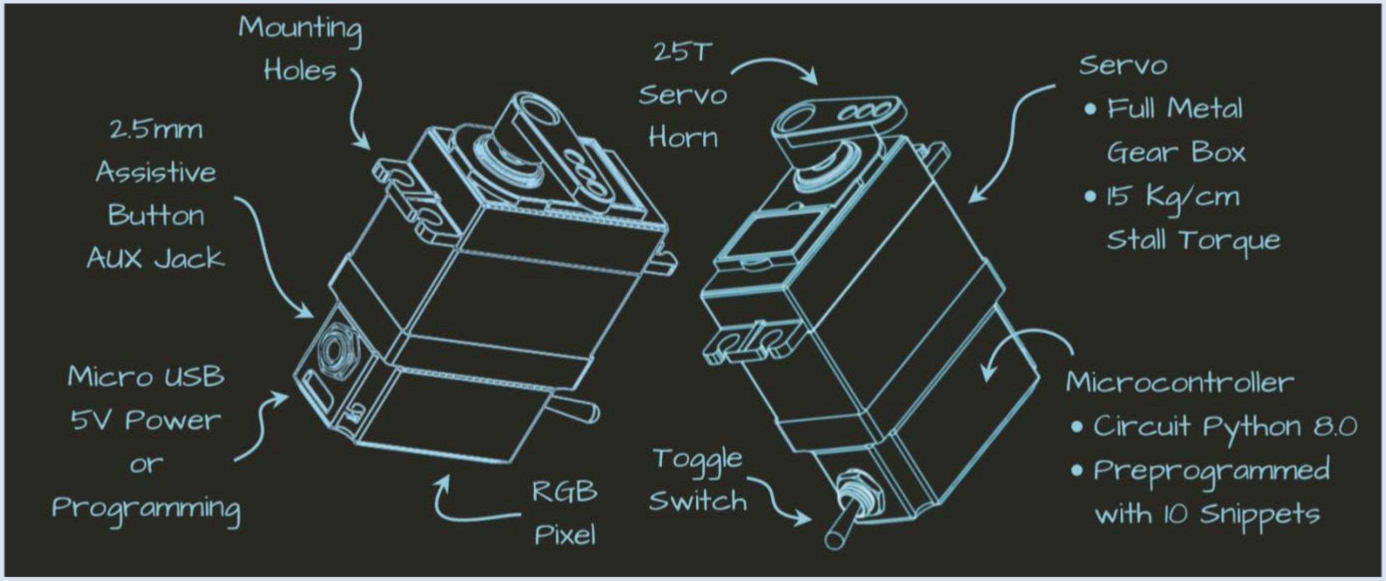


```
48      time.sleep(0.0)
49  '''
50  ''
51  #Snippet #1 - Blinking Red LED
52  import time
53  import board
54  from digitalio import DigitalInOut, Direction, Pull
55  led = DigitalInOut(board.LED)
56  led.direction = Direction.OUTPUT
57  while True:
58      led.value = 1
59      time.sleep(1)
60      led.value = 0
61      time.sleep(.5)
62  ''
63  '''
64  #Snippet #2 - Switch Detector
```

Smart Servo ignores all code except lines 50 to 62.

4: Parts of the Smart Servo

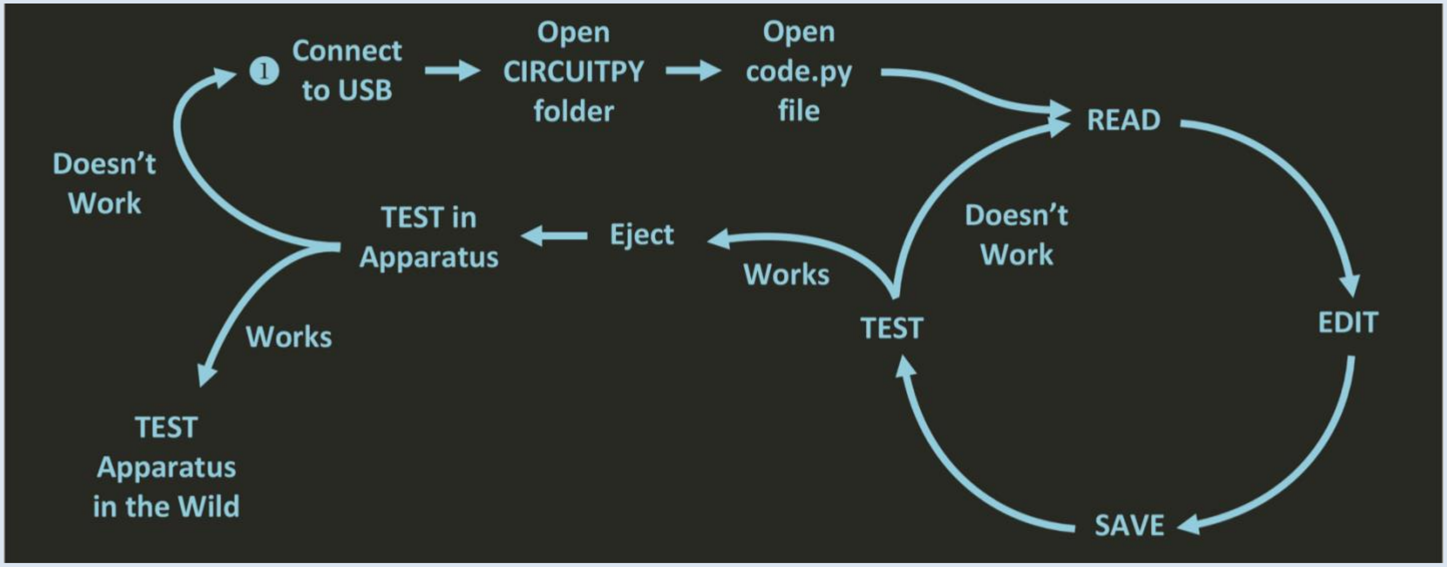
As some of the code refers to specific components of the Smart Servo, the diagram below gives us a chance to review some of the main features.



SMART SERVO GUIDES: RE-CODING

5: Feedback Loop

Because the Smart Servo's micro controller does all of the code compiling right in real time, it's easy to establish a rapid feedback loop. This graphics below summarizes this starting with ① where we connected the Smart Servo to a computer.

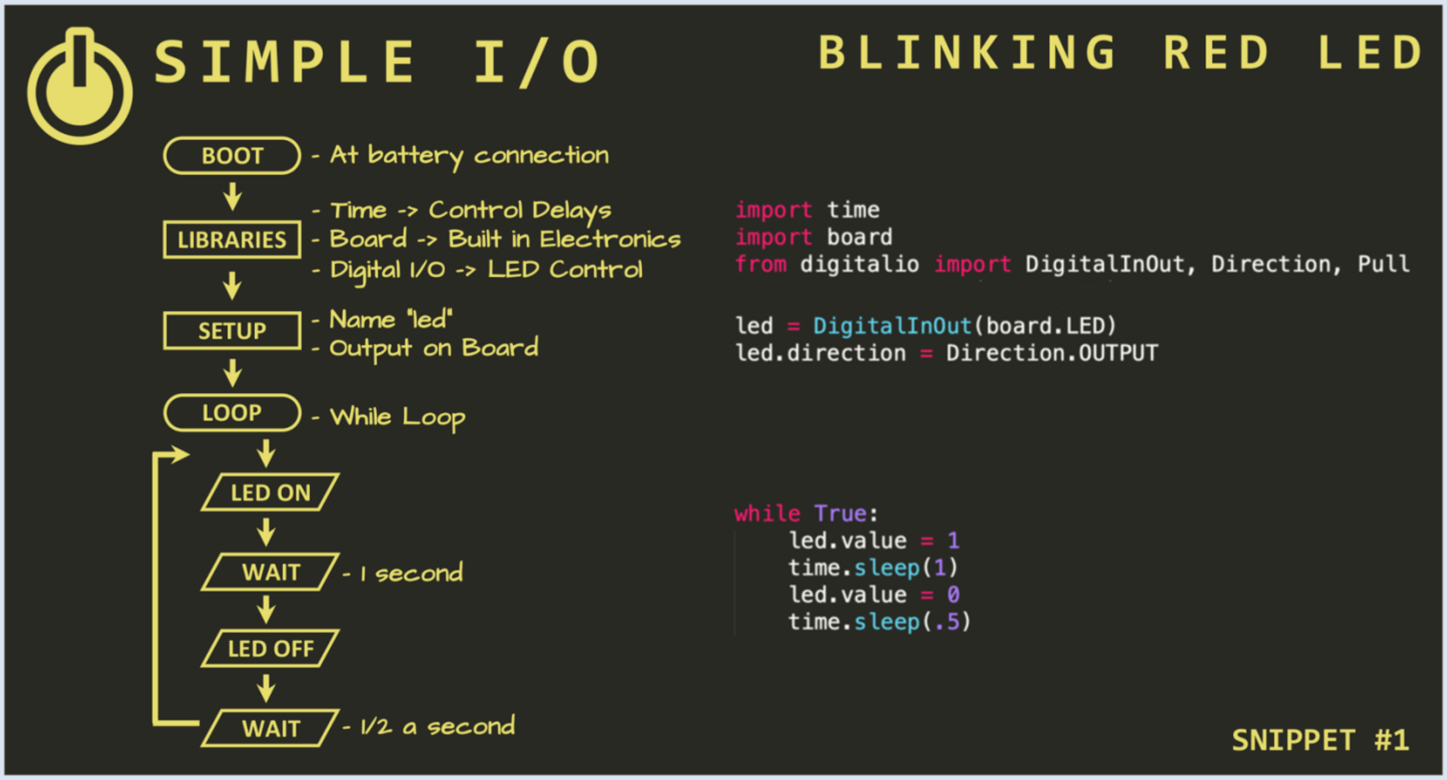


6: Work through each Snippet.

It is recommended that beginners start by working through each snippet on the Smart Servo. Each snippet introduces a new input or output or shows how different strategies can be used to make the Smart Servo more sophisticated in what it can do. With each snippet, try to mess with some of the code to see what it does. If something stops working, undo the step that made it stop working. If things get really out of whack, replace the files on the Smart Servo with the ones found here:

tinyurl.com/SmartServoSnips

Below, you'll find the snips with some reminders and references. Also included is a flow chart to illustrate the logic behind each snippet.



