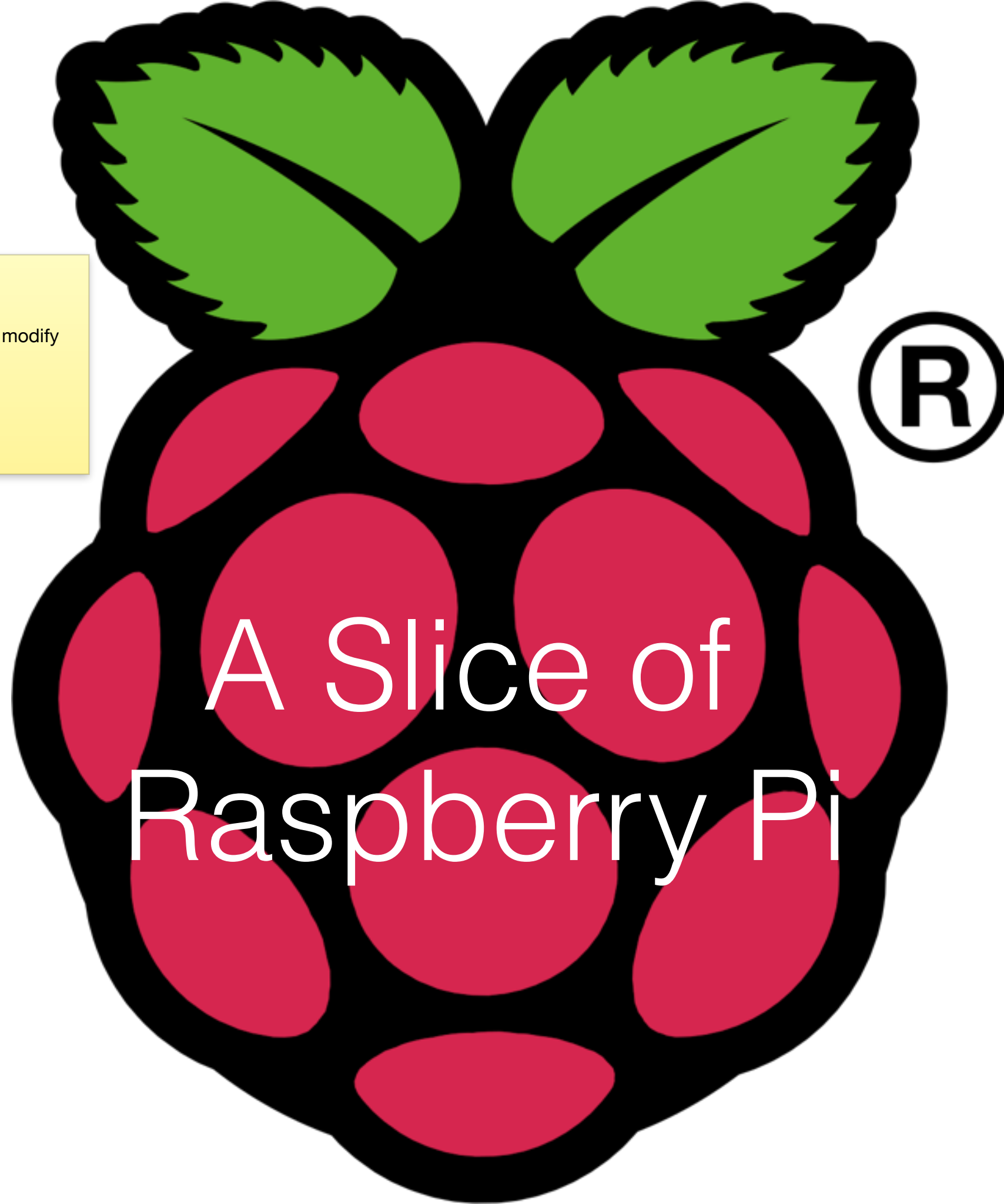


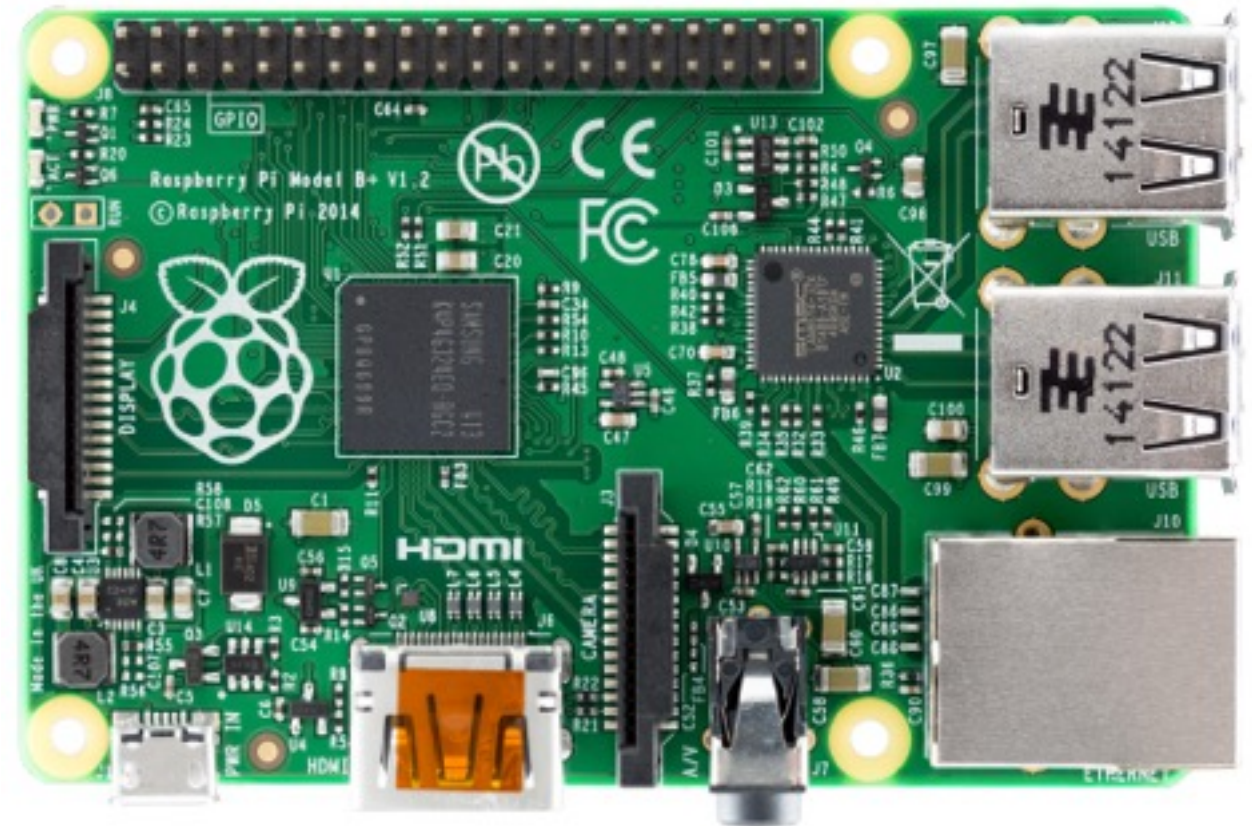
Main points:
Be sure to indicate which slides modify
your original program and



Class goals

Write a program called **piTempLogger.py** that

- Links services to an LED indicator
- Uses a 7 segment display to show more data.
- Reads temperature from a sensor
- Emails the temperature to yourself





Indicate project directory

Python

Put something down here

Hello world!

Open terminal or command prompt and type in “python”

In python, type in the following:

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

```
print "hello world"
```



Hello World!

Math and Variables

Set variable by typing in

```
x = 1
```

Multiply, divide, add, subtract

* / + -

Print variables by omitting
quotes

