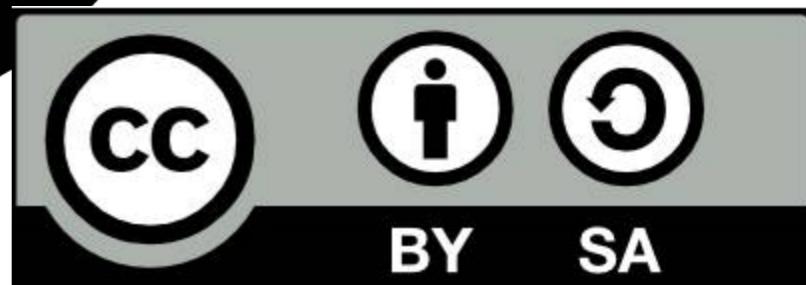


# A Slice of Raspberry Pi

Original By Spencer Martin

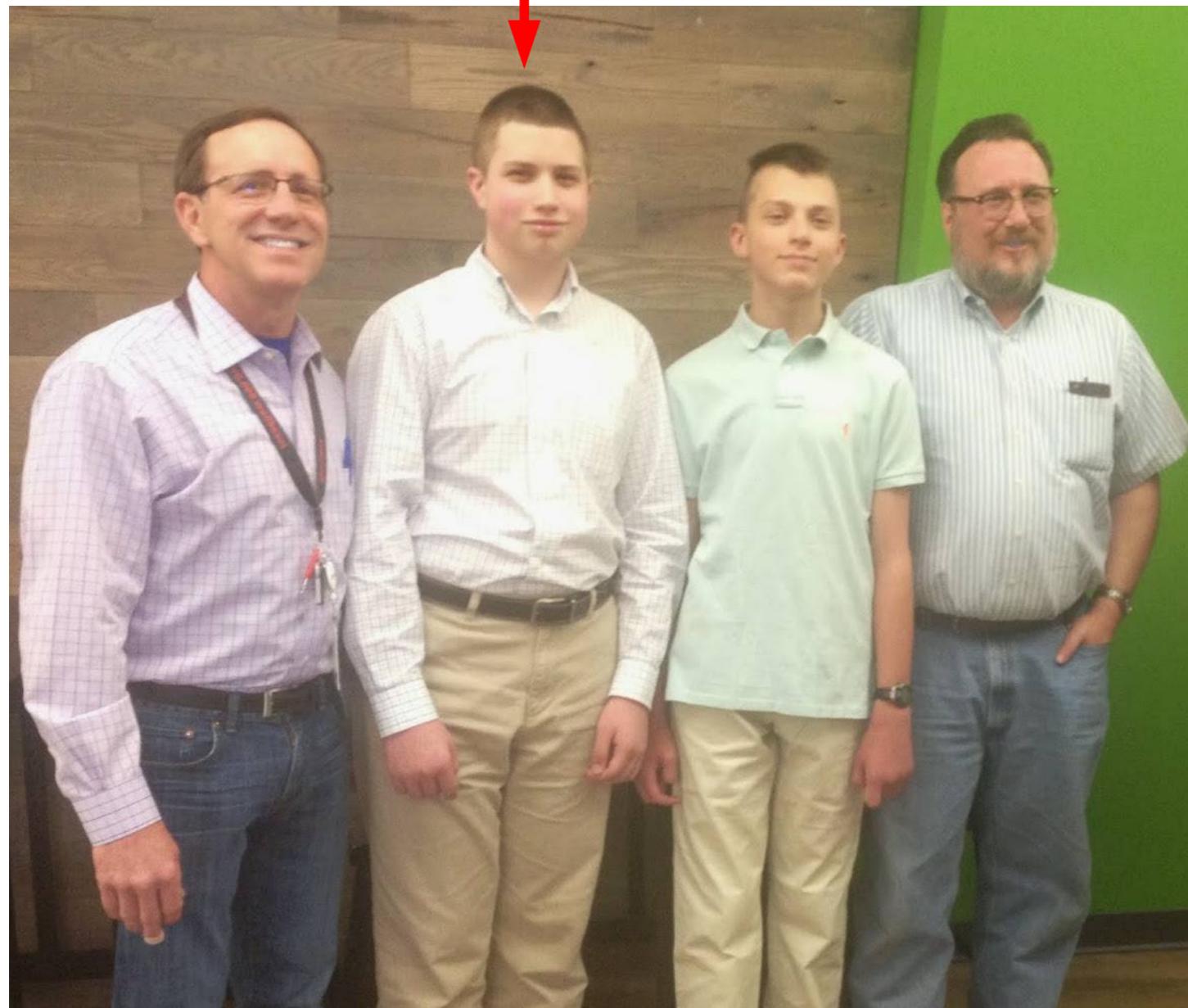
Revised By Thomas "Tweeks" Weeks



# About ~~Me~~ US



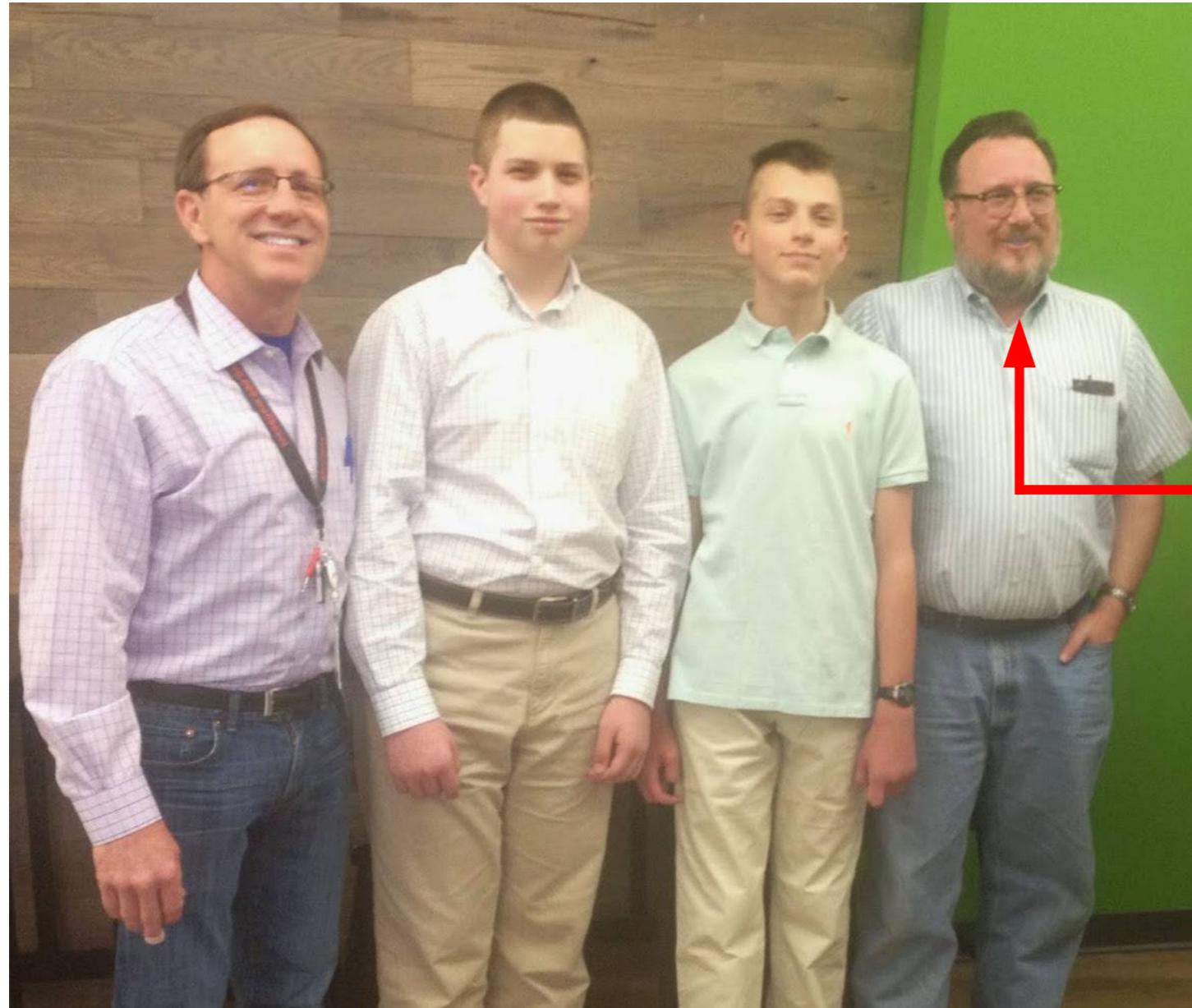
# About ~~Me~~ *us*



## Spencer Martin

- 14 (when he wrote this LCBB class)
- Youngest Intern ever at Rackspace
- Knows 6+ programming languages
- Loves coding
- <https://github.com/vtsman>

# About ~~Me~~ *us*



## Spencer Martin

- 14 (when he wrote this LCBB class)
- Youngest Intern ever at Rackspace
- Knows 6+ programming languages
- Loves coding
- <https://github.com/vtsman>

## Thomas "Tweeks" Weeks

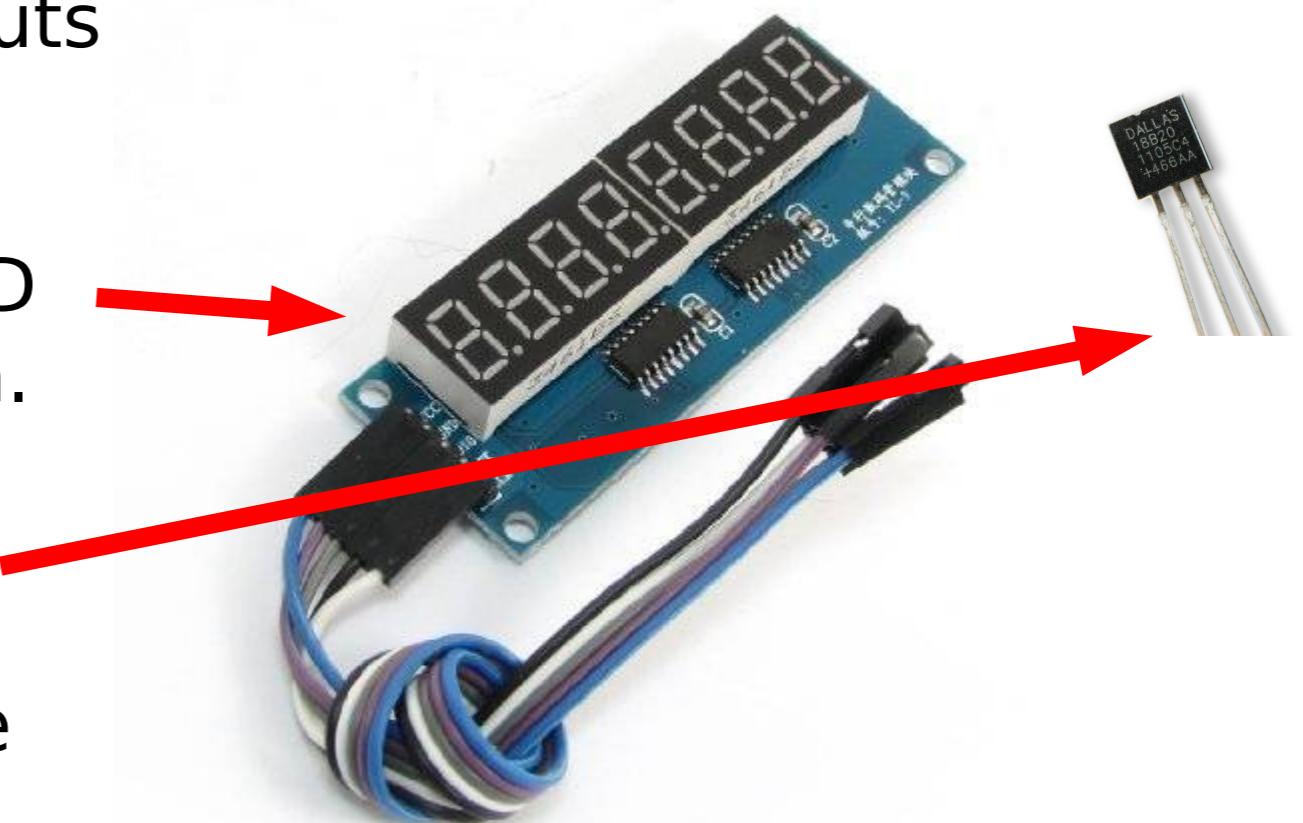
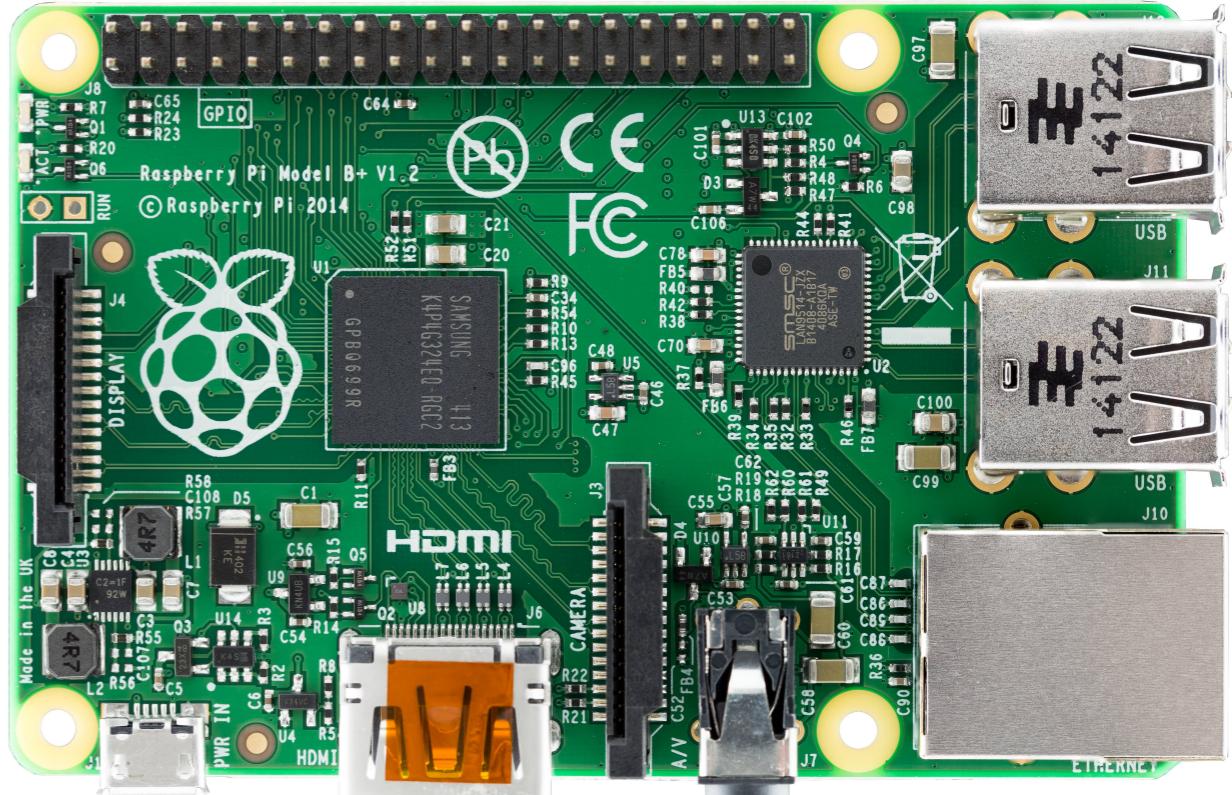
- Oldest (tenured Racker) of 6000+
- Co-authored two Linux books
- Loves kids, rockets, electronics & Linux
- Started HP Rocket club [www.nrvr.org](http://www.nrvr.org)