SIMPLE I/O

BLINKING RED LED

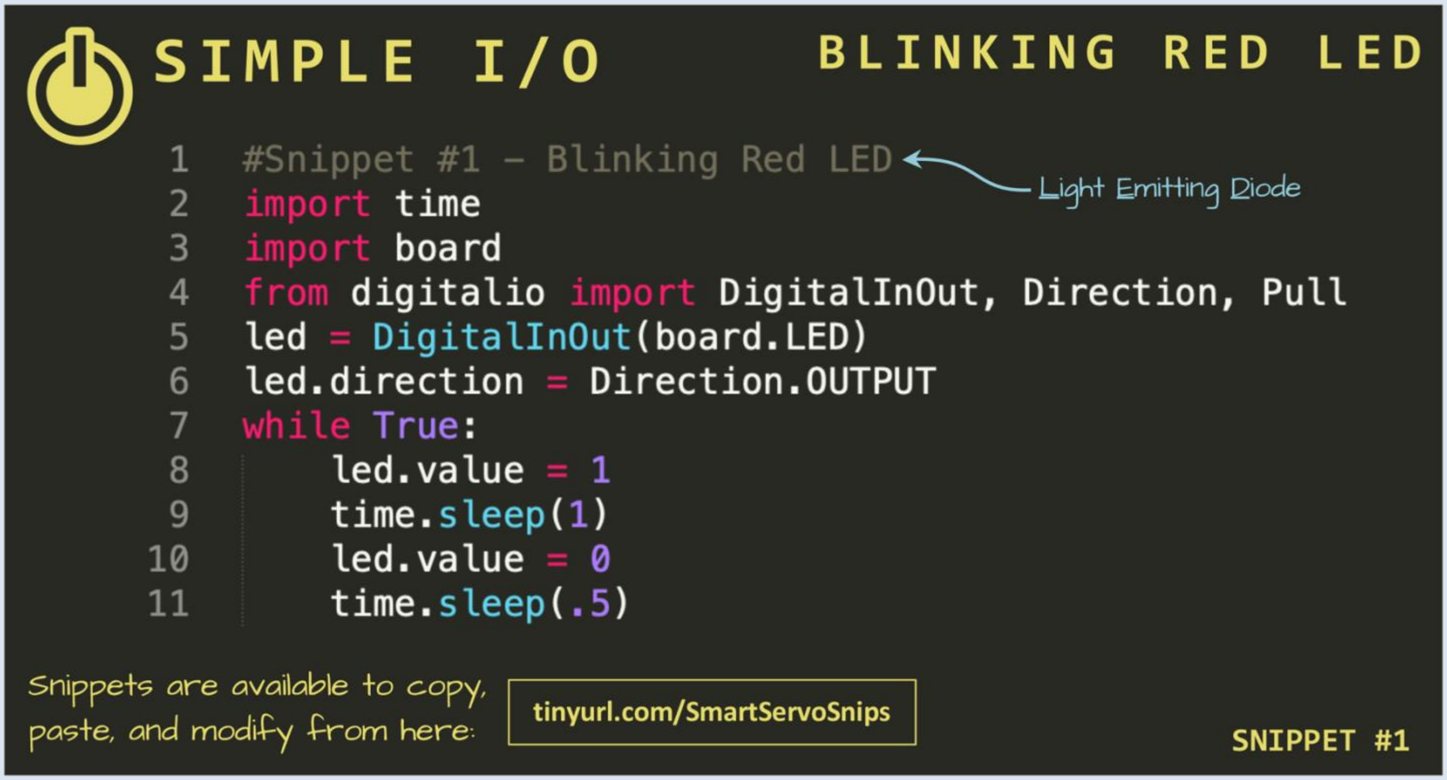
```
import time
import board
from digitalio import DigitalInOut, Direction, Pull

led = DigitalInOut(board.LED)
led.direction = Direction.OUTPUT



while True:
    led.value = 1
    time.sleep(1)
    led.value = 0
    time.sleep(.5)
```

SNIPPET #1



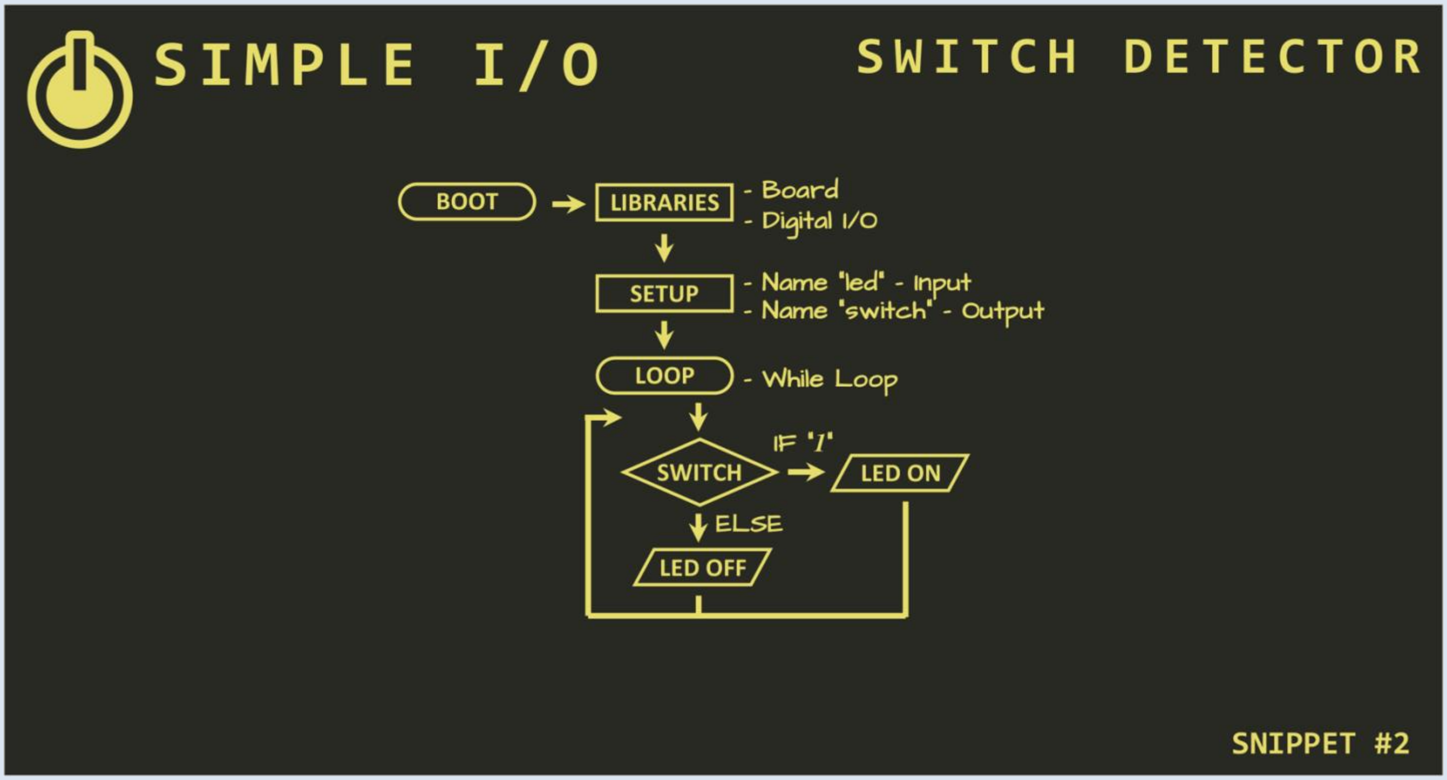
SIMPLE I/O

BLINKING RED LED

```
#Snippet #1 - Blinking Red LED
import time
import board
from digitalio import DigitalInOut, Direction, Pull
led = DigitalInOut(board.LED)
led.direction = Direction.OUTPUT
while True:
    led.value = 1
    time.sleep(1)
    led.value = 0
    time.sleep(.5)
```

Snippets are available to copy, paste, and modify from here: tinyurl.com/SmartServoSnips