```
print x
```

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

Input

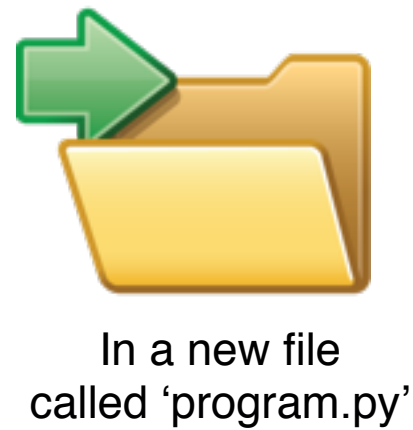
Simplify or remove need to cast string to int

Remove need for `raw_input()` and replace with better alternative?

Prompts the user to enter text and saves it to a variable

```
x = raw_input()
```

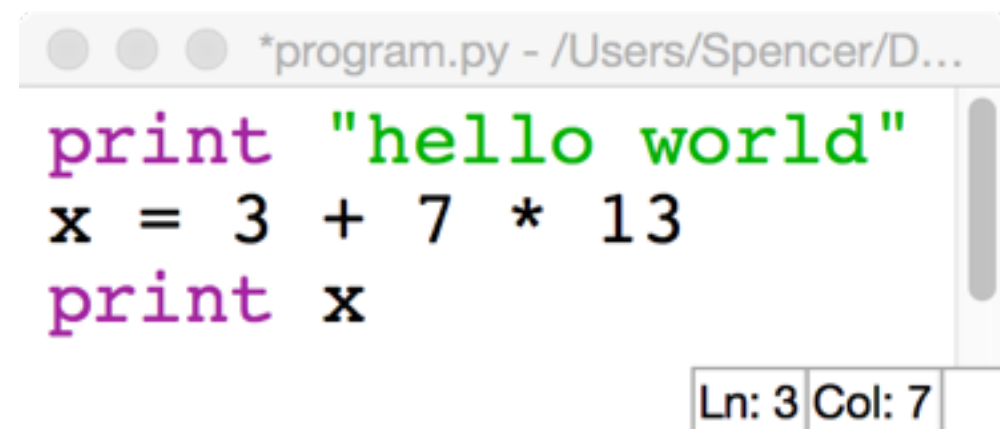
```
>>> x = raw_input()  
hello world!  
>>> print x  
hello world!  
>>> █
```



Multi line programs

- Type in all commands into a text file using idle
- Save said file as program.py
- Close out of the python REPL
- Open your program by typing in "python program.py"

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



```
print "hello world"  
x = 3 + 7 * 13  
print x
```

Ln: 3 Col: 7



In a new file
called 'logger.py'