# Class goals

1) Learning a bit about the python language

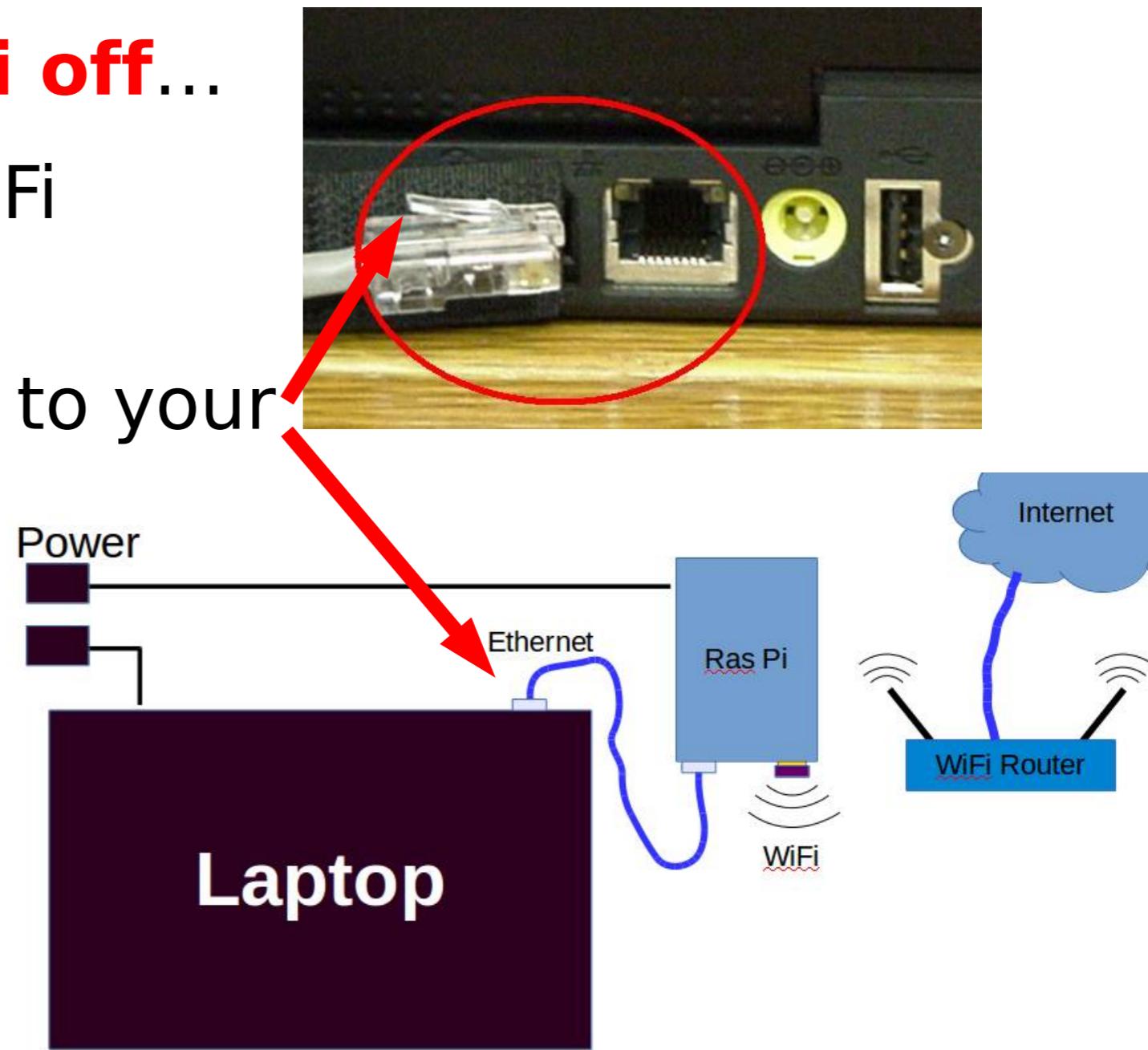
2) Write a final program called **piTempLogger.py** that:

- Measures things and outputs results to a blinking LED
- Output to a 7 segment LED display to show more data.
- Read temp from a sensor
- Research how to email the temperature to yourself in python



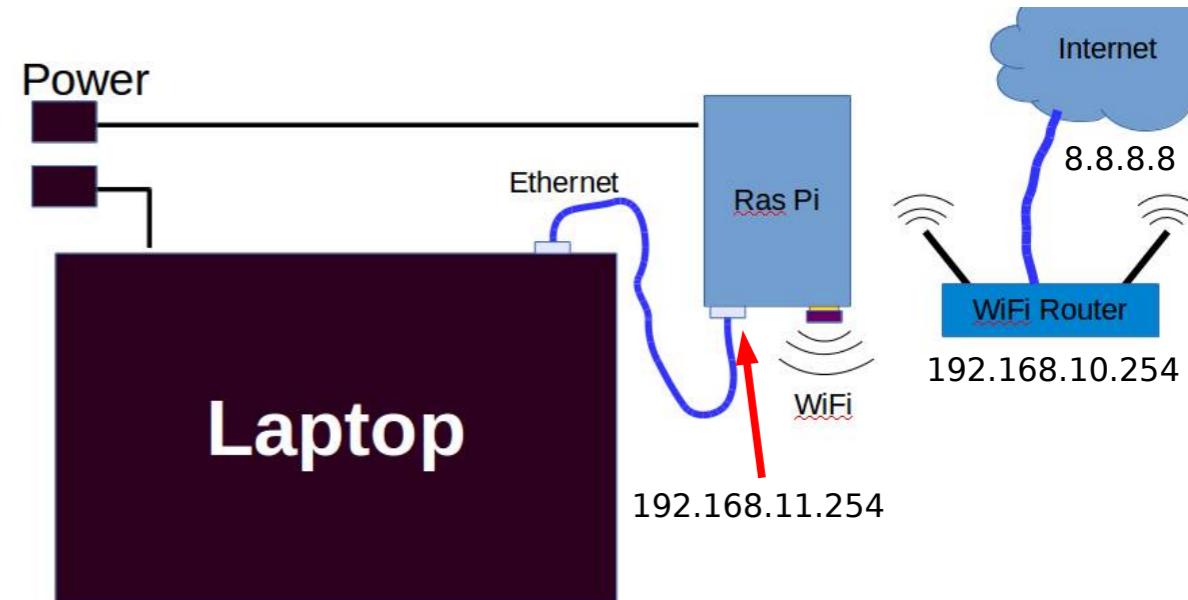
# Connecting Your Laptop To Your Raspberry Pi

- With **Laptop on** and **RPi off...**
- Turn off your laptop's WiFi
- Connect one end of the provided Ethernet cable to your laptop...
- Connect the other end to your RPi's Ethernet port...
- Power on the RPi
- Wait 60 seconds



# Failure: Testing Network Links

- To test your connection to the RPi and the RPi's connection to the Internet...
- **From the Laptop:**  
Open up a terminal or CMD window and the following pings will show you how far you're connected.



Check off which ones work for you:

- ping 8.8.8.8      Good connection from Laptop all the way to Internet
- ping 192.168.10.254      Good connection from Laptop through RPi to Router
- ping 192.168.11.254      Good connection to from Laptop to RPi

**TA/Instructor Sign-Off:**

# From Laptop, Connect to RPi's Remote Desktop

Connect to the Pi using “Remote Desktop” clients:  
(if prompted username/password = pi / letscode)

- For Windows “Run” window, type:

**mstsc /v:192.168.11.254**



- For Mac OSX install/use:

10.6 <http://tinyurl.com/mac-rdp-10-6>

10.7 <http://tinyurl.com/mac-rdp-10-7>

- For Linux type :