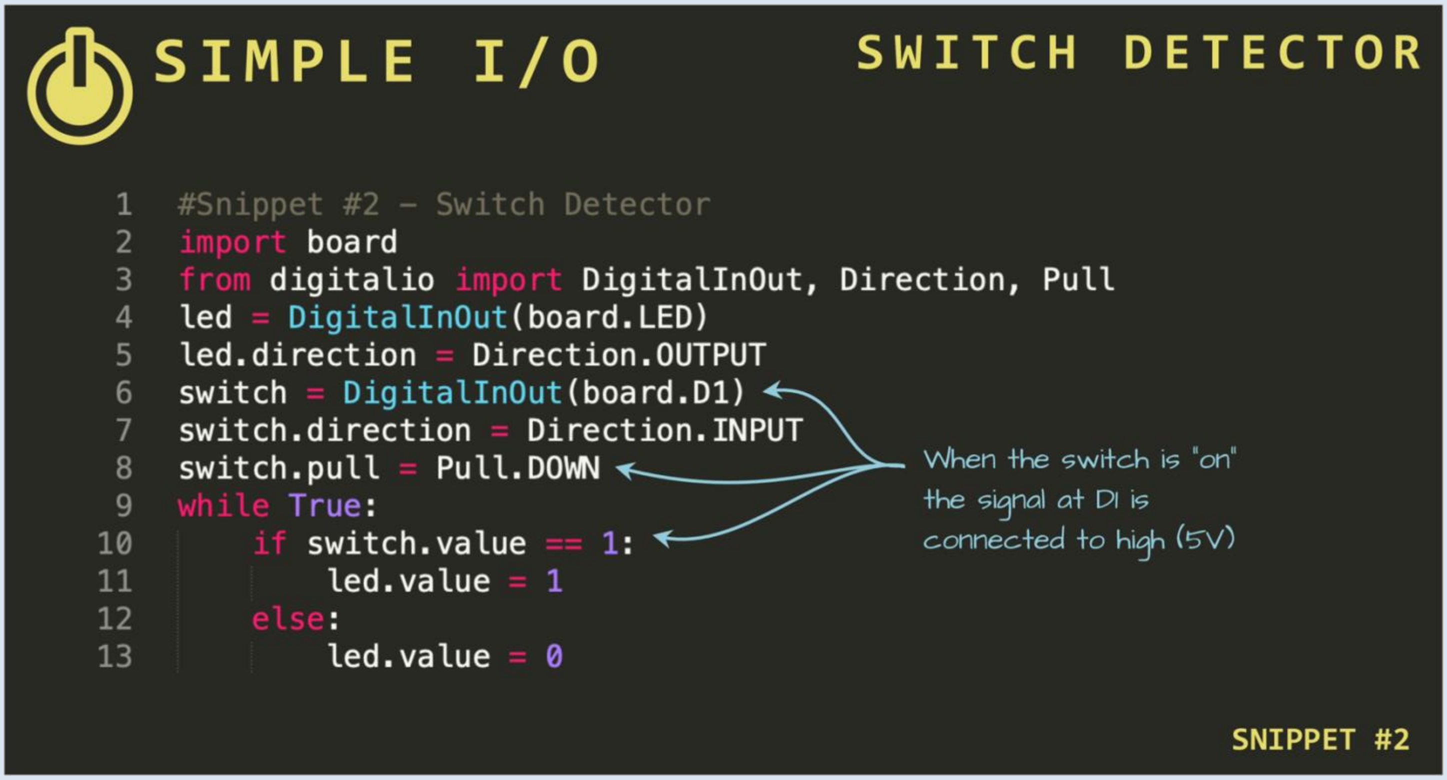
SNIPPET #1



SIMPLE I/O

SWITCH DETECTOR

SNIPPET #2



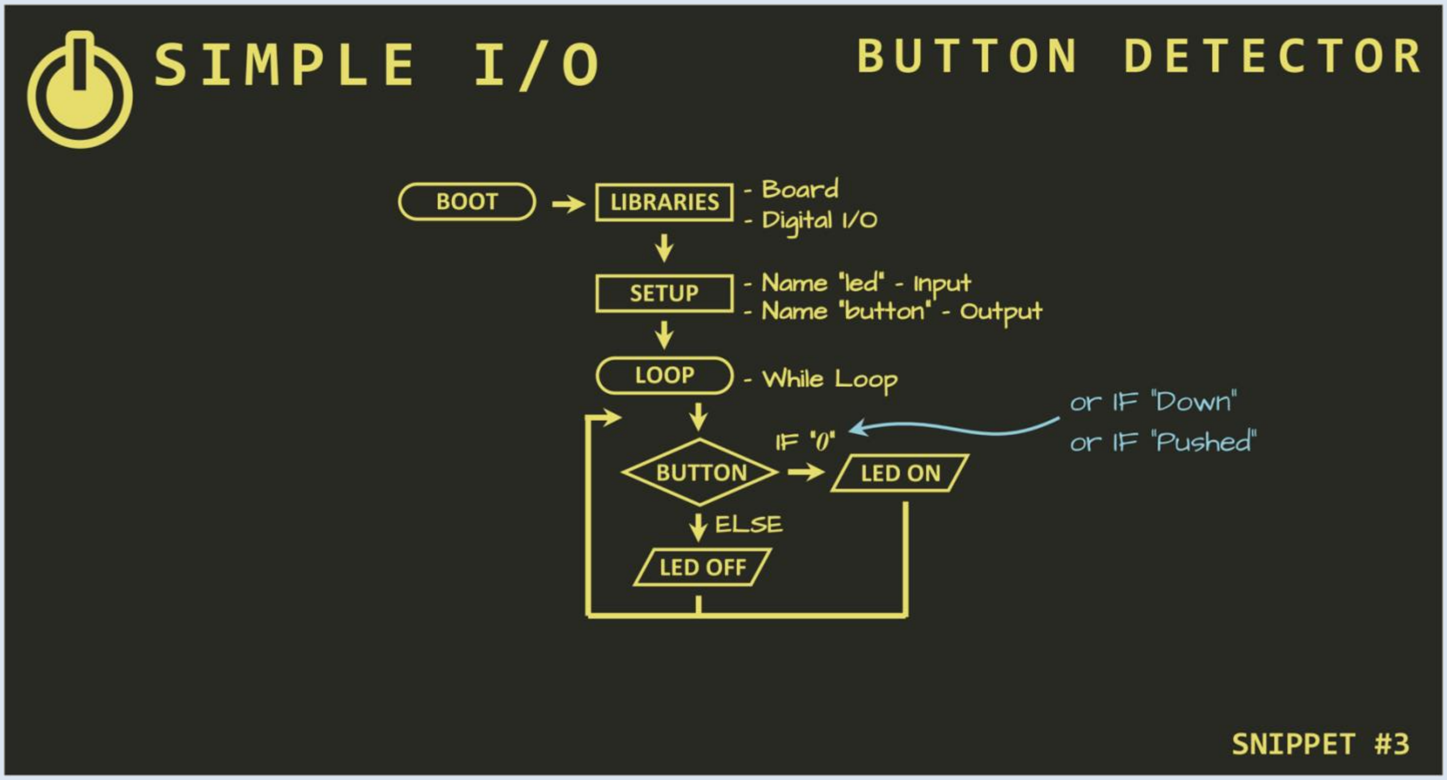
SIMPLE I/O

SWITCH DETECTOR

```
#Snippet #2 - Switch Detector
import board
from digitalio import DigitalInOut, Direction, Pull
led = DigitalInOut(board.LED)
led.direction = Direction.OUTPUT
switch = DigitalInOut(board.D1)
switch.direction = Direction.INPUT
switch.pull = Pull.DOWN
while True:
    if switch.value == 1:
        led.value = 1
    else:
        led.value = 0
```

When the switch is "on" the signal at D1 is connected to high (5V)

SNIPPET #2



SIMPLE I/O

BUTTON DETECTOR

BOOT → LIBRARIES
- Board
- Digital I/O

SETUP
- Name "led" - Input
- Name "button" - Output

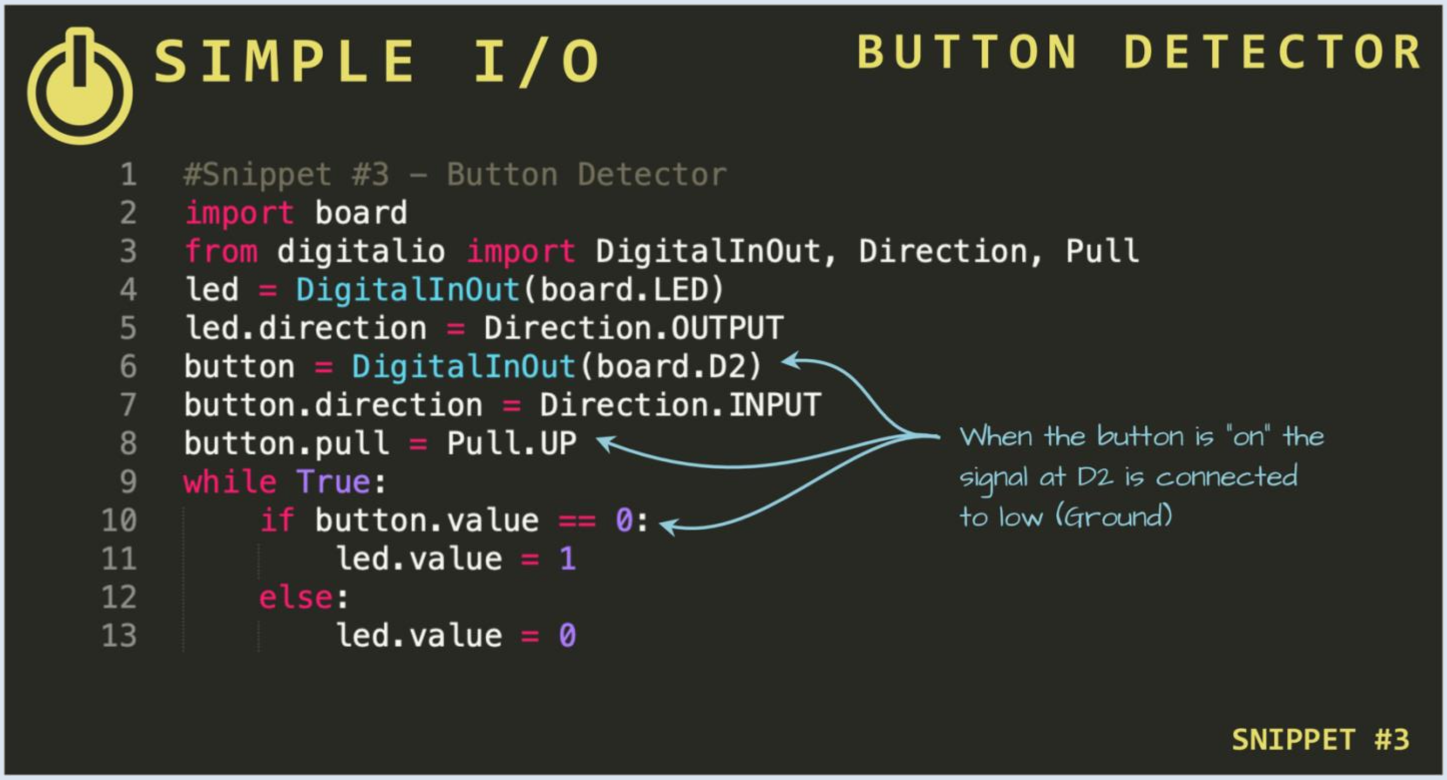
LOOP
- While Loop

BUTTON → IF "0" → LED ON

ELSE → LED OFF

or IF "Down"
or IF "Pushed"

SNIPPET #3



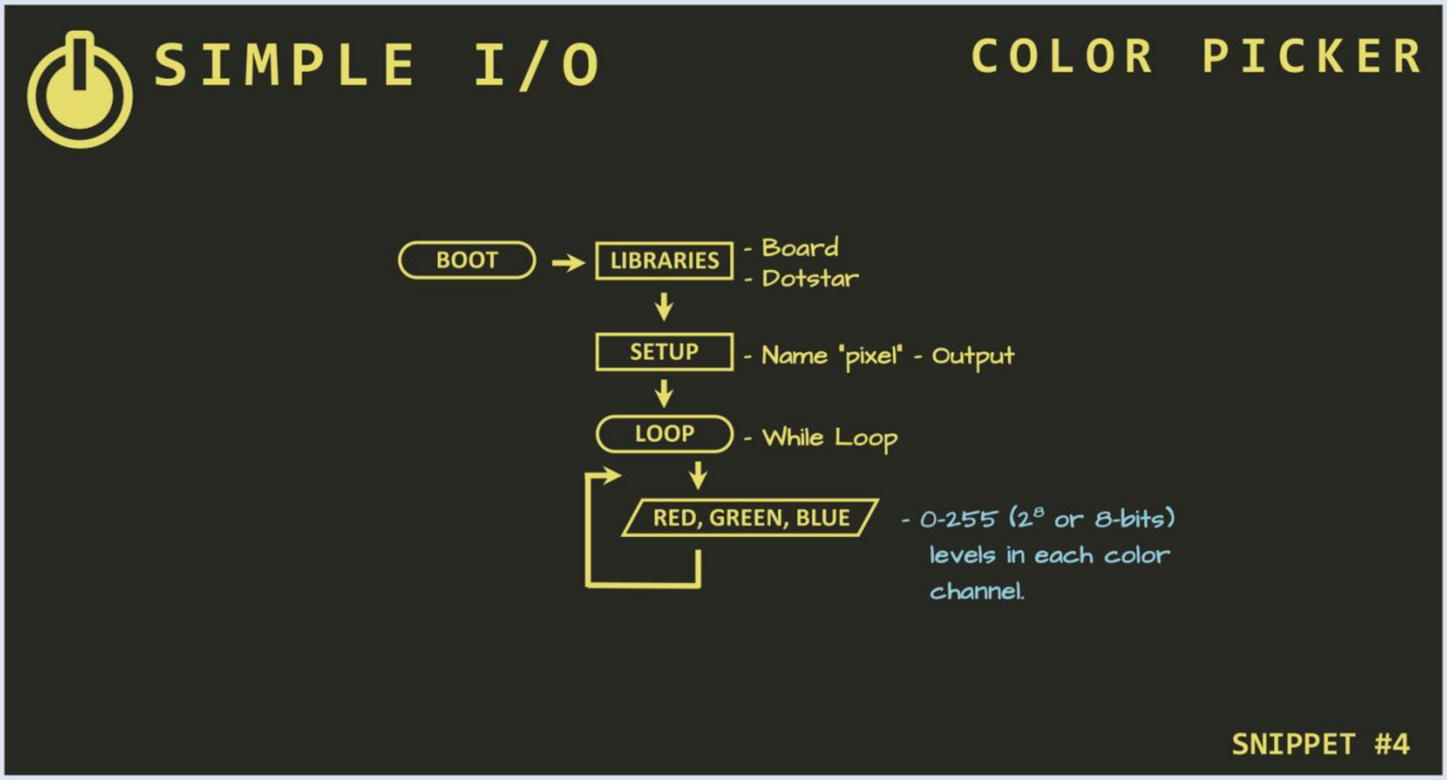
SIMPLE I/O

BUTTON DETECTOR

```
#Snippet #3 – Button Detector
import board
from digitalio import DigitalInOut, Direction, Pull
led = DigitalInOut(board.LED)
led.direction = Direction.OUTPUT
button = DigitalInOut(board.D2)
button.direction = Direction.INPUT
button.pull = Pull.UP
while True:
    if button.value == 0:
        led.value = 1
    else:
        led.value = 0
```