For loops

Again and again

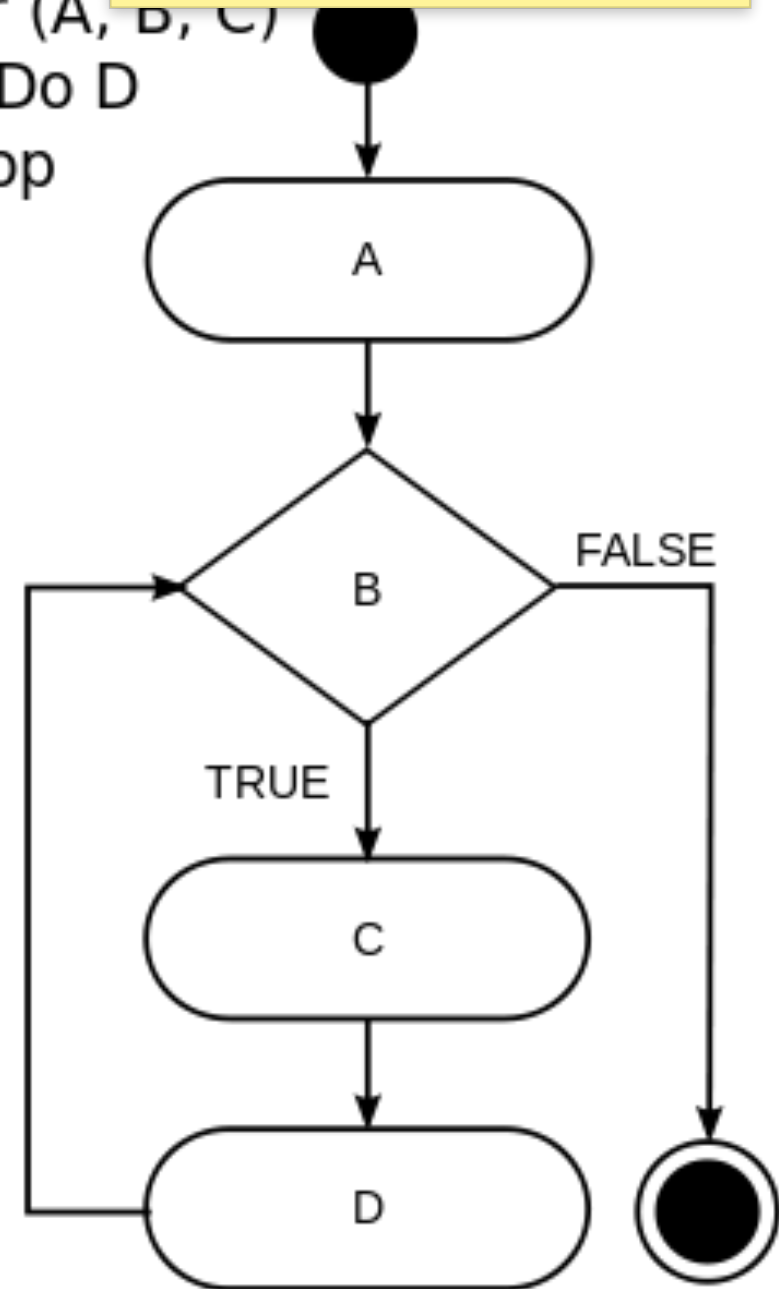
Demo range on its own

For loops count between 2
numbers.

Code in for's are indented by
spaces, and conditions are
ended with colons.

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

For (A, B, C)
Do D
Loop





In 'logger.py'

