In this lesson you will connect a LED to your Raspberry Pi and turn the LED on and off using Adafruit IO. You will need an account for Adafruit IO. If you do not have an account, follow the directions in a previous lesson to create an account.

Install the Adafruit IO Python library

If you have not already done so, you will need to install the Adafruit IO Python library on your Raspberry Pi. Open the Terminal and issue this command:

sudo pip install adafruit-io

Connect LED to your Raspberry Pi

1. Connect the anode (long wire) of LED to GPIO pin 23 (physical pin 16).
2. Connect a resistor of about 500 to 1000 ohms to the cathode of the LED
3. Connect the other end of the resistor to a GPIO ground pin (such as physical pin 6)

Prepare Adafruit IO to control the LED

1. Go to web site: https://io.adafruit.com and sign in
2. Create a feed (I named my feed **Bulb1**)
3. Create a dashboard (I named mine LED)
4. Open the dashboard you just created
5. Click the blue button with + symbol to add a block to the dashboard
6. Select the **Toggle** (on/off) block type (this is an on/off switch)
7. In the **Choose feed** step, select the feed you just made
8. In the **Block settings** step, add a title, then click the **Create block** button

Write the Python program for LED control.

You will need to use Python version 2. The code is provided below. You can skip the comments marked with #

```
# Import standard python modules.
import RPi.GPIO as GPIO
import sys

# Import Adafruit IO MQTT client.
from Adafruit_IO import MQTTClient

# use GPIO pin numbering scheme
GPIO.setmode(GPIO.BCM)

# Set to your Adafruit IO key & username below.
ADAFRUIT_IO_USERNAME    =  ' '    # add your user name between the quotes
 ADAFRUIT_IO_KEY        = ' '          # add your Adafruit key between the quotes

# Set to the ID of the feed to subscribe to for updates.
FEED_ID1 = 'Bulb1'          # I used a feed name of Bulb1, change if your name is different
```

```
# GPIO pin 23 set as output type
GPIO.setup(23, GPIO.OUT)

# set pin 23 to low, i.e., LED off
GPIO.output(23, GPIO.LOW)

# Define callback functions which will be called when certain events happen.

def connected(client):

    # Connected function will be called when the client is connected to Adafruit IO.
    # This is a good place to subscribe to feed changes.  The client parameter
    # passed to this function is the Adafruit IO MQTT client so you can make
    # calls against it easily.

    print 'Connected to Adafruit IO!  Listening for {0} changes...'.format(FEED_ID1)

     # Subscribe to changes on the feed
    client.subscribe(FEED_ID1)


def disconnected(client):

    # Disconnected function will be called when the client disconnects.

    print 'Disconnected from Adafruit IO!'

    GPIO.cleanup()

    sys.exit(1)


def message(client, feed_id, payload):

    # Message function will be called when a subscribed feed has a new value.
    # The feed_id parameter identifies the feed, and the payload parameter has
    # the new value.

    print 'Feed {0} received new value: {1}'.format(feed_id, payload)

    if feed_id=="Bulb1" and payload=="ON":

        GPIO.output(23, GPIO.HIGH)  #  i.e., turn on LED

    if feed_id=="Bulb1" and payload=="OFF":

       GPIO.output(23, GPIO.LOW)  # i.e., turn LED off


# Create an MQTT client instance.
client = MQTTClient(ADAFRUIT_IO_USERNAME, ADAFRUIT_IO_KEY)
```

```
# Setup the callback functions defined above.
client.on_connect    = connected
client.on_disconnect = disconnected
client.on_message    = message

# Connect to the Adafruit IO server.
client.connect()

# Start a message loop that blocks forever waiting for MQTT messages to be
# received.  Note there are other options for running the event loop like doing
# so in a background thread--see the mqtt_client.py example to learn more.
client.loop_blocking()
```