When the button is "on" the signal at D2 is connected to low (Ground)

SNIPPET #3

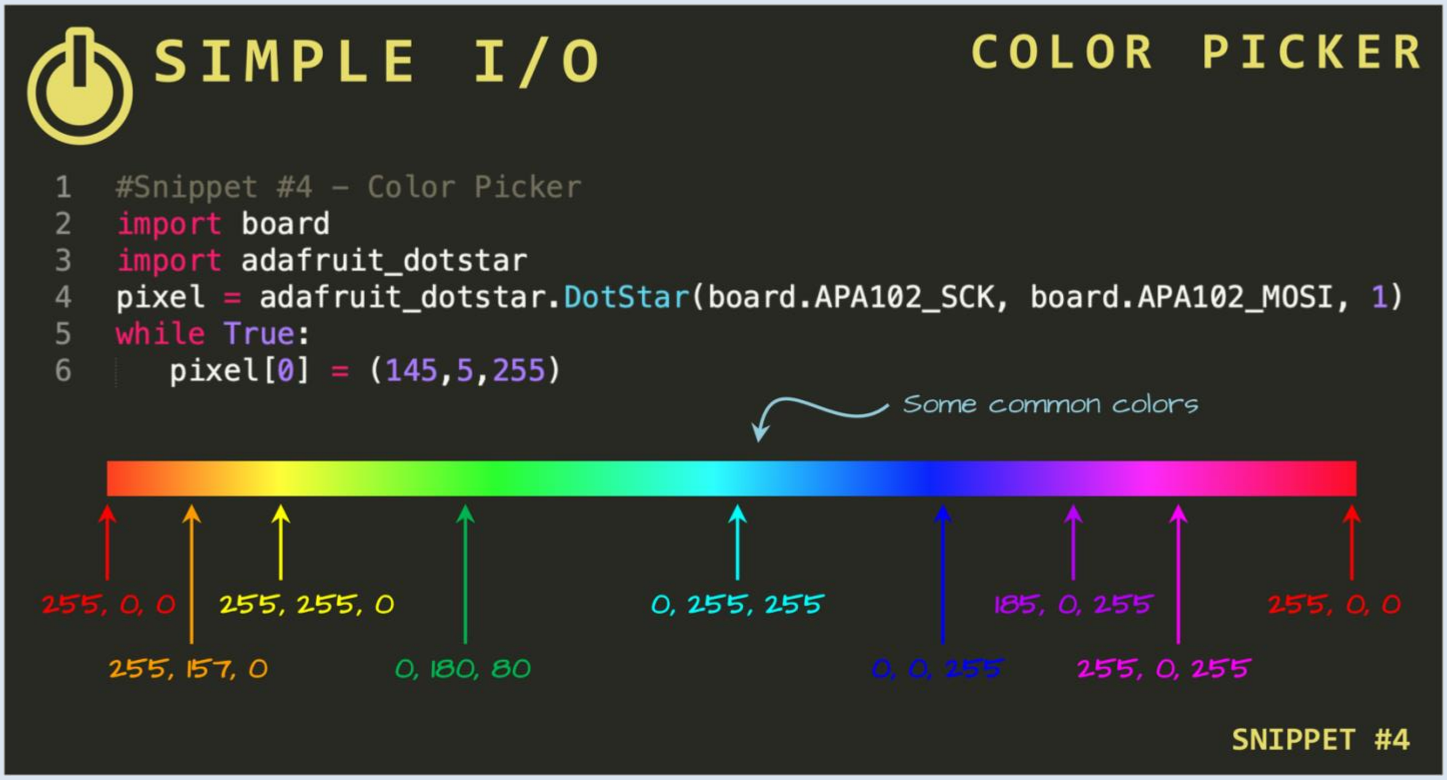


SIMPLE I/O

COLOR PICKER

- 0-255 (2^8 or 8-bits) levels in each color channel.

SNIPPET #4



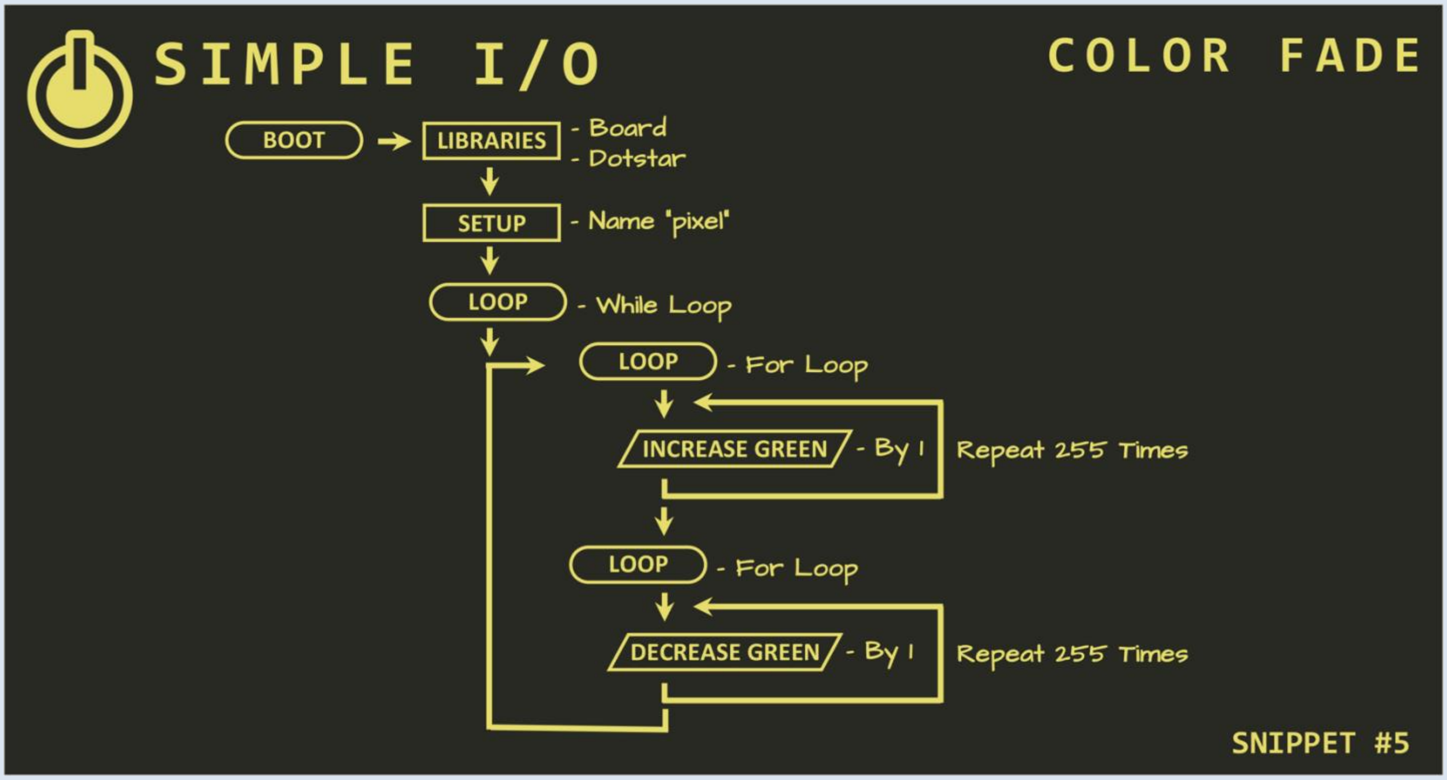
SIMPLE I/O

COLOR PICKER

```
#Snippet #4 - Color Picker
import board
import adafruit_dotstar
pixel = adafruit_dotstar.DotStar(board.APA102_SCK, board.APA102_MOSI, 1)
while True:
    pixel[0] = (145,5,255)
```