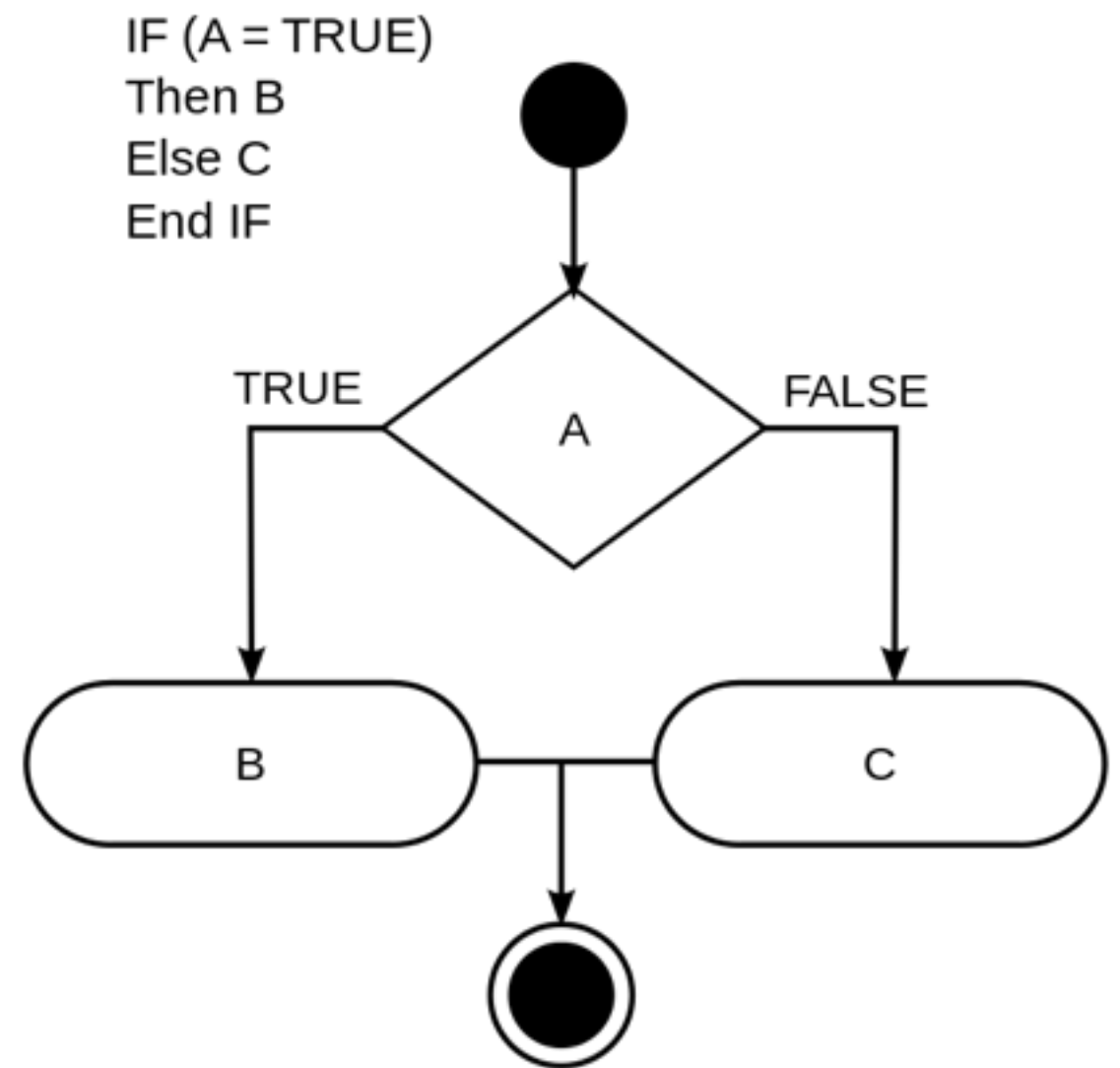
Ifs

If this, then that

Ifs execute different code depending on a condition.

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

```
for x in range(0, 10):  
    if x == 2:  
        print("Hello World")  
    elif x == 3:  
        print("Hallo Weld")  
    else:  
        print("Hola Mundo")  
print "goodbye"
```





In 'logger.py'