**rdesktop -u pi -g 1024x768 192.168.11.254**

**or**

**vinagre --geometry=1024x768 192.168.11.254**



**TA/Instructor Sign-Off:**

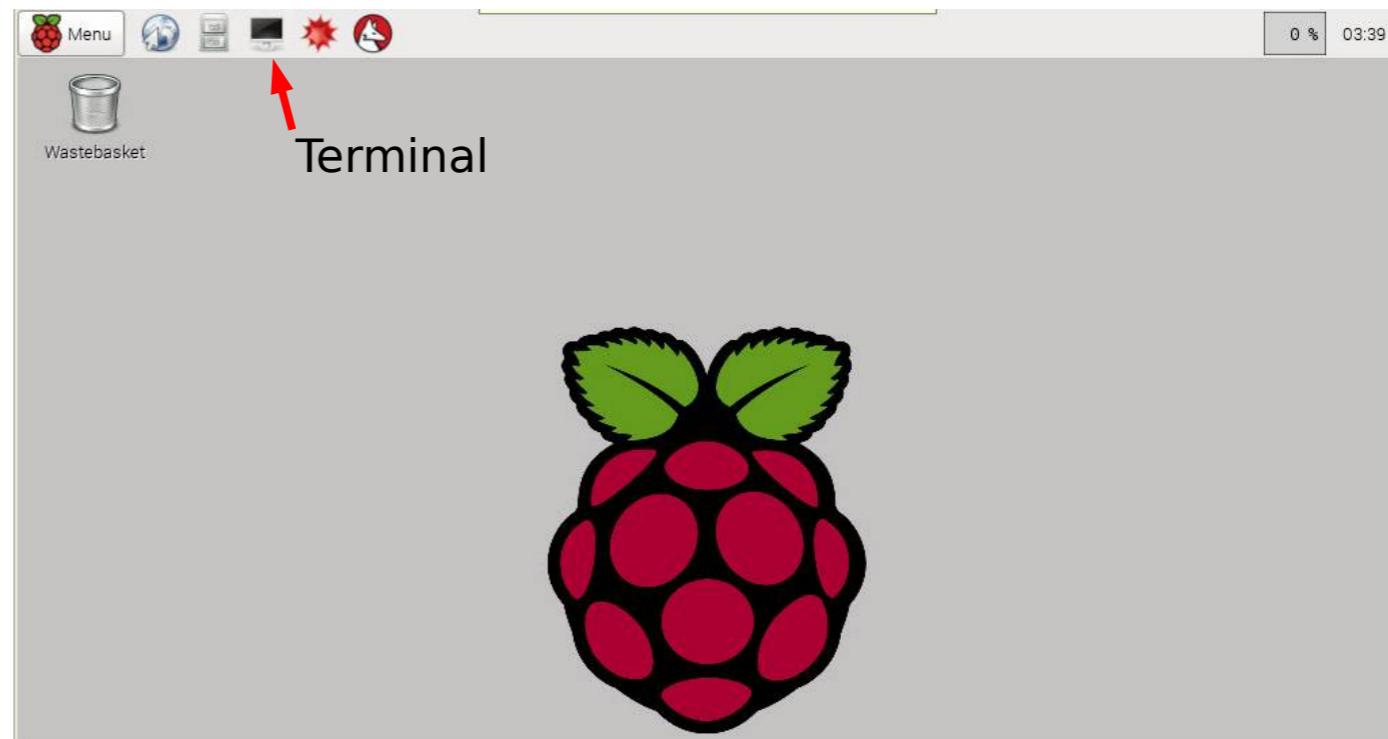
# From Laptop, Connect to RPi's Remote Desktop

If remote desktop/VNC client session is successful, you will see a screen like this:

## **Failure:**

If you are unable to pull up a remote desktop session with your remote desktop client:

- Check your previous ping tests
- Make sure you're going to 192.168.11.254
- Try a different remote protocol (either rdp or vnc)



If you can not get this step working, please call over a TA-helper.

**TA/Instructor Sign-Off:**



1) Getting Familiar with  
the Python Language

# Hello world!

Open terminal or command prompt and type in **python**

At the “>>>” python prompt, type in:

**print "hello world"**



```
>>> print "hello world!"  
hello world!  
>>> █
```



# Math and Variables

Try setting a variable by  
typing in:

`x = 1`

You multiply, divide, add,  
subtract using operators:

`*`      `/`      `+`      `-`

Print variables by omitting  
quotes

`print x`

```
>>> x = 1
>>> y = (x+2)*7
>>> print y
21
>>> print "y"
y
>>> 
```

# Math and Variables

Try setting a variable by typing in:

x = 1

You multiply, divide, add, subtract using operators:

\*        /        +        -

Print variables by omitting quotes

print x

```
>>> x = 1
>>> y = (x+2)*7
>>> print y
21
>>> print "y"
y
>>>
```

**WARNING: Standard mathematical “order of operations” applies. Remember PEMDAS ?**

# Input

The command **input()** is good for taking numbers as input from users and

**raw\_input()** is good for taking text (without needing quotes).

Either quietly prompts the user to enter data.

*Tip: To exit the python interpreter, hit CTRL-D*

Here's how you might use each:

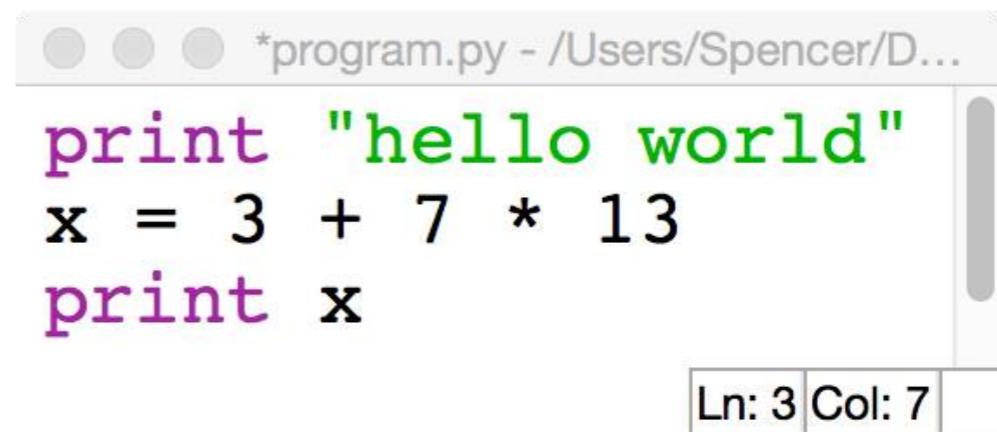
```
name = raw_input()  
age = input()
```

And what it would look like using them:

```
$python  
Python 2.7.3 (default, Mar 18 2014, 05:13:23)  
[GCC 4.6.3] on linux2  
Type "help", "copyright", "credits" or "license"  
>>> name=raw_input()  
Tweeks  
>>> age=input()  
32  
>>> print("%s is %s yrs old." %(name, age))  
Tweeks is 32 yrs old.  
>>>
```

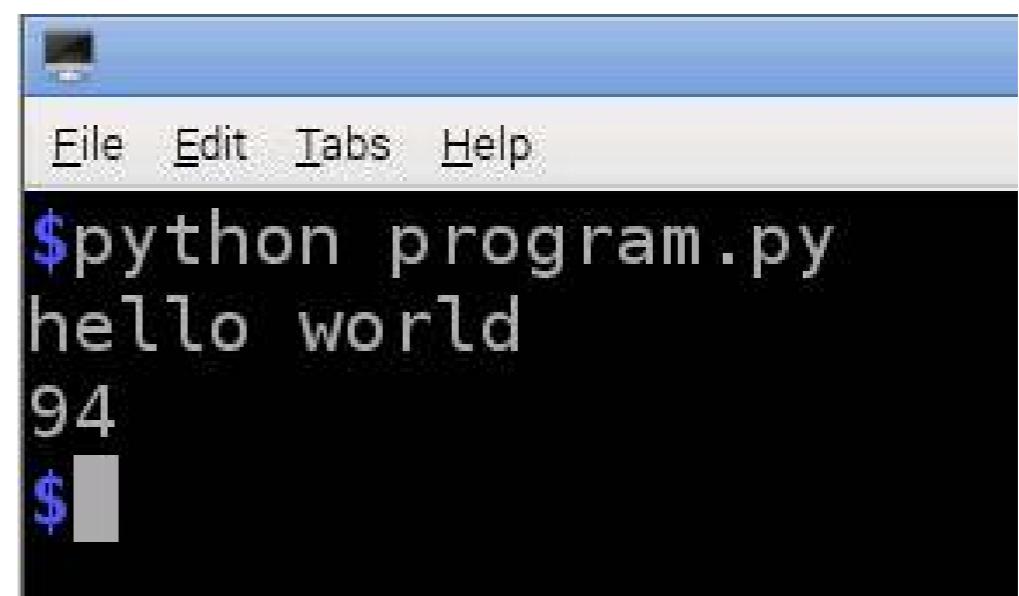
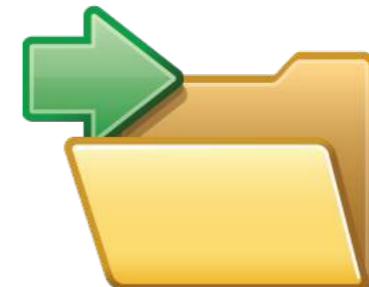
# Writing Multi line programs

- From your terminal, open the idle python interface by typing **idle.sh** and select File / New Window
- Type your python commands into the **idle** programming window
- Save-As your program to “program.py”
- Close all idle windows
- Run your program from the terminal, by typing in **python program.py**



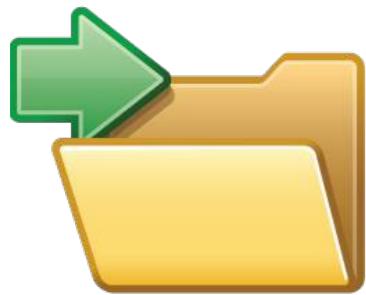
```
*program.py - /Users/Spencer/D...  
print "hello world"  
x = 3 + 7 * 13  
print x
```

Ln: 3 Col: 7



```
$python program.py  
hello world  
94  
$
```

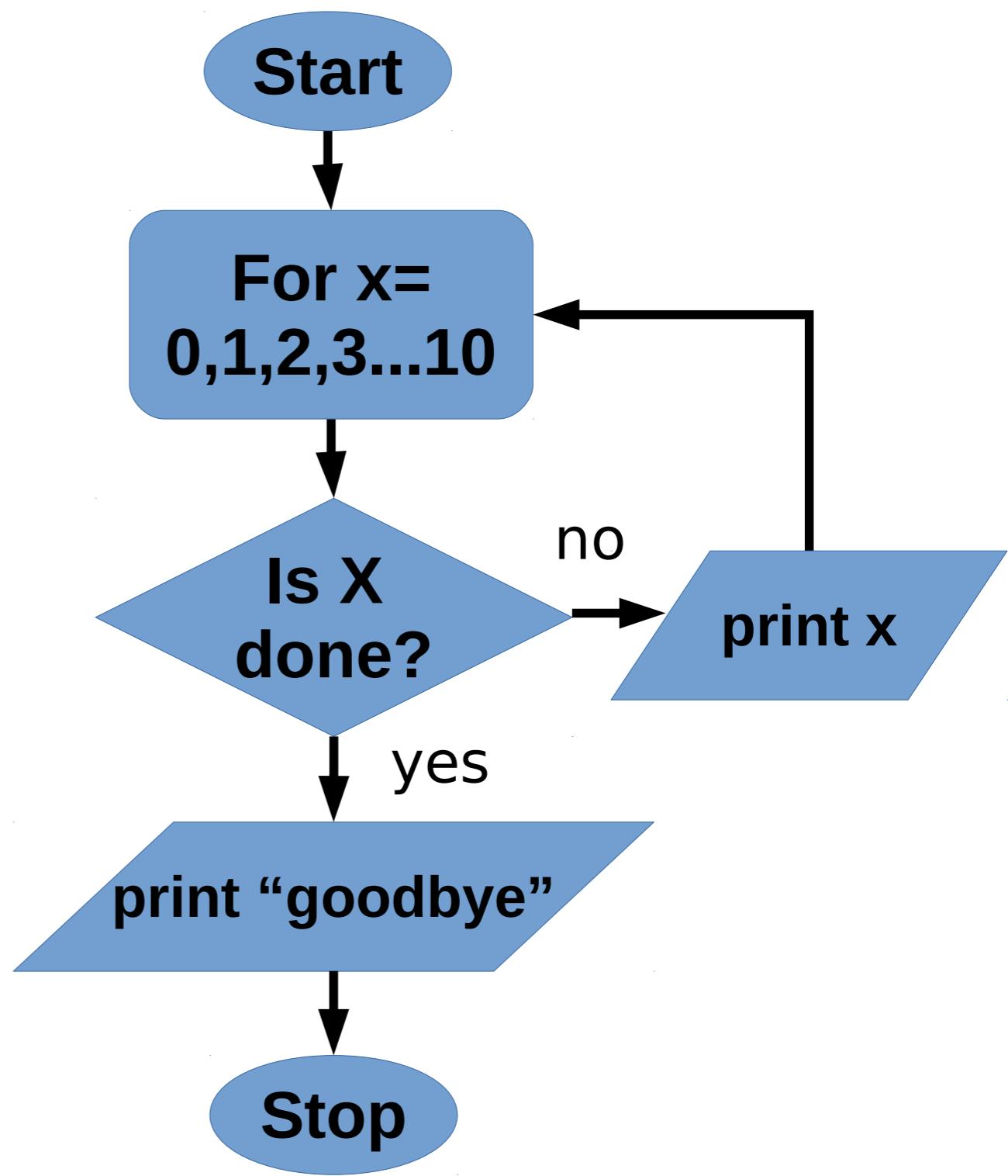
# For loops



*For loops* step through lists  
(in this case a range of numbers).