Some common colors

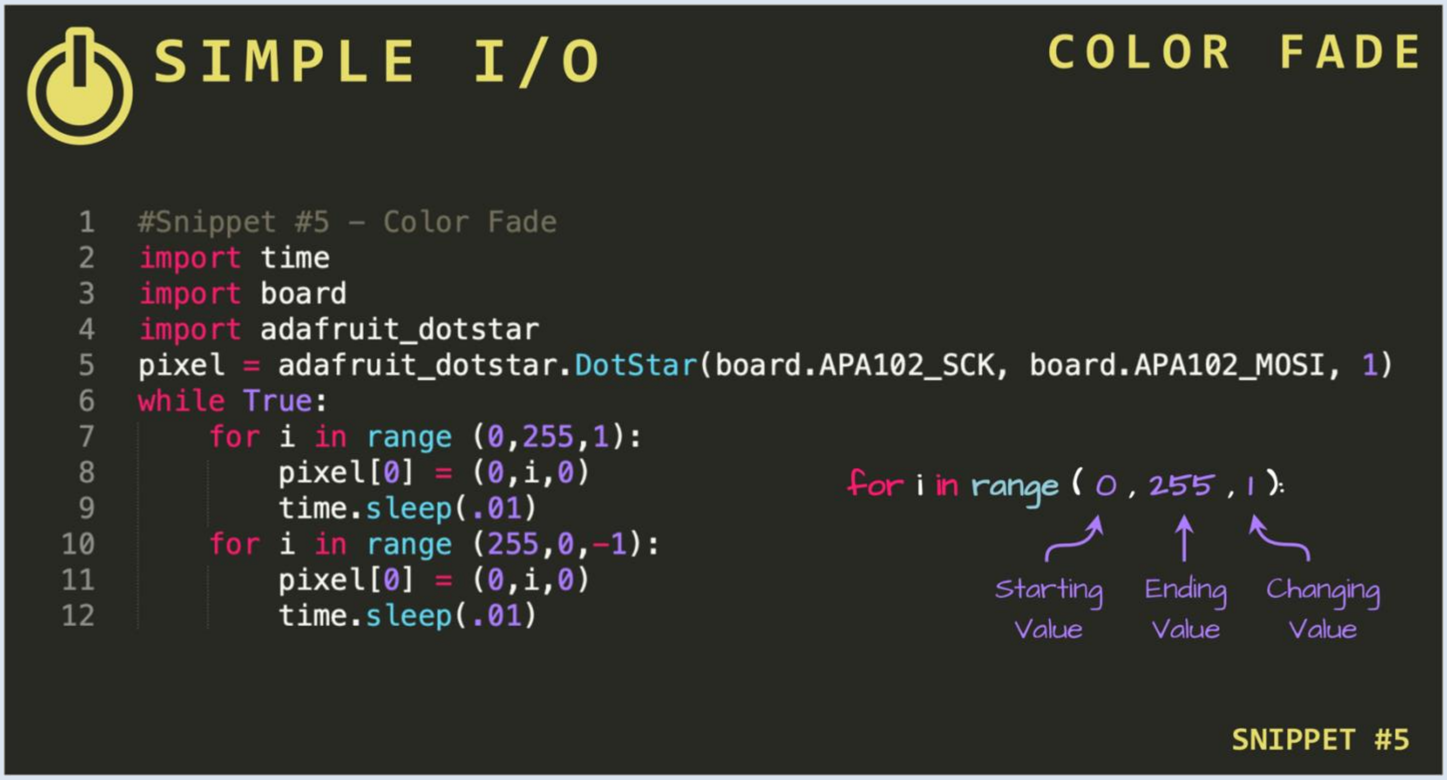
SNIPPET #4



SIMPLE I/O

COLOR FADE

SNIPPET #5



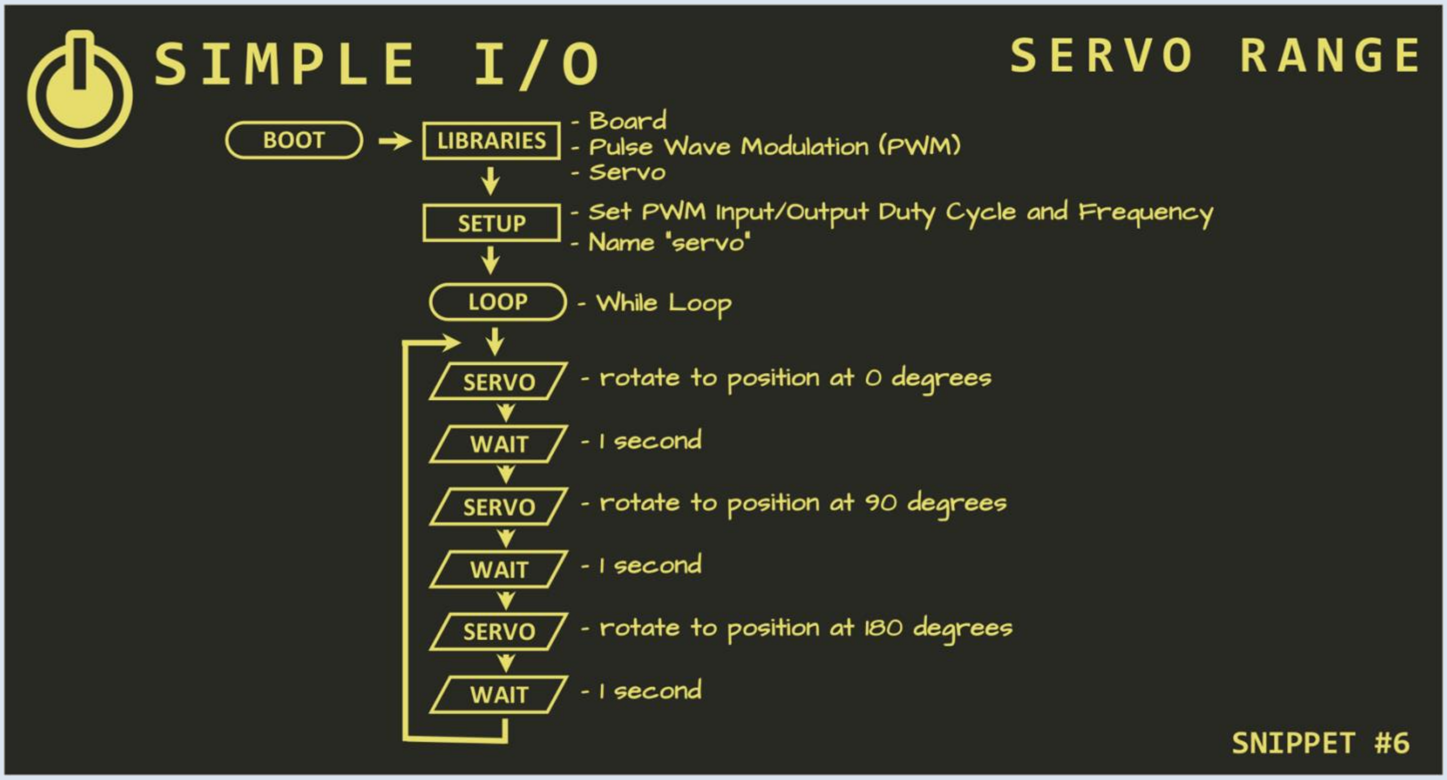
SIMPLE I/O

COLOR FADE

```
#Snippet #5 – Color Fade
import time
import board
import adafruit_dotstar
pixel = adafruit_dotstar.DotStar(board.APA102_SCK, board.APA102_MOSI, 1)
while True:
    for i in range (0,255,1):
        pixel[0] = (0,i,0)
        time.sleep(.01)
    for i in range (255,0,-1):
        pixel[0] = (0,i,0)
        time.sleep(.01)
```

for i in range (0 , 255 , 1):

SNIPPET #5

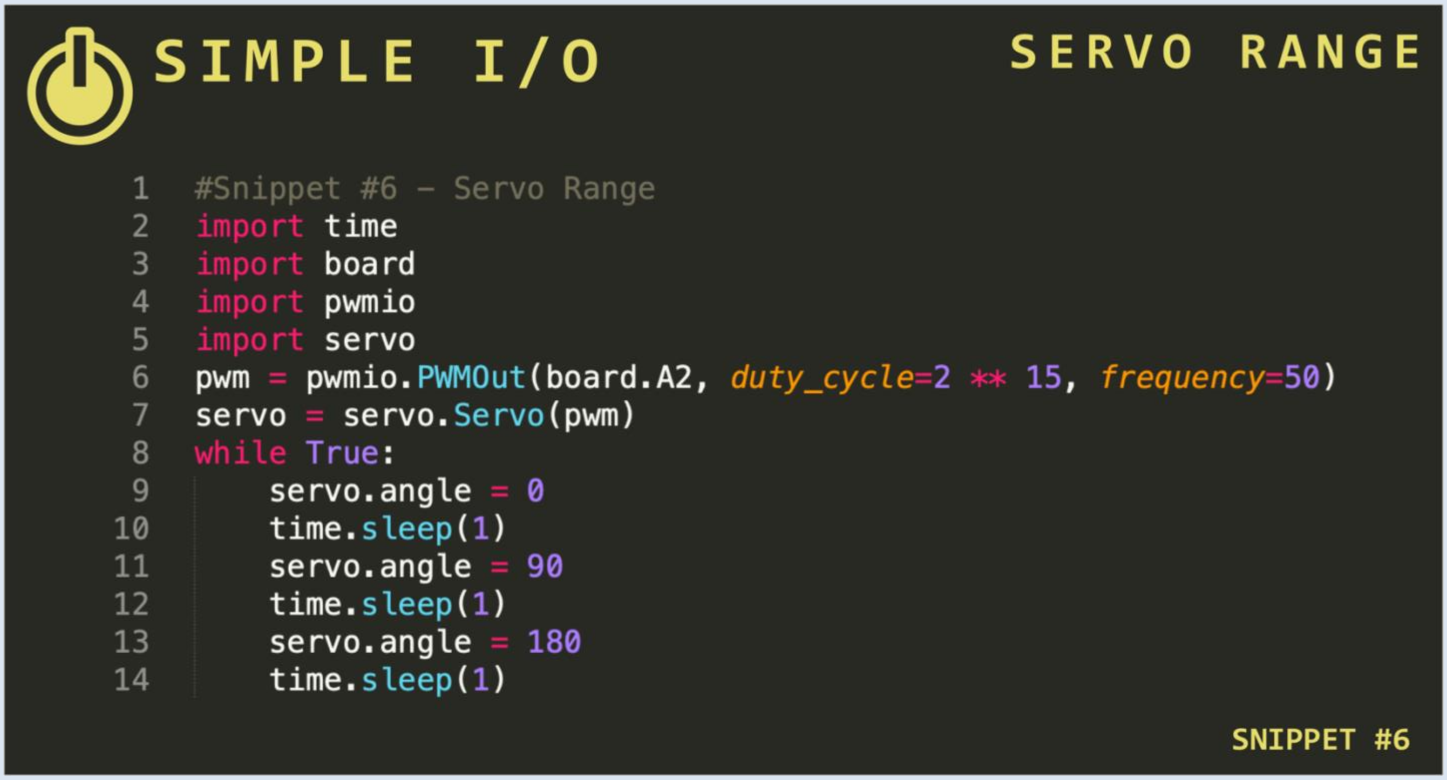


SIMPLE I/O

SERVO RANGE

BOOT
LIBRARIES
- Board
- Pulse Wave Modulation (PWM)
- Servo
SETUP
- Set PWM Input/Output Duty Cycle and Frequency
- Name 'servo'
LOOP
- While Loop
SERVO
- rotate to position at 0 degrees
WAIT
- 1 second
SERVO
- rotate to position at 90 degrees
WAIT
- 1 second
SERVO
- rotate to position at 180 degrees
WAIT
- 1 second