While loops

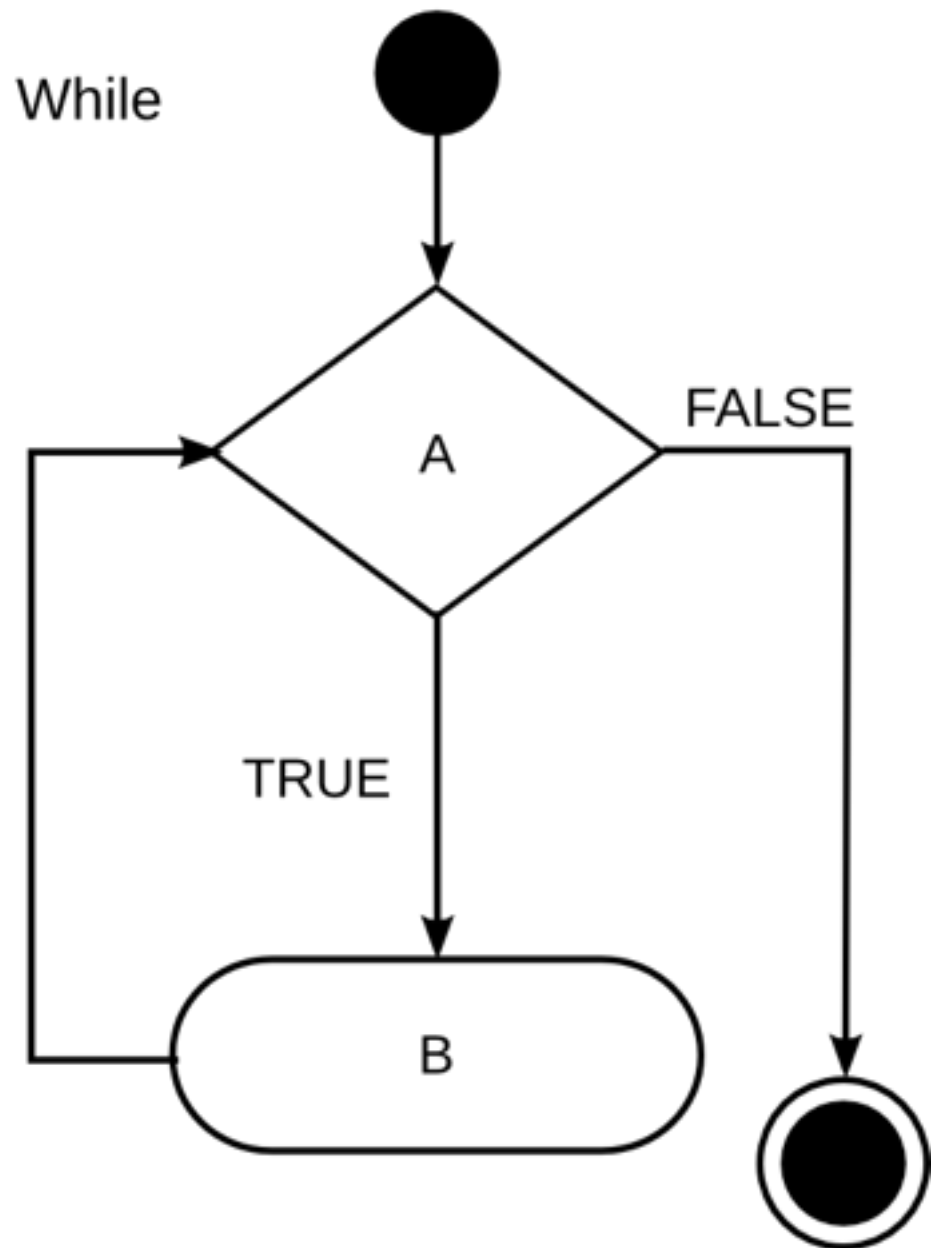
Déjà vu

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 Weld")
    else:
        print("Hola Mundo")
    x = x + 1
print "goodbye"
```

While (A = TRUE) Do
B
End While





In a new file
called 'calc.py'

Challenge

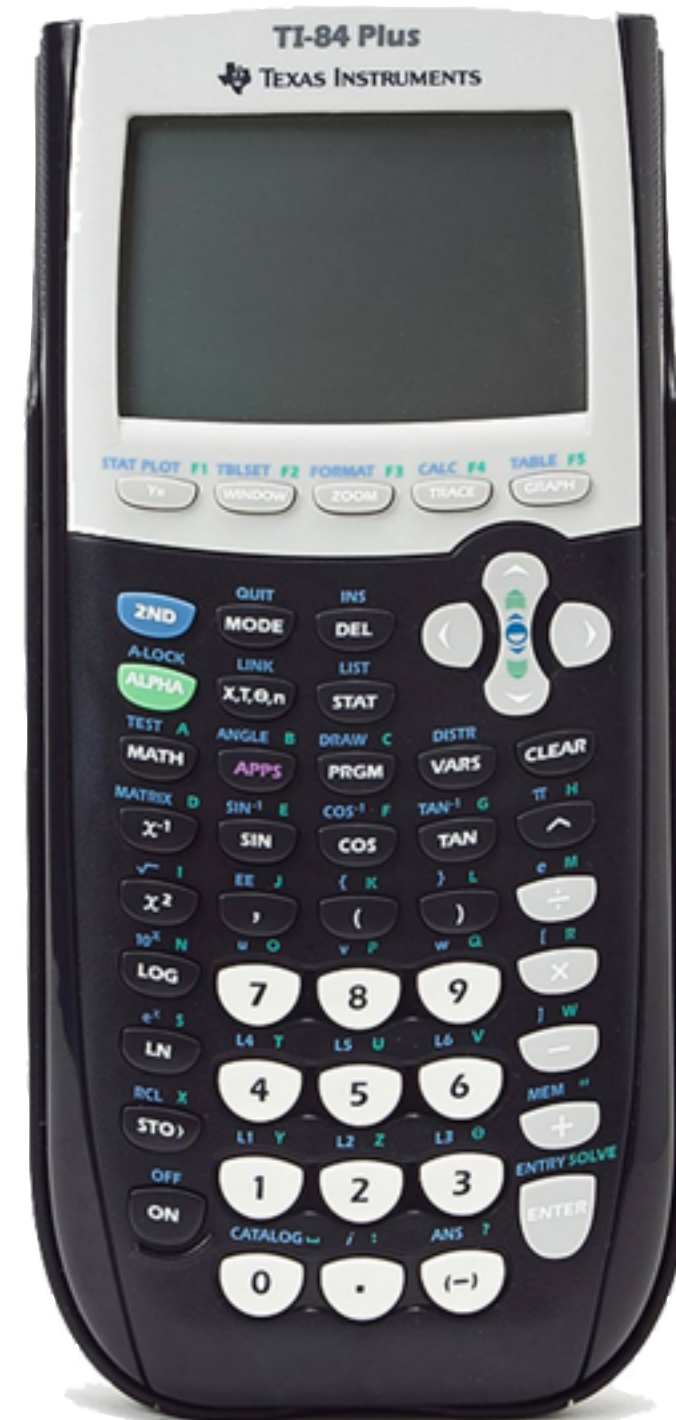
Make a 4 function calculator in python that takes in a number, then an operation (+-*/), and finally an other number. The input '12', then '+', and finally '13' should print 25.