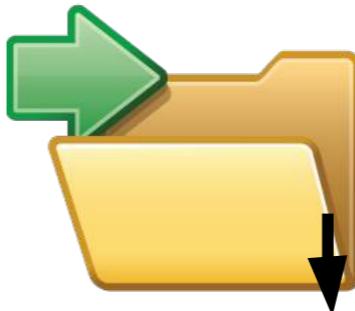
A *for loop* condition ends with a colon, and the code block within a for loop is all indented by spaces.

```
for x in range(0, 10):  
    print x  
print "goodbye"
```



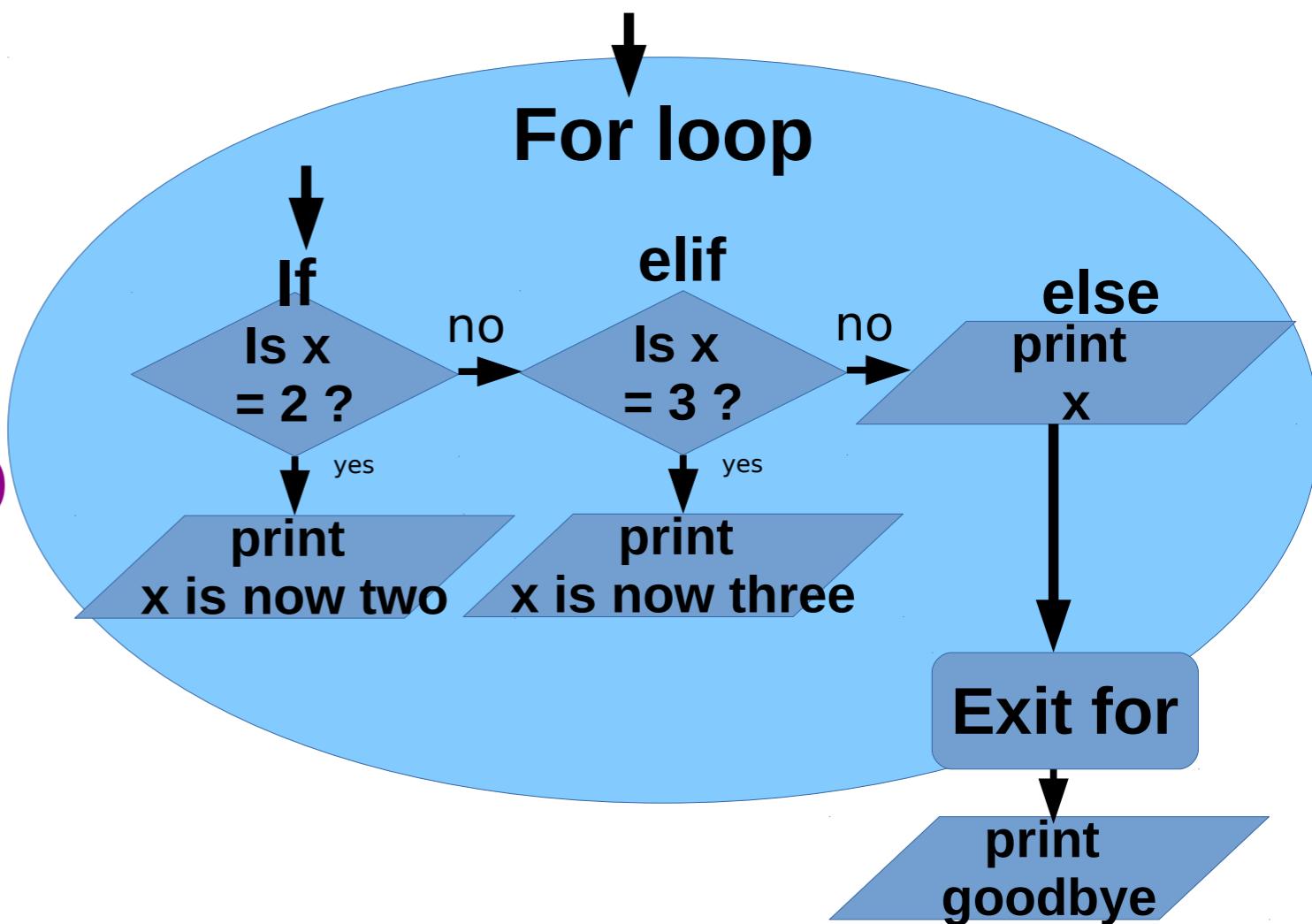
# Ifs

If checks a condition and executes code if the condition is true.



If conditions end with a colon, and the code block within an if is indented with spaces.

```
for x in range(0, 10):
    if x == 2:
        print("x is now two")
    elif x == 3:
        print("x is now three")
    else:
        print(x)
print "goodbye"
```

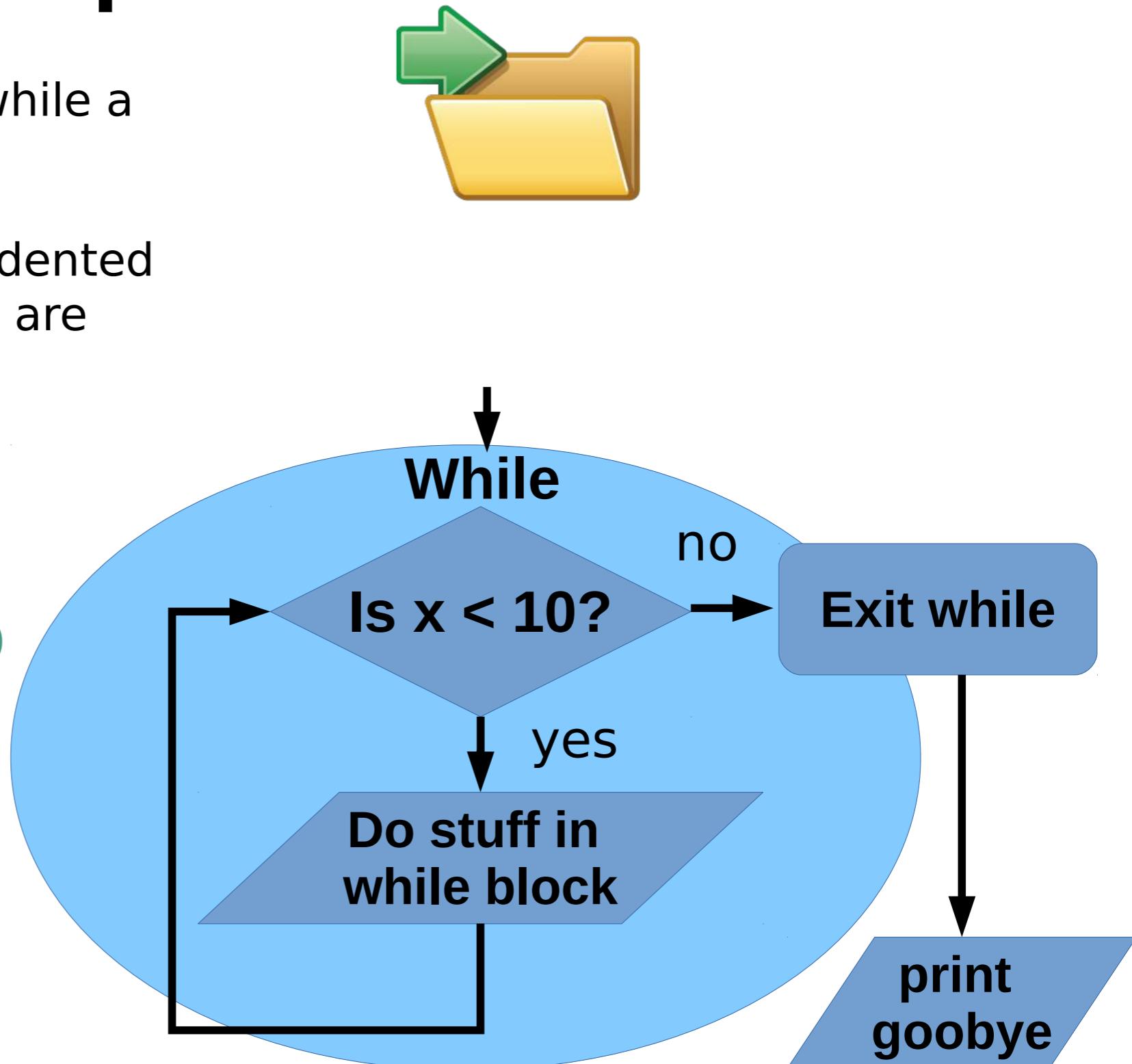


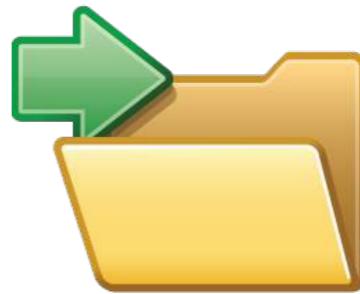
# While loops

While loops repeat code while a condition is true.

Code in while loops are indented by spaces, and conditions are ended with colons.

```
x = 0
while x < 10:
    if x == 2:
        print("Hello World")
    elif x == 3:
        print("Hallo Welt")
    else:
        print("Hola Mundo")
    x = x + 1
print "goodbye"
```





# Challenge

Write a 4 function calculator in python that takes as input, a number, then an operation (+ - \* / ), the another number and prints the answer.

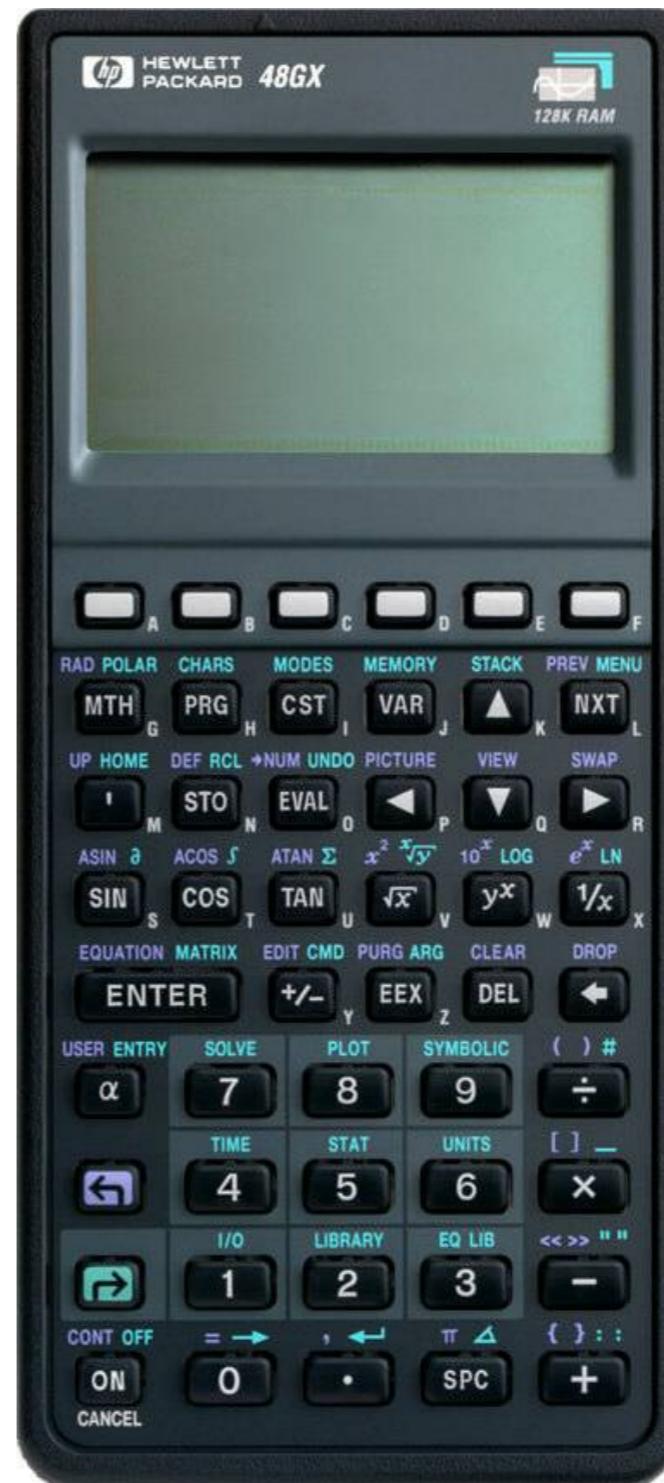
For example: If you input '12' <enter>, then '+<enter>, and finally '13' <enter>, then it should print 25 as the answer.