SNIPPET #6

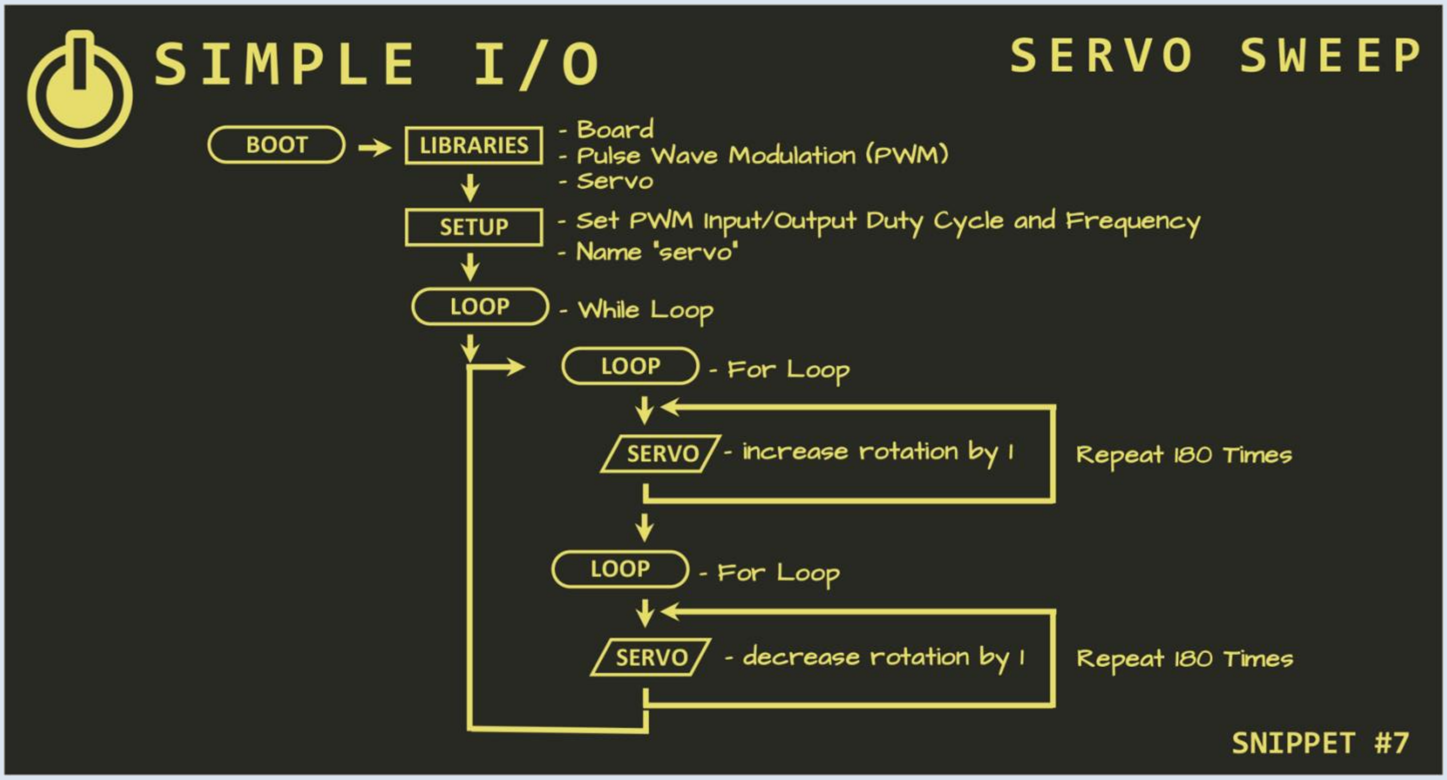


SIMPLE I/O

SERVO RANGE

```
#Snippet #6 - Servo Range
import time
import board
import pwmio
import servo
pwm = pwmio.PWMOut(board.A2, duty_cycle=2 ** 15, frequency=50)
servo = servo.Servo(pwm)
while True:
    servo.angle = 0
    time.sleep(1)
    servo.angle = 90
    time.sleep(1)
    servo.angle = 180
    time.sleep(1)
```

SNIPPET #6



SIMPLE I/O

SERVO SWEEP

BOOT → LIBRARIES
- Board
- Pulse Wave Modulation (PWM)
- Servo

SETUP
- Set PWM Input/Output Duty Cycle and Frequency
- Name 'servo'

LOOP - While Loop

LOOP - For Loop

SERVO - increase rotation by 1

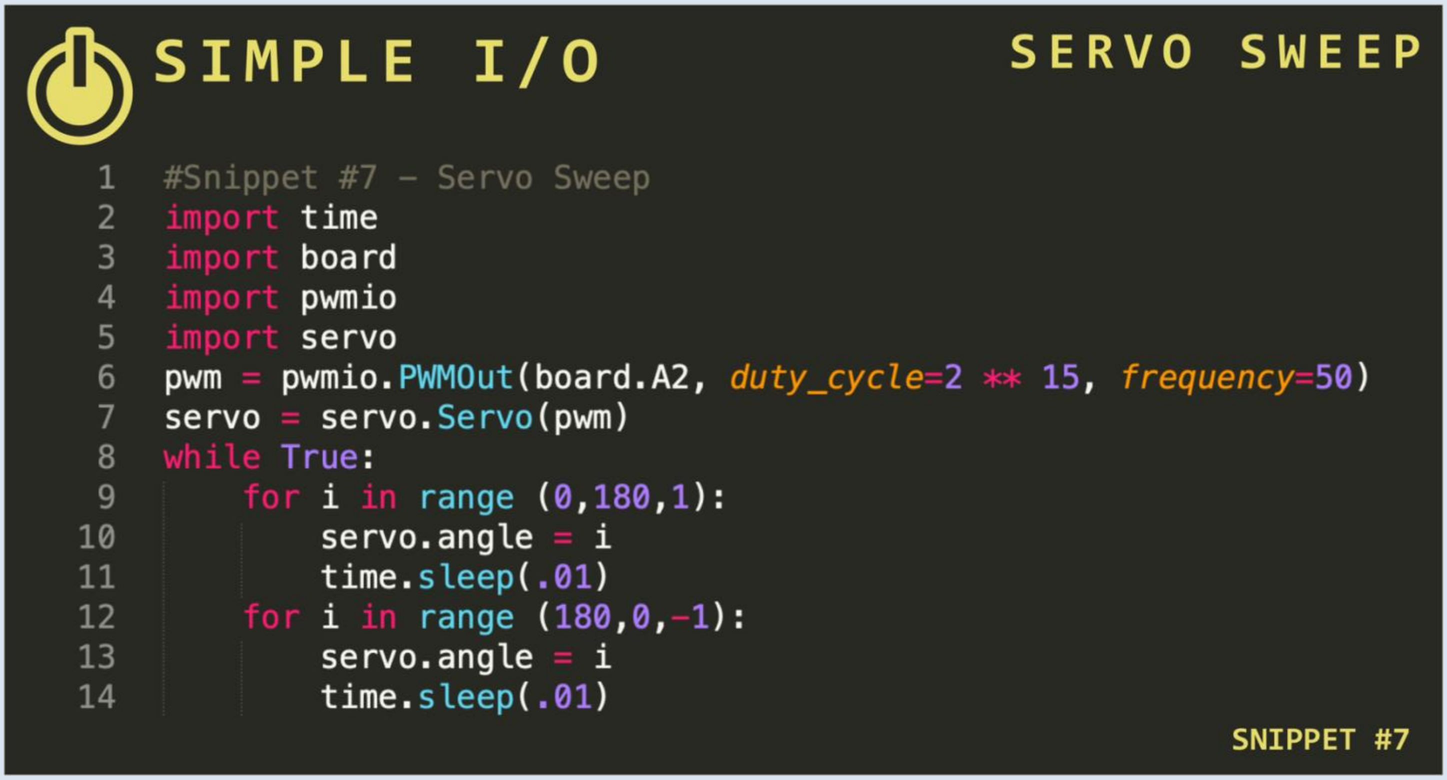
Repeat 180 Times

LOOP - For Loop

SERVO - decrease rotation by 1

Repeat 180 Times

SNIPPET #7

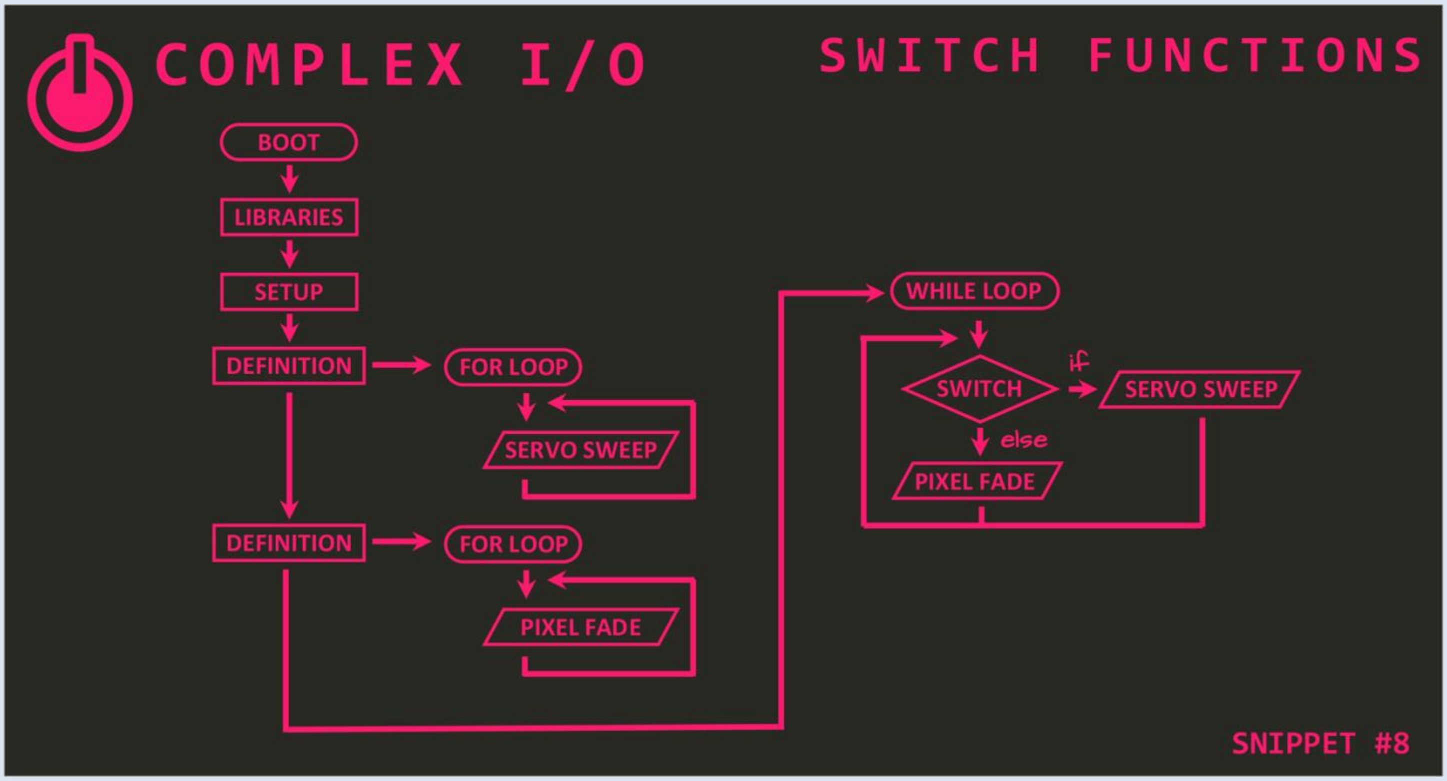


SIMPLE I/O

SERVO SWEEP

```
#Snippet #7 - Servo Sweep
import time
import board
import pwmio
import servo
pwm = pwmio.PWMOut(board.A2, duty_cycle=2 ** 15, frequency=50)
servo = servo.Servo(pwm)
while True:
    for i in range (0,180,1):
        servo.angle = i
        time.sleep(.01)
    for i in range (180,0,-1):
        servo.angle = i
        time.sleep(.01)
```