```
>python calculator.py
12
+
13
25.0
>
```

Calculator solution

```
x = float(raw_input())  
op = raw_input()  
y = float(raw_input())
```

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

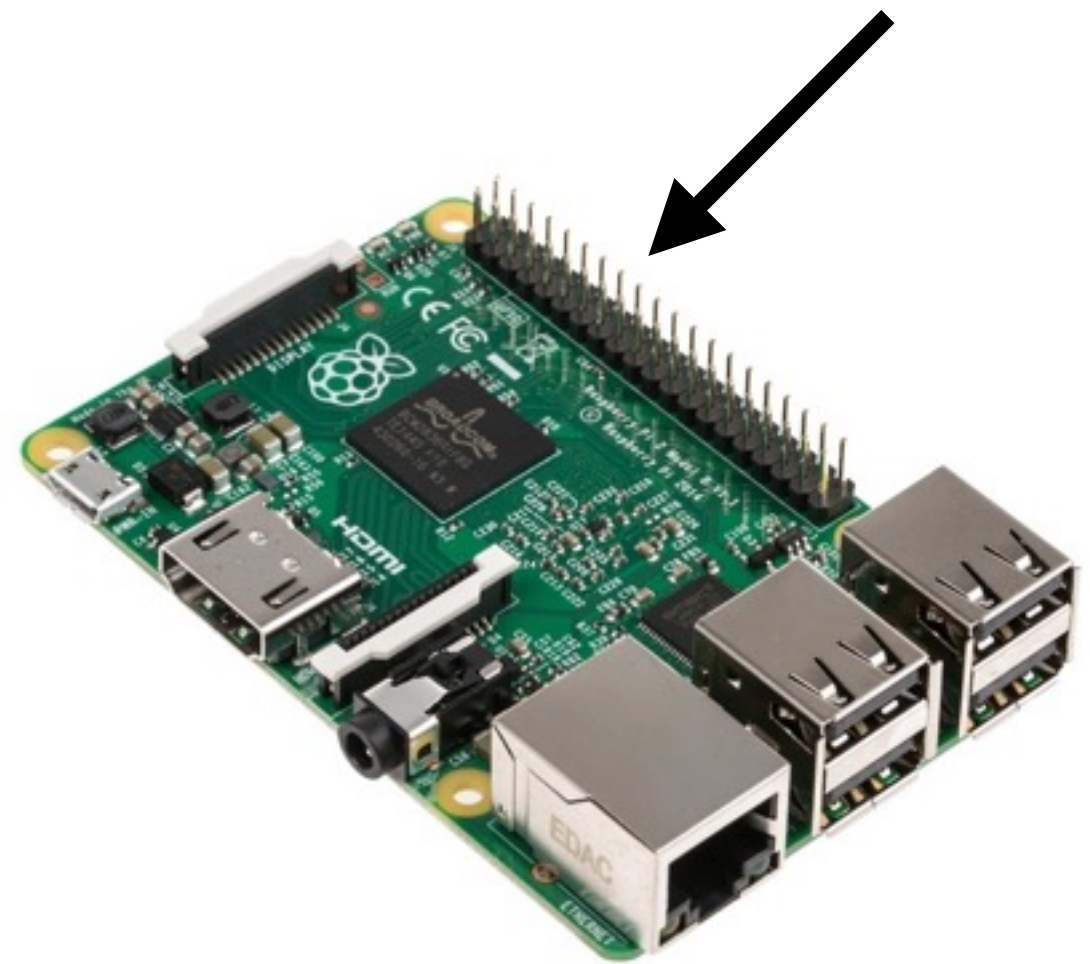


Update pi graphic to 40 pins
Mention what module is
Explain syntax of import