A screenshot of a terminal window. The window has a blue title bar with a monitor icon and the word 'File'. Below the title bar is a menu bar with 'File', 'Edit', 'Tabs', and 'Help'. The main area of the terminal is black with white text. It shows the command '\$python calc.py' followed by the input '12', the operator '+', the input '13', and the output '25.0'. A blue dollar sign '\$' is visible at the bottom left, indicating the prompt for the next command.

```
$python calc.py
12
+
13
25.0
$
```

# Calculator solution

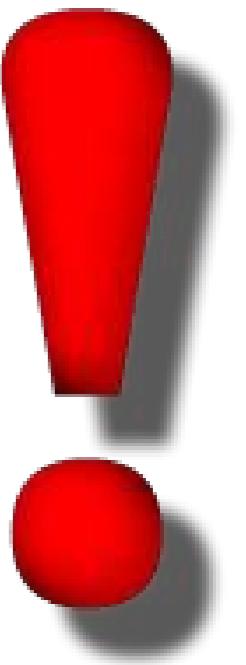
```
x = input()  
op = raw_input()  
y = input()  
  
if op == '+':  
    n = x + y  
elif op == '-':  
    n = x - y  
elif op == '*':  
    n = x * y  
elif op == '//':  
    n = x / y  
print(n)
```



## 2) Writing piTemoLogger.py

The goal of this class is to learn how to write a python **logger.py** program on the RaspberryPi we will eventually build up to a program called **piTempLogger.py** that will:

- measure the room temperature,
- displays it on a seven segment LED,
- If we finish everything else, email the data to ourselves.



## 2) Writing piTemoLogger.py

But to do all this, we must first learn how to:

- Talk to the outside world using the Pis GPIO port
- Output data to a 7 segment “595 driver” LED display
- Write nested loops and conditionals
- Read the temperature from a DS18B20 temp probe
- Send email to Gmail over SMTP using python

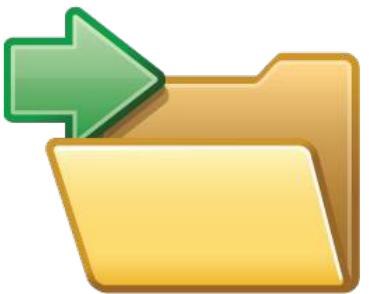
# 2) Writing piTemoLogger.py

But to do all this, we must first learn how to:

- Talk to the outside world using the Pis GPIO port
- Output data to a 7 segment “595 driver” LED display
- Write nested loops and conditionals
- Read the temperature from a DS18B20 temp probe
- Send email to Gmail over SMTP using python



**Let's get Started!!**



# GPIO Port Support

The RaspberryPi's GPIO, or *General Purpose Input/Output port*, is your Pi's gate to the outside world. It can be used to read values for sensors, or activate LEDs or motors.

It is used in python by importing (or loading) the GPIO module like this.

```
import RPi.GPIO as GPIO
```



**GPIO Port  
Hardware**

**GPIO Port  
Software**

# GPIO Pin Secret Decoder Ring

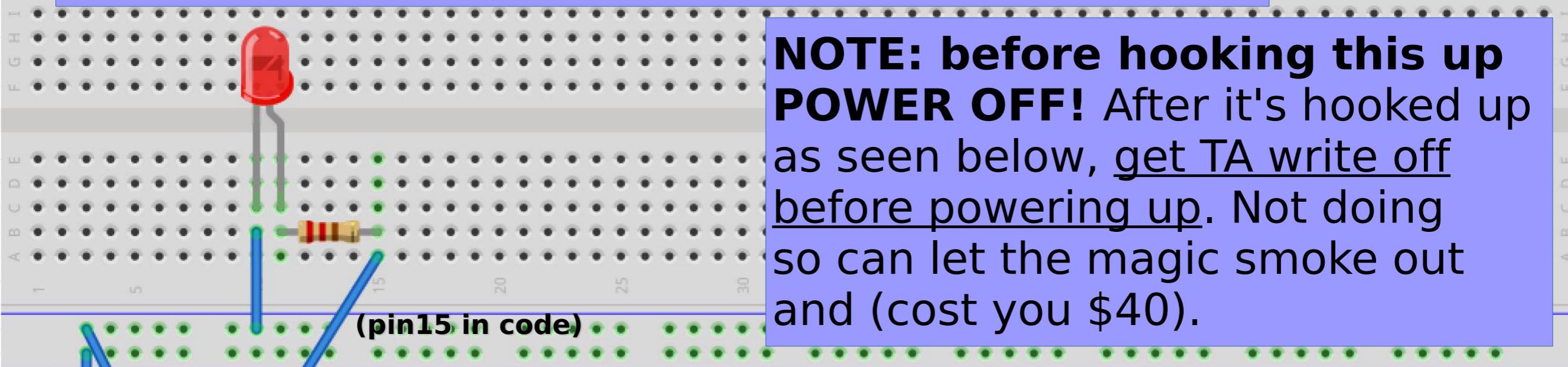
(keep this handy)



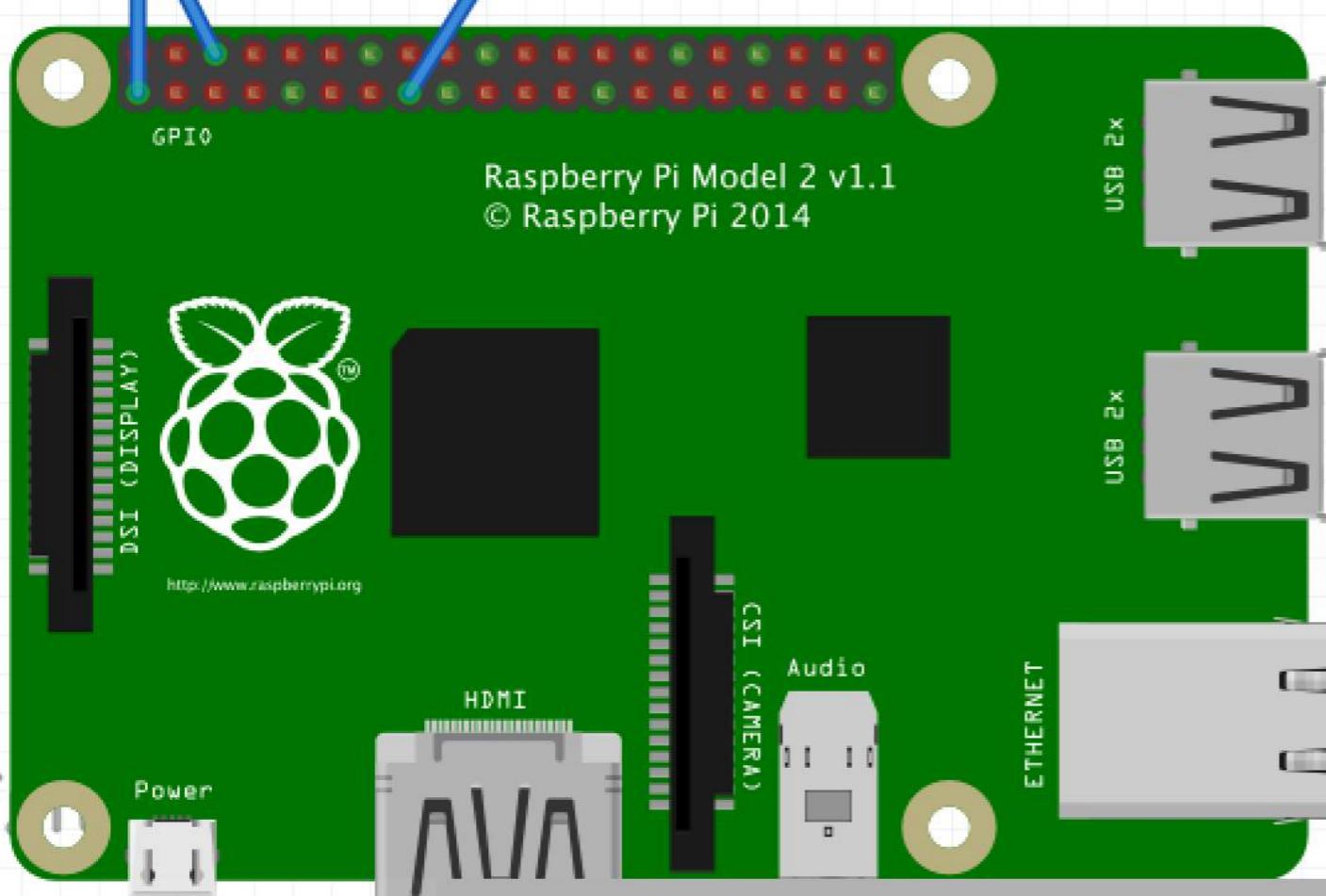
Raspberry Pi2 GPIO Header

Pin#	NAME	NAME	Pin#
01	3.3v DC Power	DC Power 5v	02
03	GPIO02 (SDA1 , I <sup>2</sup> C)	DC Power 5v	04
05	GPIO03 (SCL1 , I <sup>2</sup> C)	Ground	06
07	GPIO04 (GPIO_GCLK)	(TXD0) GPIO14	08
09	Ground	(RXD0) GPIO15	10
11	GPIO17 (GPIO_GEN0)	(GPIO_GEN1) GPIO18	12
13	GPIO27 (GPIO_GEN2)	Ground	14
15	GPIO22 (GPIO_GEN3)	(GPIO_GEN4) GPIO23	16
17	3.3v DC Power	(GPIO_GEN5) GPIO24	18
19	GPIO10 (SPI_MOSI)	Ground	20
21	GPIO09 (SPI_MISO)	(GPIO_GEN6) GPIO25	22
23	GPIO11 (SPI_CLK)	(SPI_CE0_N) GPIO08	24
25	Ground	(SPI_CE1_N) GPIO07	26
27	ID_SD (I <sup>2</sup> C ID EEPROM)	(I <sup>2</sup> C ID EEPROM) ID_SC	28
29	GPIO05	Ground	30
31	GPIO06	GPIO12	32
33	GPIO13	Ground	34
35	GPIO19	GPIO16	36
37	GPIO26	GPIO20	38
39	Ground	GPIO21	40

# Blinking an LED

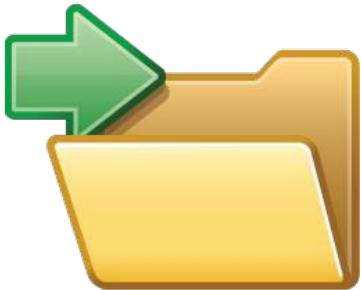


**NOTE: before hooking this up  
POWER OFF!** After it's hooked up as seen below, get TA write off before powering up. Not doing so can let the magic smoke out and (cost you \$40).



Here's how to hook up the Pi (GPIO22=pin15) to your breadboard and LED. Note that your kit may have a big rainbow ribbon cable and "T" shaped board connecting your Pi to the breadboard. Do not go beyond this section until you have it working.