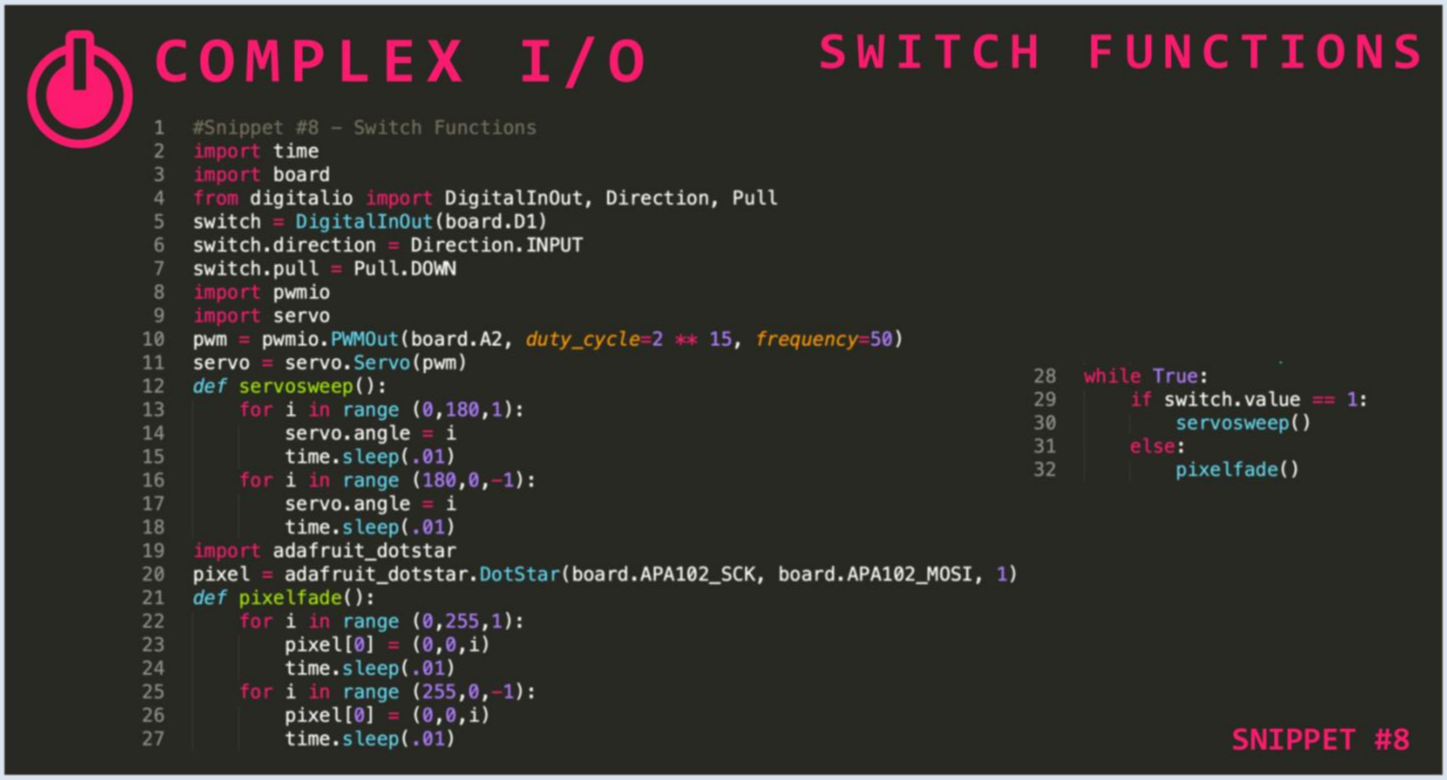
SNIPPET #7



COMPLEX I/O

SWITCH FUNCTIONS

SNIPPET #8



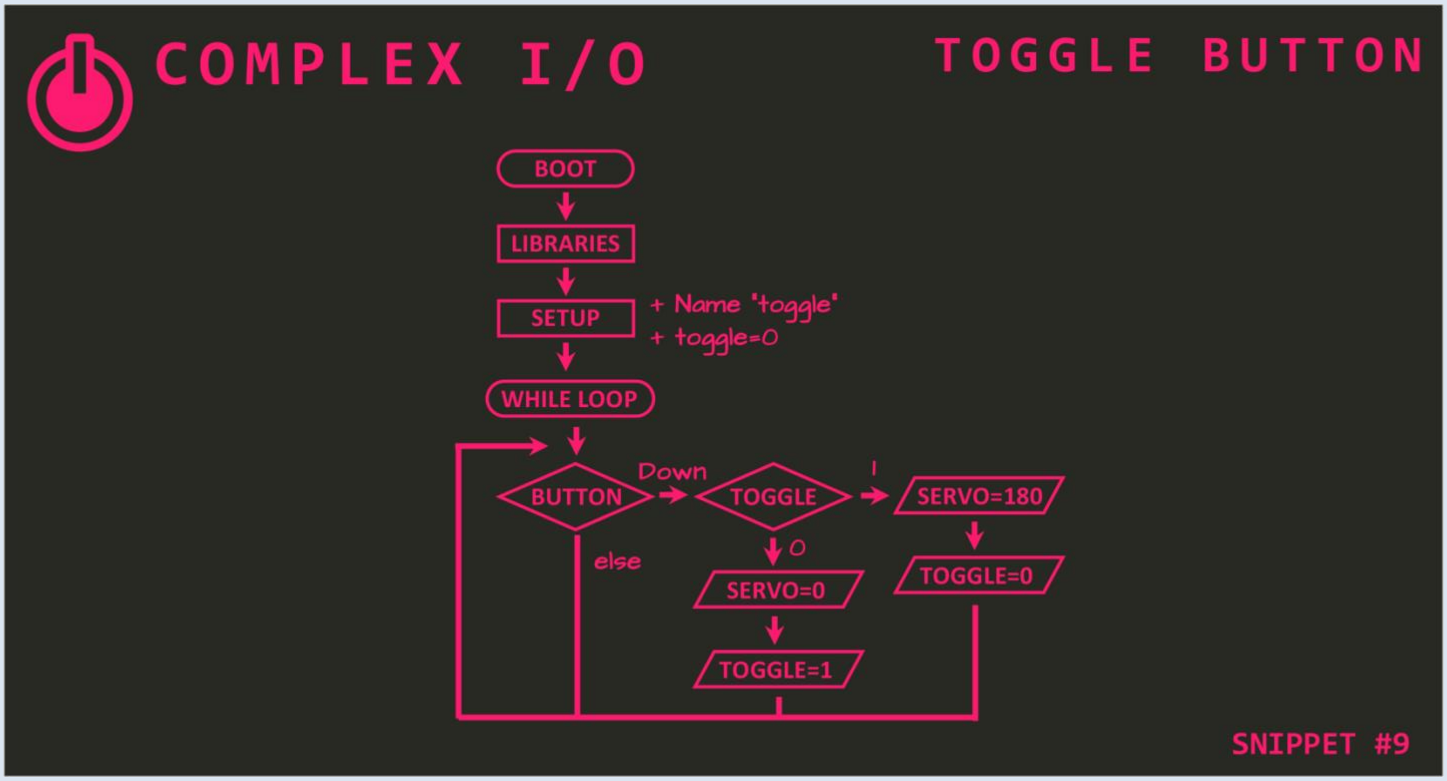
COMPLEX I/O

SWITCH FUNCTIONS

```
#Snippet #8 - Switch Functions
import time
import board
from digitalio import DigitalInOut, Direction, Pull
switch = DigitalInOut(board.D1)
switch.direction = Direction.INPUT
switch.pull = Pull.DOWN
import pwmio
import servo
pwm = pwmio.PWMOut(board.A2, duty_cycle=2 ** 15, frequency=50)
servo = servo.Servo(pwm)
def servosweep():
    for i in range (0,180,1):
        servo.angle = i
        time.sleep(.01)
    for i in range (180,0,-1):
        servo.angle = i
        time.sleep(.01)
import adafruit_dotstar
pixel = adafruit_dotstar.DotStar(board.APA102_SCK, board.APA102_MOSI, 1)
def pixelfade():
    for i in range (0,255,1):
        pixel[0] = (0,0,i)
        time.sleep(.01)
    for i in range (255,0,-1):
        pixel[0] = (0,0,i)
        time.sleep(.01)
```

```
while True:
    if switch.value == 1:
        servosweep()
    else:
        pixelfade()
```