GPIO

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 can be used in python by using the GPIO module.

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





Blinking an LED

Wiring diagram

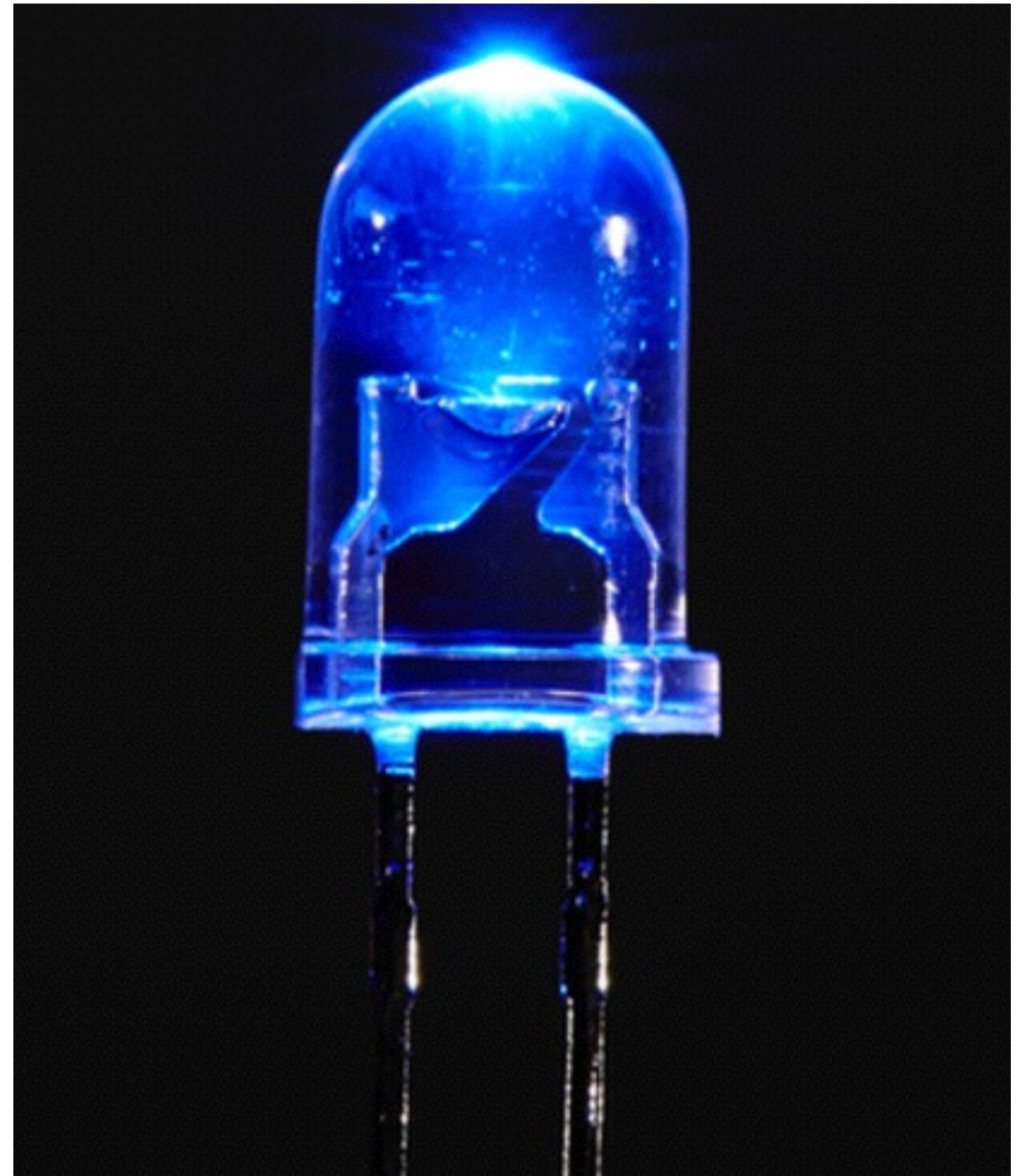
Use fritzing

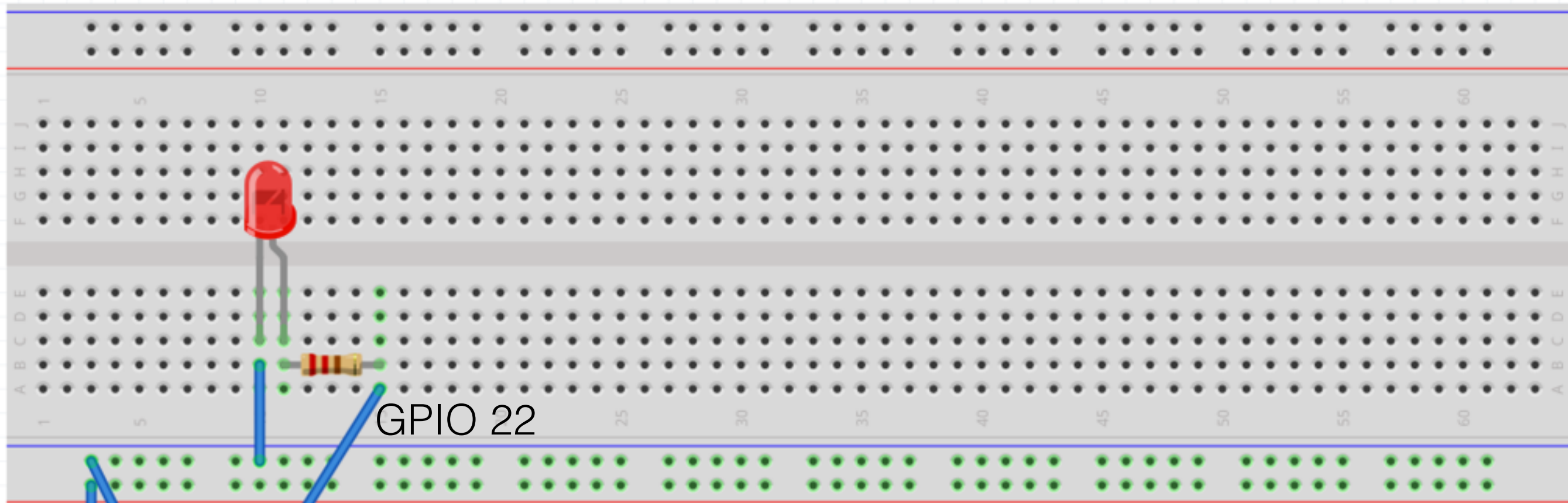
In 'logger.py'