**TA Initials**



# Blinking an LED

After importing the GPIO module (software) and connecting up the previous circuit (hardware), here's how we program the GPIO pinouts (pin 15 below = GPIO22 on the Pi's GPIO pins). After running this program, if your hardware is all correct, your program should slowly turn the LED on, wait, and turn it off. Get TA sign-off below.

```
#This imports the GPIO module
import RPi.GPIO as GPIO
#This imports time for sleep function
import time

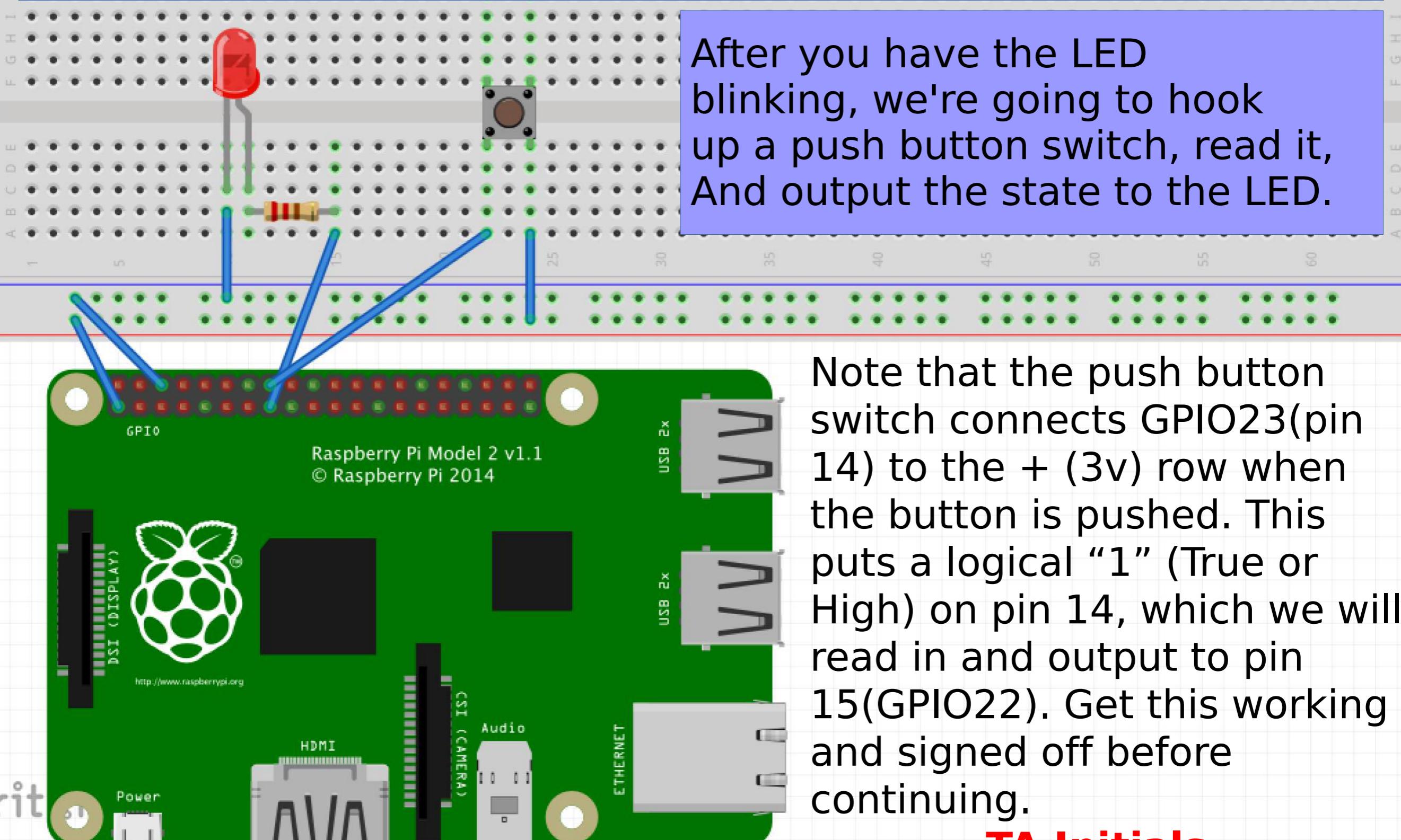
#initialize GPIO to use Raspberry Pi pinouts
GPIO.setmode(GPIO.BOARD)
#set pin 15 to output mode
GPIO.setup(15, GPIO.OUT)

while True:
    #Turn on LED and wait 1 second
    GPIO.output(15,True)
    time.sleep(1)
    #turn off LED and wait 1 second
    GPIO.output(15,False)
    time.sleep(1)
```

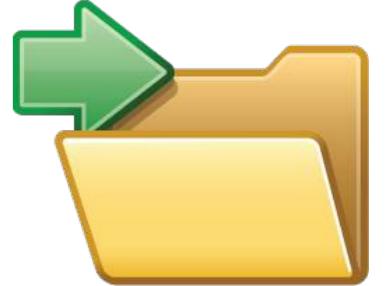


**TA Initials**

# Input from buttons



**TA Initials**



# Input from buttons

After connecting the push button switch to the Pi's GPIO23 (pin 14, in the code below), add the new lines of code from the program below. The button program will watch the button state, and if pressed (True), it will output that logic “True” state (on) to the LED output pin, turning on the LED. Be sure to get TA sign-off (below).

```
#get access to GPIO module
import RPi.GPIO as GPIO

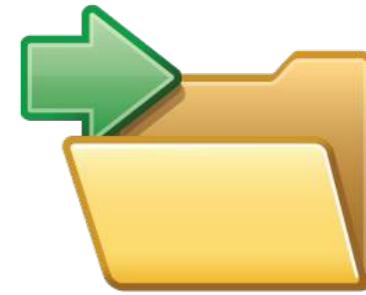
#set up pins
GPIO.setmode(GPIO.BOARD)
GPIO.setup(14, GPIO.IN)
GPIO.setup(15, GPIO.OUT)

#check whether button is pressed and
#change LED accordingly
while True:
    button = GPIO.input(14)
    GPIO.output(15,button)
```



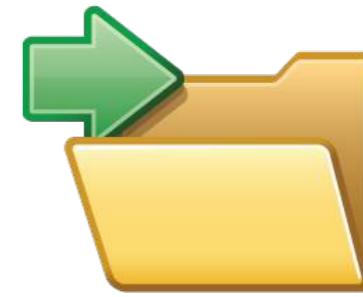
**TA Initials**

# Blinker Challenge



Using your logger.py code as a starting place (saving it to blinker.py) modifying the code so that the Pi blinks the LED **while** the button is released, and stops blinking when the button is pressed.

# Blinker Solution



```
#import GPIO module
import RPi.GPIO as GPIO

#import time for sleep function
import time

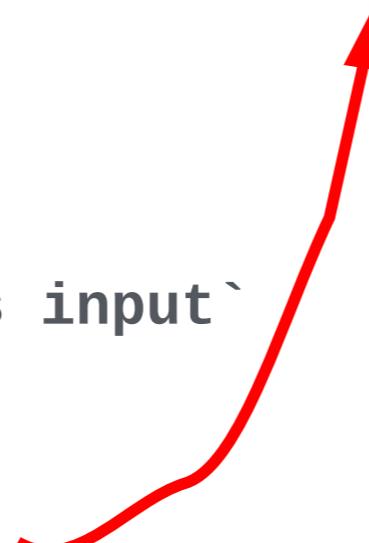
#initialize GPIO to use
#Raspberry Pi pinouts
GPIO.setmode(GPIO.BOTH)

#set pin 15 to output,14 as input
GPIO.setup(15, GPIO.OUT)
GPIO.setup(14, GPIO.IN)
```

```
while True:
    button = GPIO.input(14)
    while button == True:
        print 'waiting'

    #turn on LED and wait 1 second
    GPIO.output(15,True)
    time.sleep(1)

    #turn off LED and wait 1 second
    GPIO.output(15,False)
    time.sleep(1)
```



# Using 7-segment LED display modules

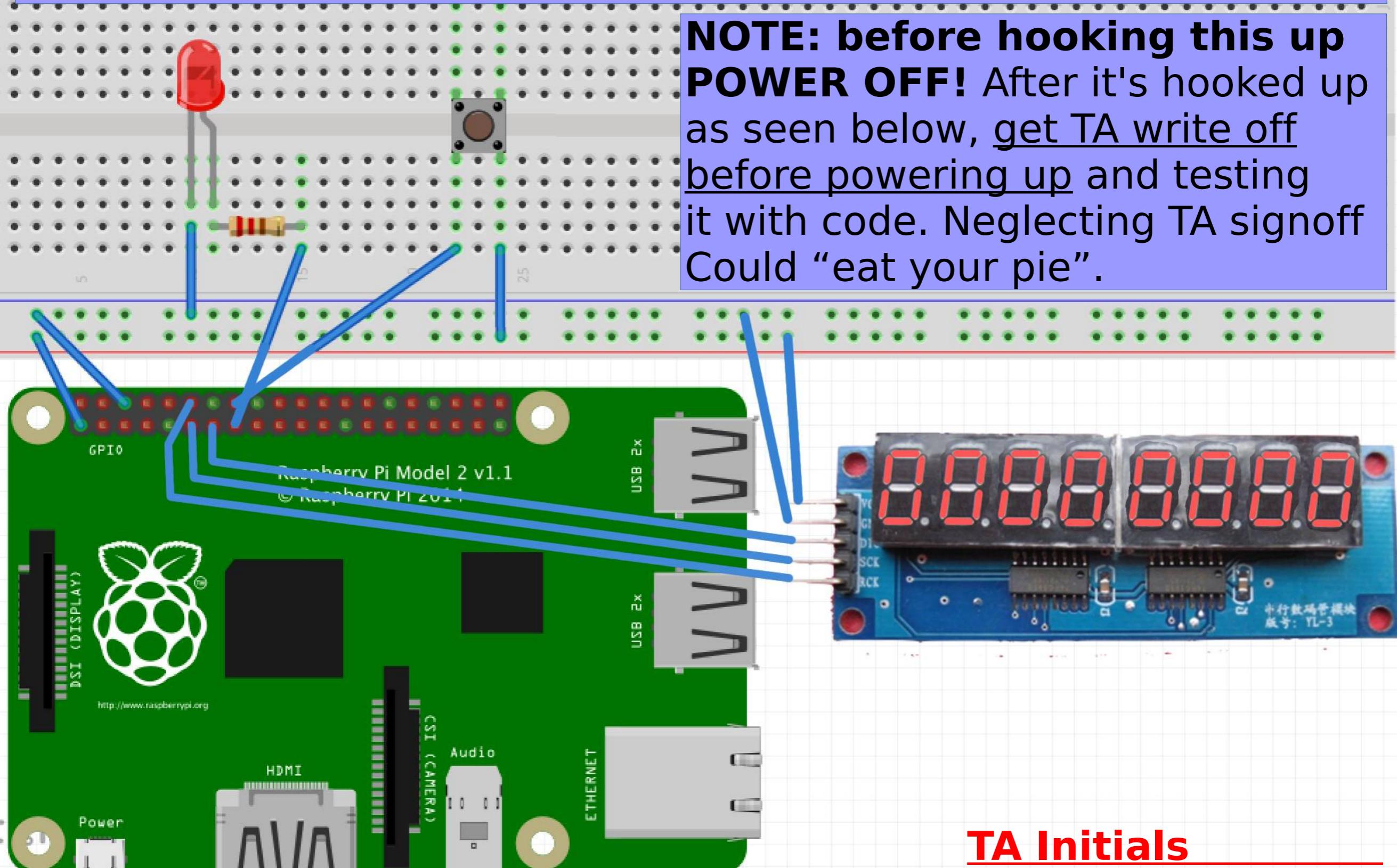
The seven-segment LED module “shifts” in the display number data into those two small “74LS595” TTL driver chips. As a result, this module only needs to be hooked up to ground, power (3.3v), a DATAinput, and two clock lines. If you hooked up eight 7-segment LEDs directly, it would take over 64 wire connections plus a bunch of electrical resistors!

If interested in how these chips work (or digital logic), then see the spec sheet for this common “TTL chips”\*.



\* - <http://www.ti.com/product/SN74LS595/technicaldocuments>

# Connecting 7-segment module

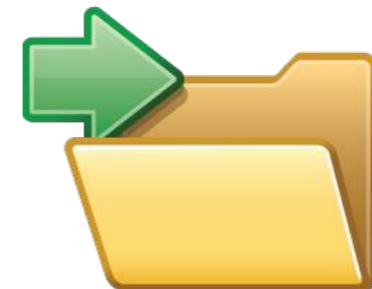


# Programming a 7-segment LED module



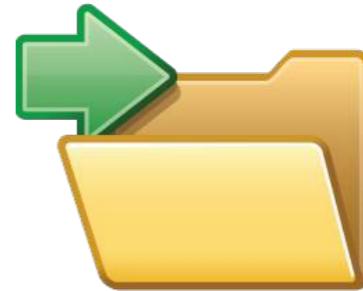
Create a new program called “test-led.py” that will use this new 7 segment LED display.