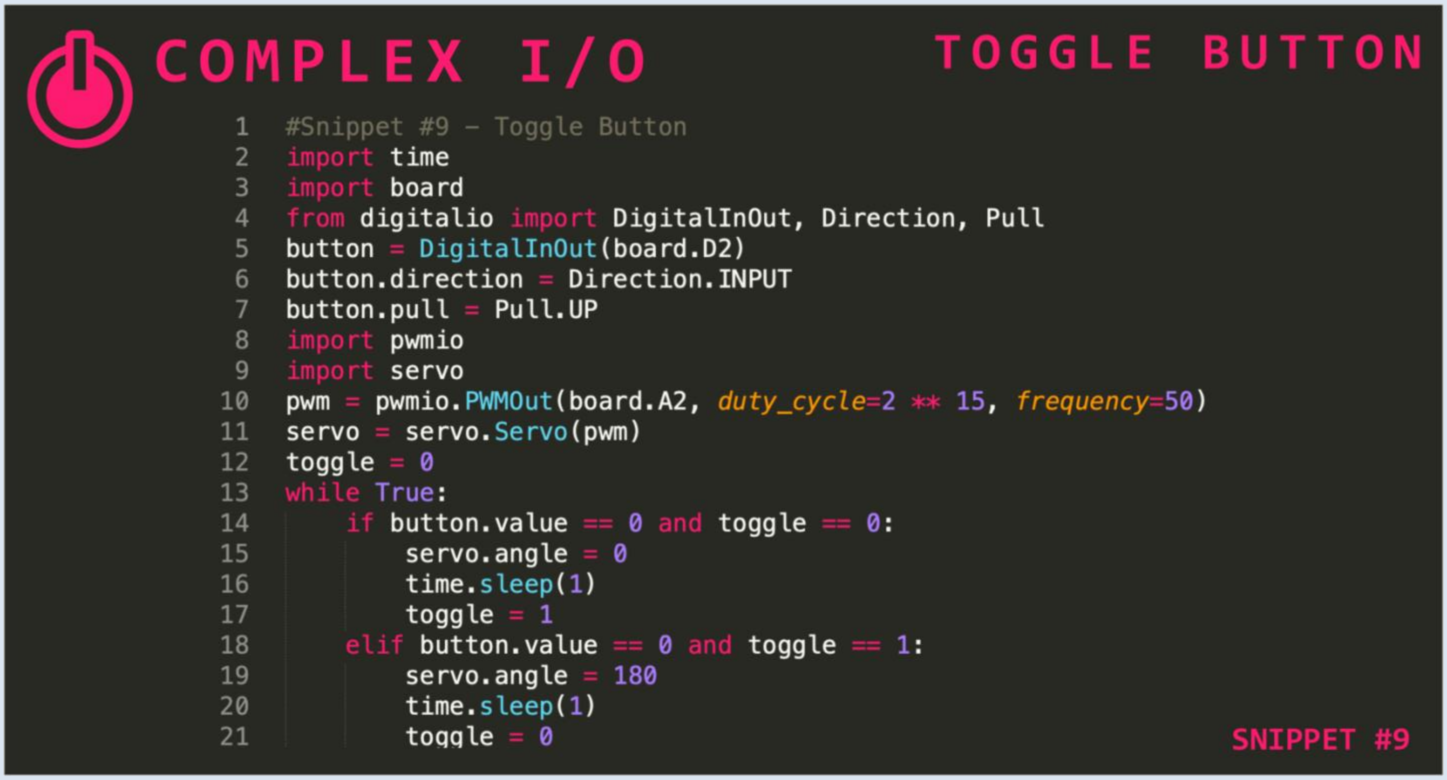
SNIPPET #8



COMPLEX I/O

TOGGLE BUTTON

SNIPPET #9

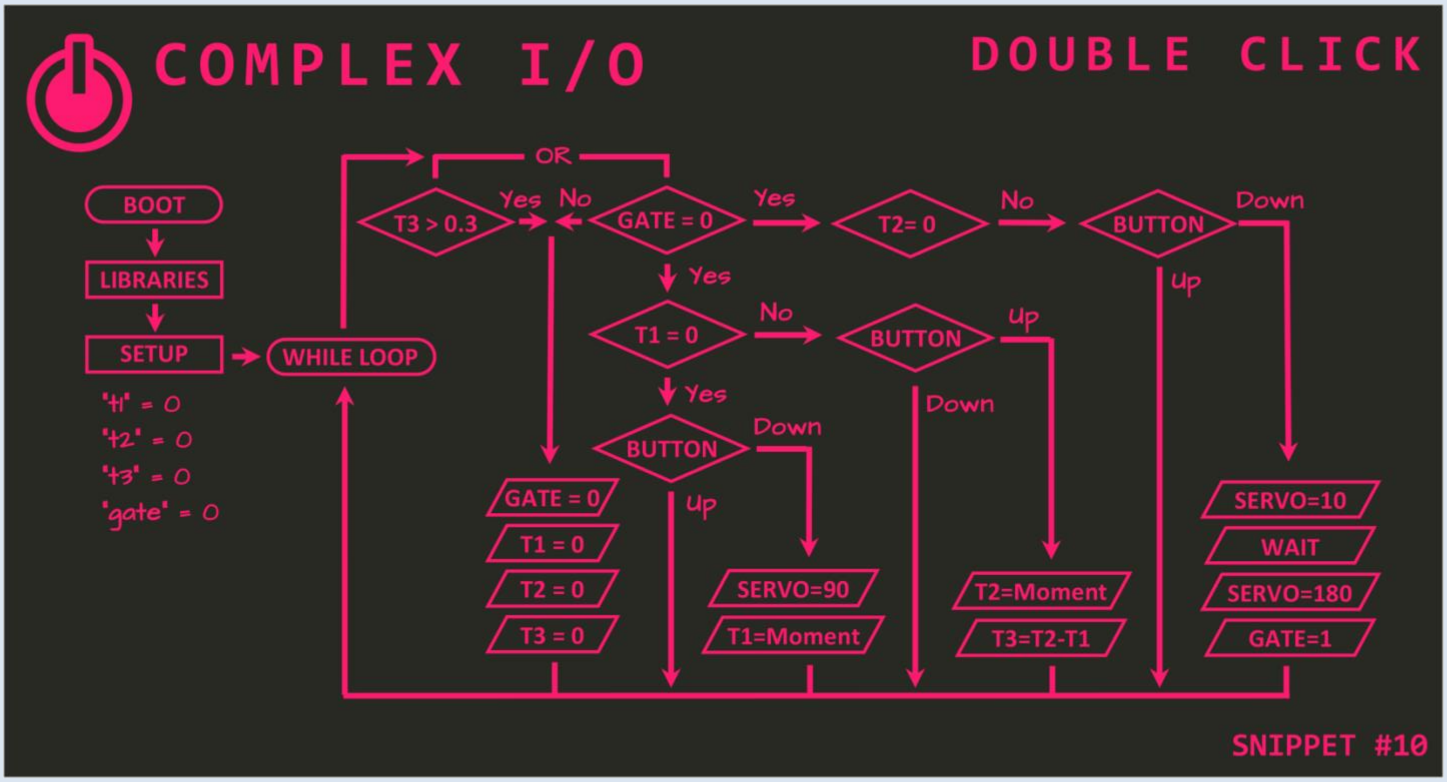


COMPLEX I/O

TOGGLE BUTTON

```
#Snippet #9 - Toggle Button
import time
import board
from digitalio import DigitalInOut, Direction, Pull
button = DigitalInOut(board.D2)
button.direction = Direction.INPUT
button.pull = Pull.UP
import pwmio
import servo
pwm = pwmio.PWMOut(board.A2, duty_cycle=2 ** 15, frequency=50)
servo = servo.Servo(pwm)
toggle = 0
while True:
    if button.value == 0 and toggle == 0:
        servo.angle = 0
        time.sleep(1)
        toggle = 1
    elif button.value == 0 and toggle == 1:
        servo.angle = 180
        time.sleep(1)
        toggle = 0
```

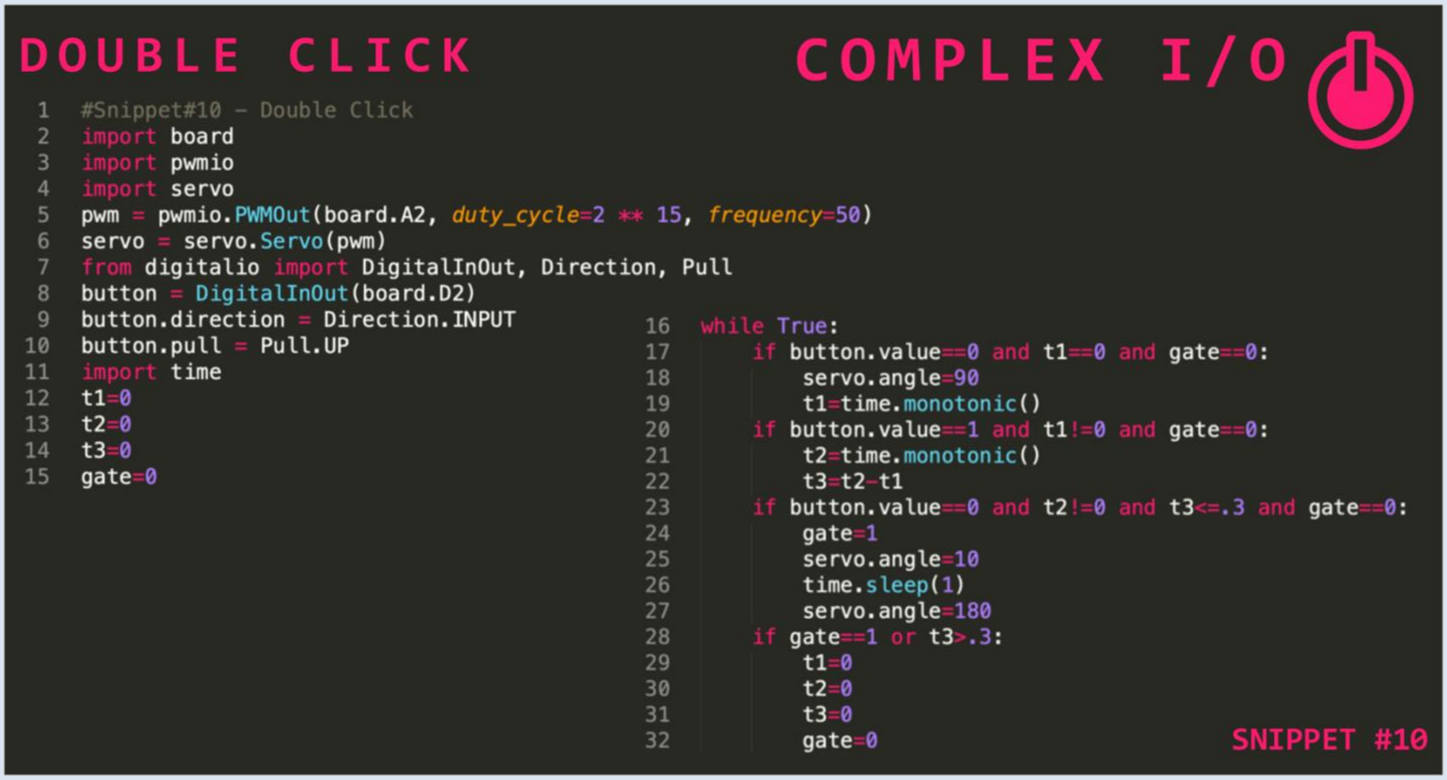
SNIPPET #9



COMPLEX I/O

DOUBLE CLICK

SNIPPET #10



DOUBLE CLICK

COMPLEX I/O

```
#Snippet#10 - Double Click
import board
import pwmio
import servo
pwm = pwmio.PWMOut(board.A2, duty_cycle=2 ** 15, frequency=50)
servo = servo.Servo(pwm)
from digitalio import DigitalInOut, Direction, Pull
button = DigitalInOut(board.D2)
button.direction = Direction.INPUT
button.pull = Pull.UP
import time
t1=0
t2=0
t3=0
gate=0
while True:
    if button.value==0 and t1==0 and gate==0:
        servo.angle=90
        t1=time.monotonic()
    if button.value==1 and t1!=0 and gate==0:
        t2=time.monotonic()
        t3=t2-t1
    if button.value==0 and t2!=0 and t3<=.3 and gate==0:
        gate=1
        servo.angle=10
        time.sleep(1)
        servo.angle=180
    if gate==1 or t3>.3:
        t1=0
        t2=0
        t3=0
        gate=0
```

SNIPPET #10