- 7 segment “595 driver” displays require a custom module called **PiSlice.py** (in LCBB-Pi dir)
- Initialize display by using **PiSlice.init()**
- Output to the display by using **PiSlice.number=n**

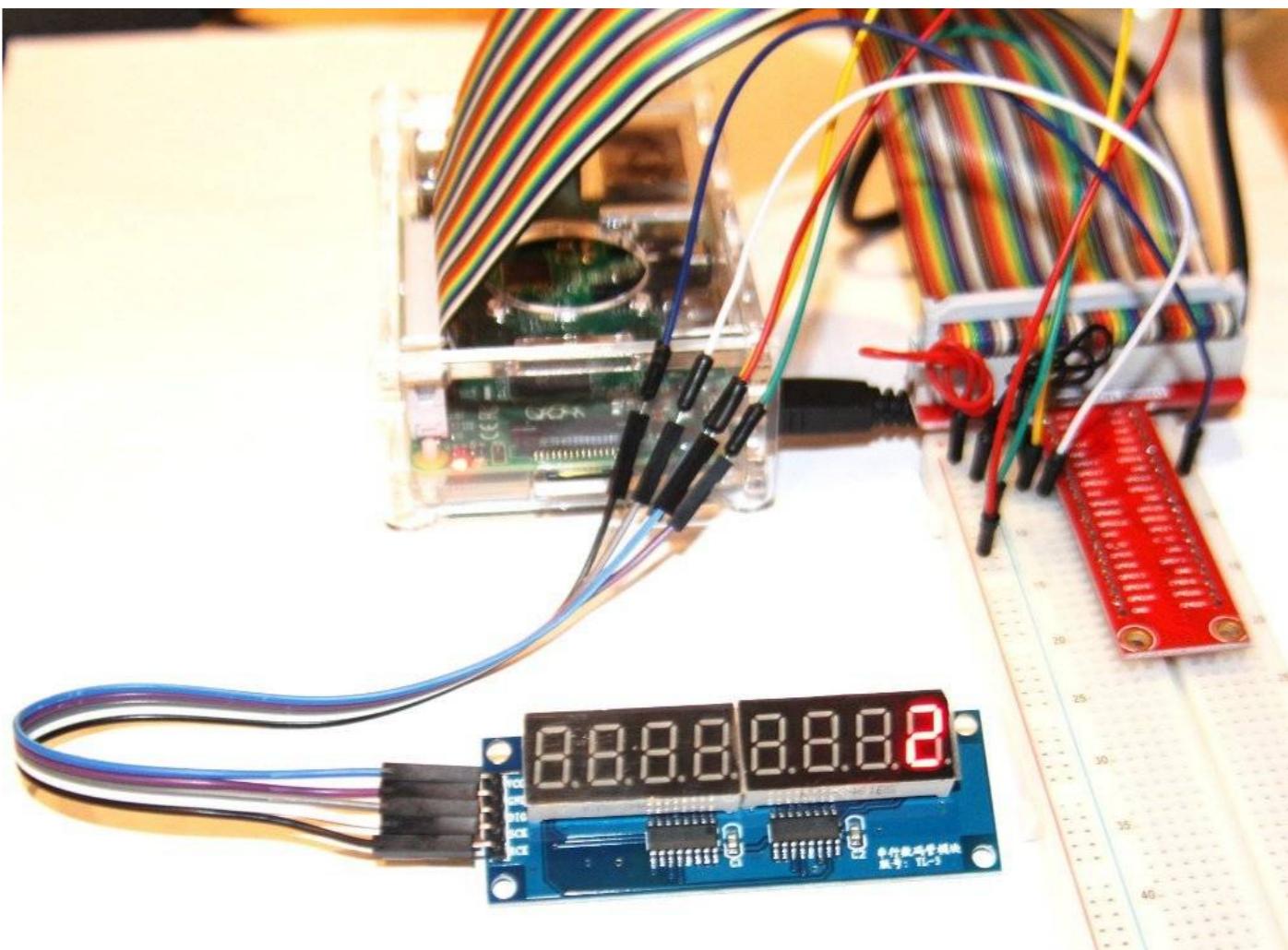


```
import PiSlice      #load module  
#DATAIN=13, CLOCK=11, LOCK=12  
PiSlice.init()    #initialize  
  
#This displays numbers  
PiSlice.number = 12345678
```

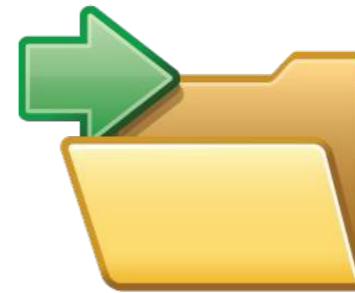
# Counter Challenge



Modify your logger.py program to use the 7-segment LED module, count your button presses, and add each press to the LED display (starting at zero).



# Counter Solution



```
#import GPIO module and PiSlice module
```

```
import RPi.GPIO as GPIO  
import PiSlice
```

```
#start up 7 segment display
```

```
PiSlice.init()
```

```
num = 0
```

```
while True:
```

```
    button = GPIO.input(14)
```

```
    while button == 0:
```

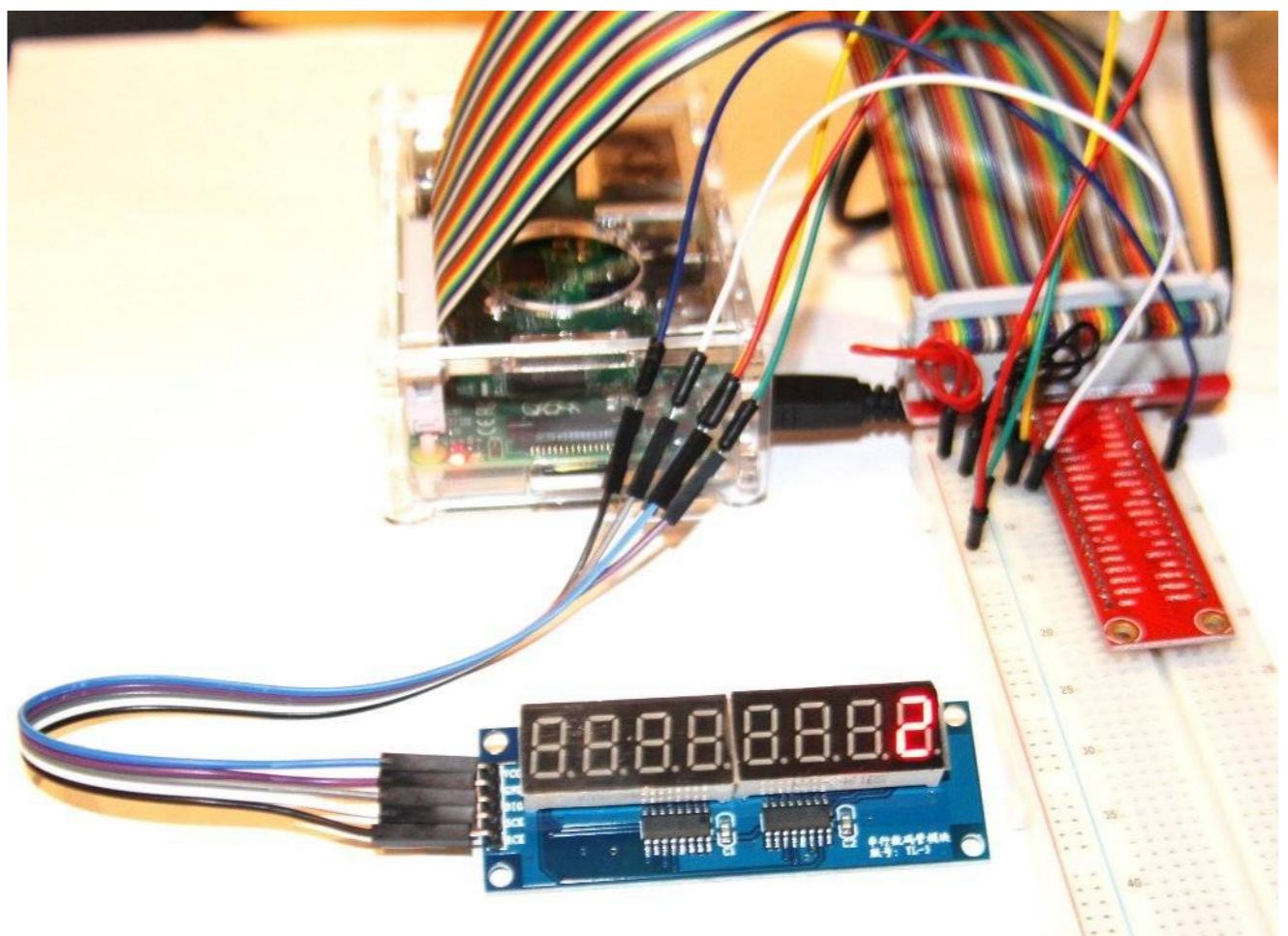
```
        button = GPIO.input(14)
```

```
    while button == 1:
```

```
        button = GPIO.input(14)
```

```
    num = num + 1
```

```
    PiSlice.number = num
```

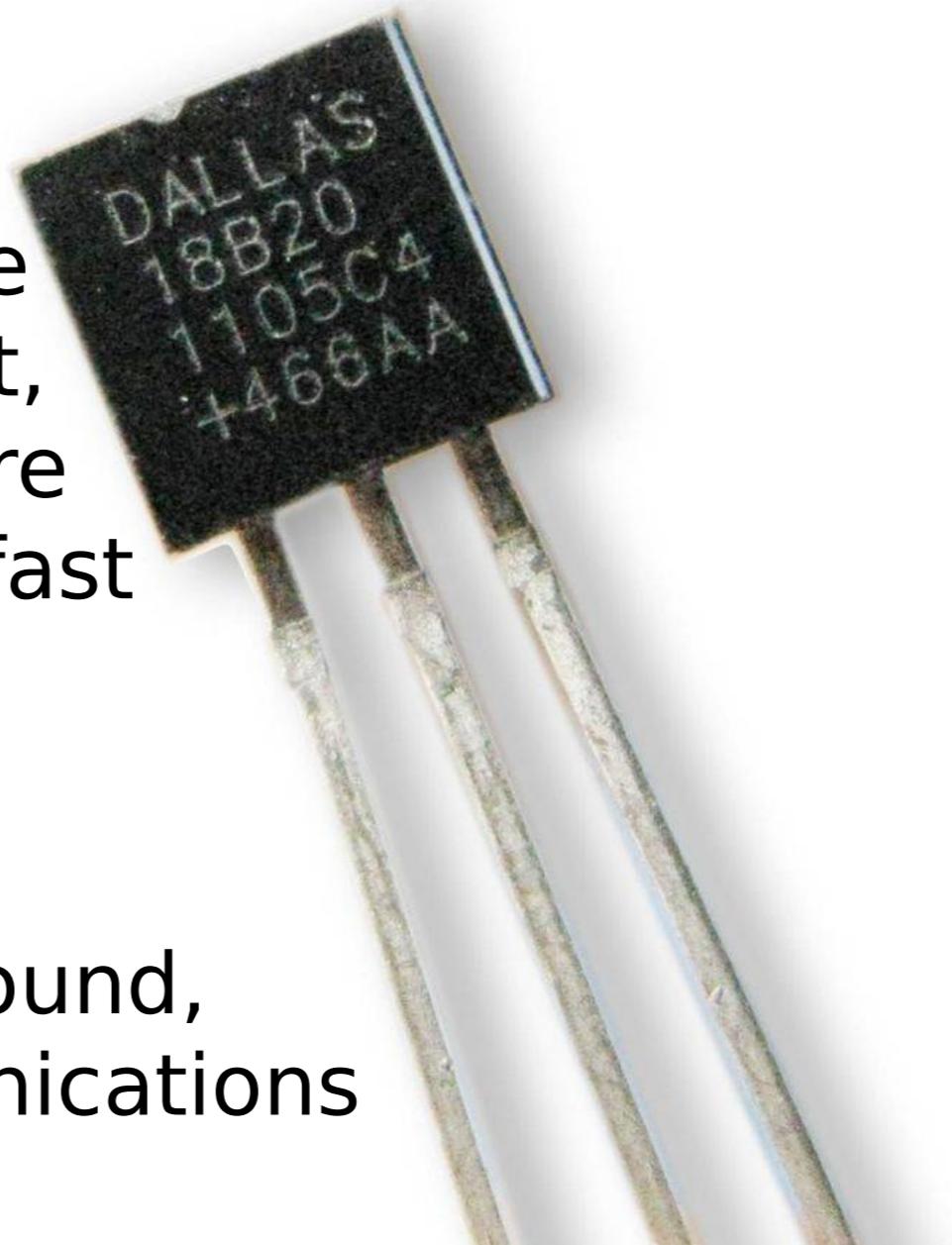


# Sensing temperature

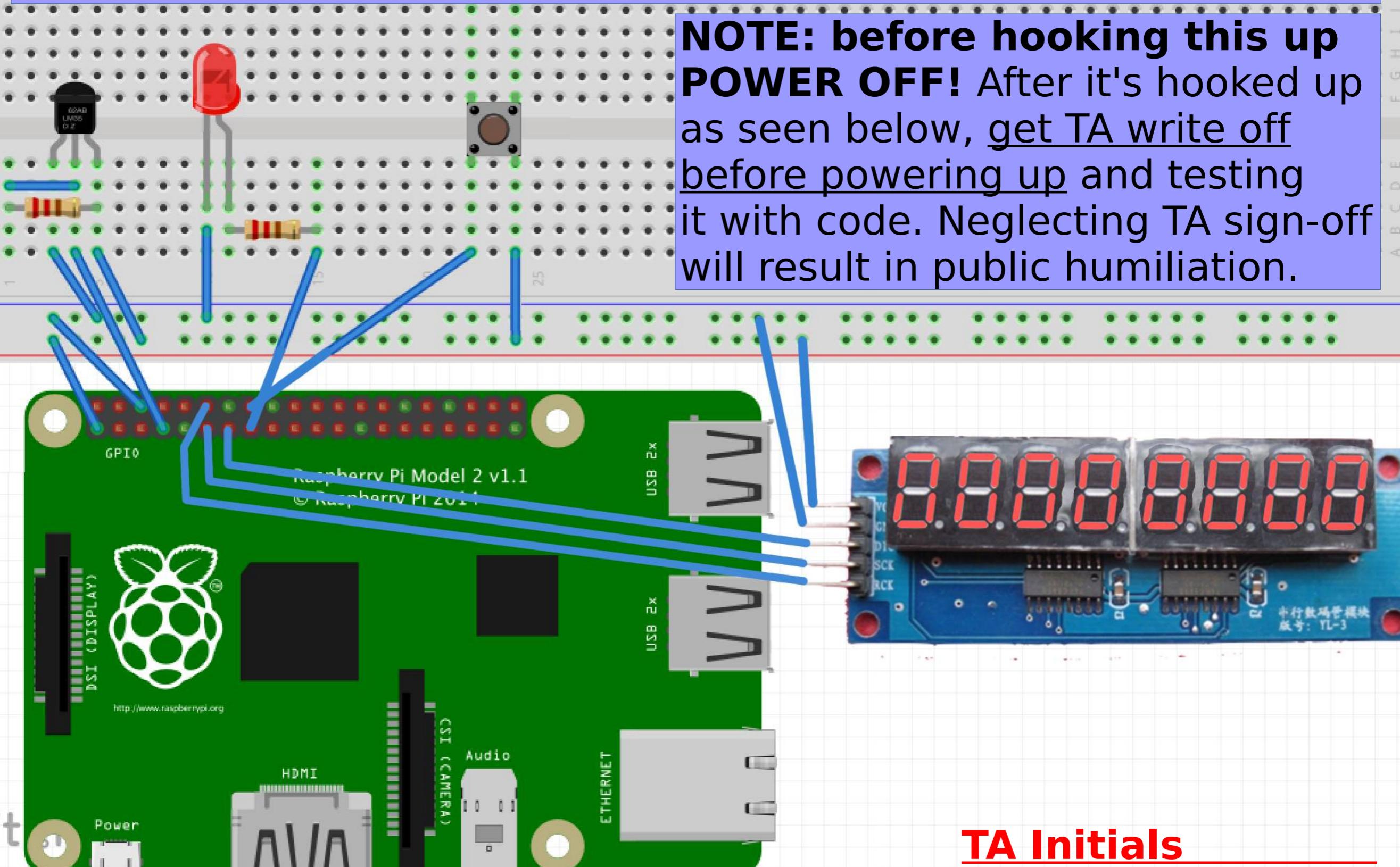
The 1-wire DS18B20 temperature sensor is a fully embedded, 12bit, serial (i2c) interfaced temperature probe that is perfect for simple, fast temperature data logging in many modern projects.

All you do it provide it power, ground, and “one wire” for serial communications

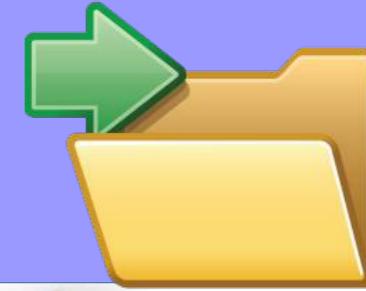
We're talking to it using our PiSlice.py python module. (next)



# Connecting 1-wire Temp Sensor



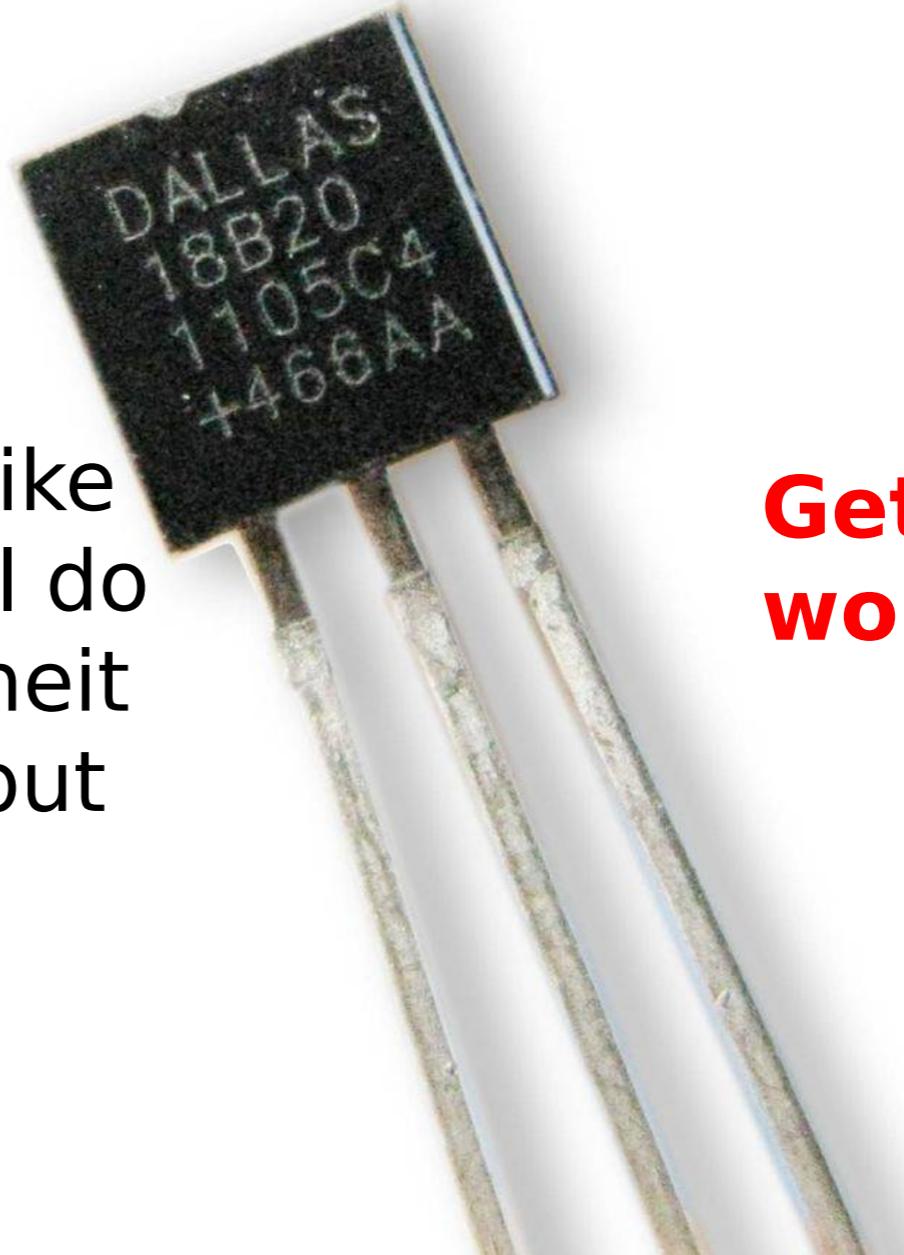
# Coding for the 1-wire Temp Sensor



Now to use the 1-wire temp sensor, we're just importing the PiSlice.py module, initializing it, and calling PiSlice.read\_temp() like this (below). The code below will do this and then output the fahrenheit value to the PiSlice.number output (LEDs).

```
import PiSlice

PiSlice.init()
PiSlice.init_temp()
while True:
    c, f = PiSlice.read_temp()
    PiSlice.number = f
```



**Get it  
working?**

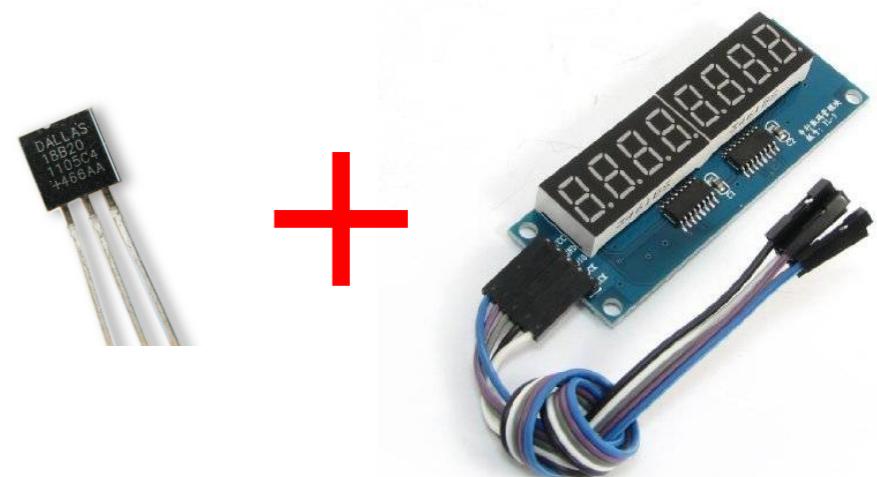
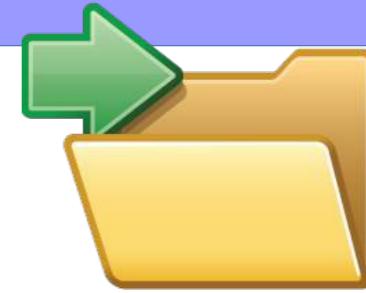


**TA Initials**

# Combine Temp Sensor and 7-segment LED

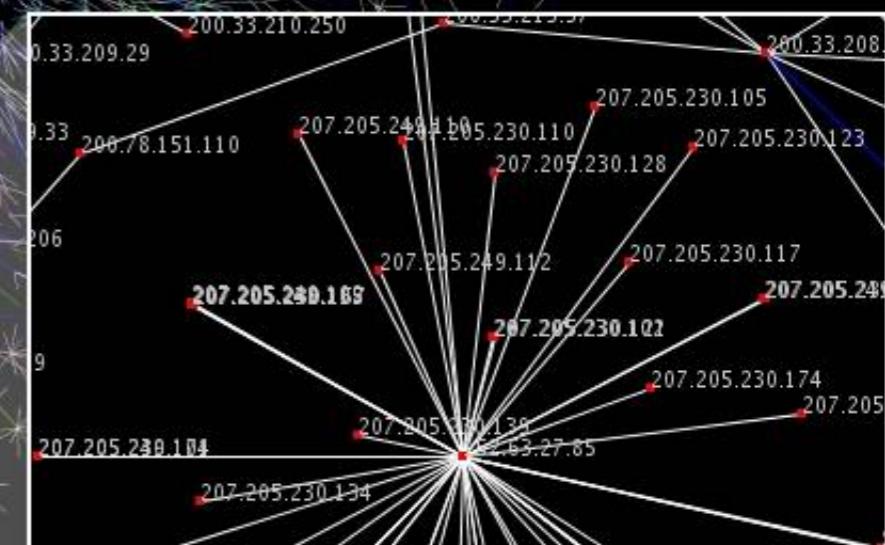
Now you have the temp sensor working and the previous LED code from logger.py. Now combine them to:

- Measure the temperature once per second
- Display the temperature on 7-segment LED display



# Pi in the sky

# Getting your Raspberry Pi project online

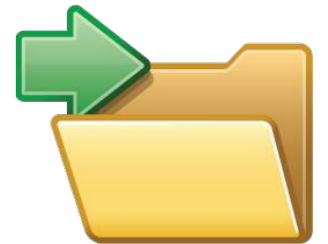


# Email Your Temp Data

- Python can send and receive emails using the SMTP protocol.
- You can send yourself data logged from the pi over email
- Your email address is [LetsCodePi@gmail.com](mailto:LetsCodePi@gmail.com)
- Ask instructor for password



# Sending emails



```
import smtplib  
fromaddr = 'LetsCodePi@gmail.com'  
toaddrs = 'to@addr.com'  
msg = 'testing'  
  
# Credentials (if needed)  
username = 'LetsCodePi@gmail.com'  
password = 'LetsCode'  
  
# The actual mail send  
server = smtplib.SMTP('smtp.gmail.com:587')  
server.ehlo()  
server.starttls()  
server.ehlo()  
server.login(username,password)  
server.sendmail(fromaddr, toaddrs, msg)  
server.quit()
```



# Final challenge



Make a program based off of the button counter that can show the number of emails you have and the current room's temperature. Cycle through these modes by pressing the button and display your output on the 7 segment displays.