```
#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.BOARD)
#set pin 7 to output mode
GPIO.setup(15, GPIO.OUT)

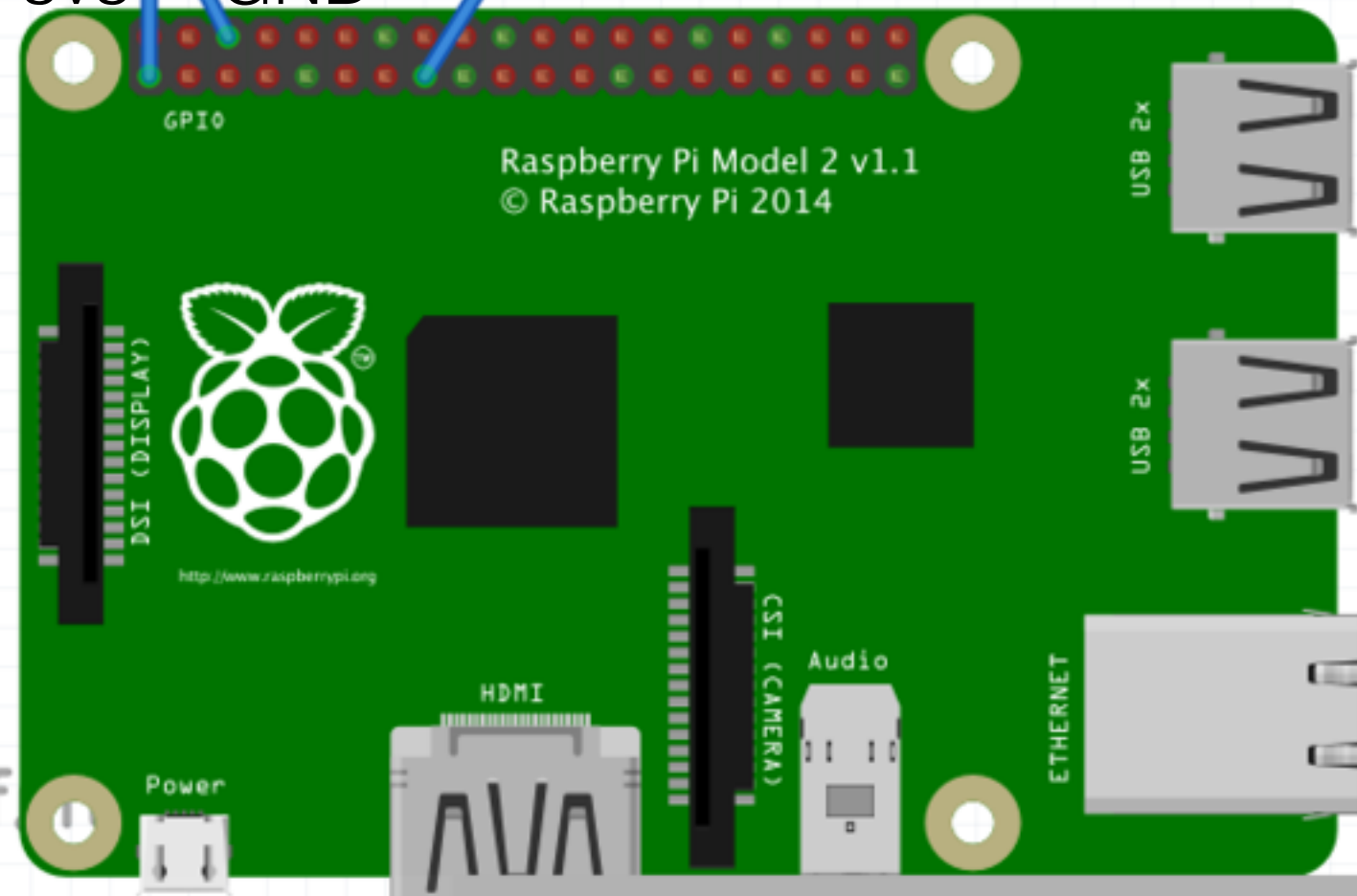
while True:
    #turn on LED and wait 1 second
    GPIO.output(15,True)
    time.sleep(1)
    #turn off LED and wait 1 second
    GPIO.output(15,False)
    time.sleep(1)
```





3v3 GND

GPIO 22





Indicate where to make new python program

In 'logger.py'

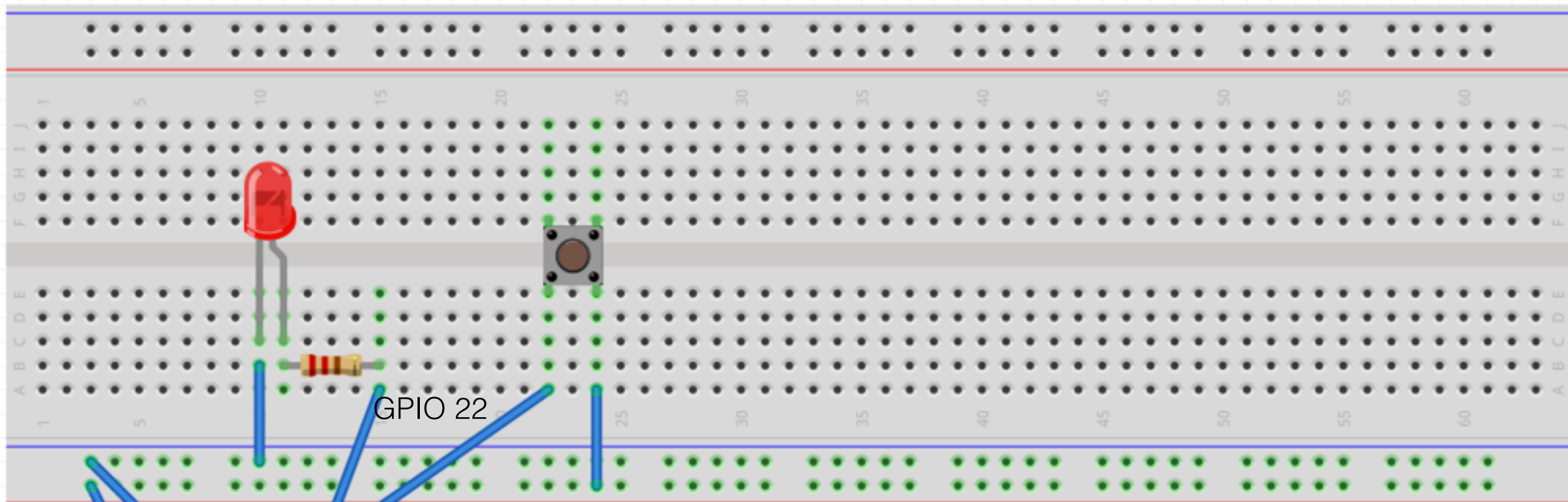
Input from buttons

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

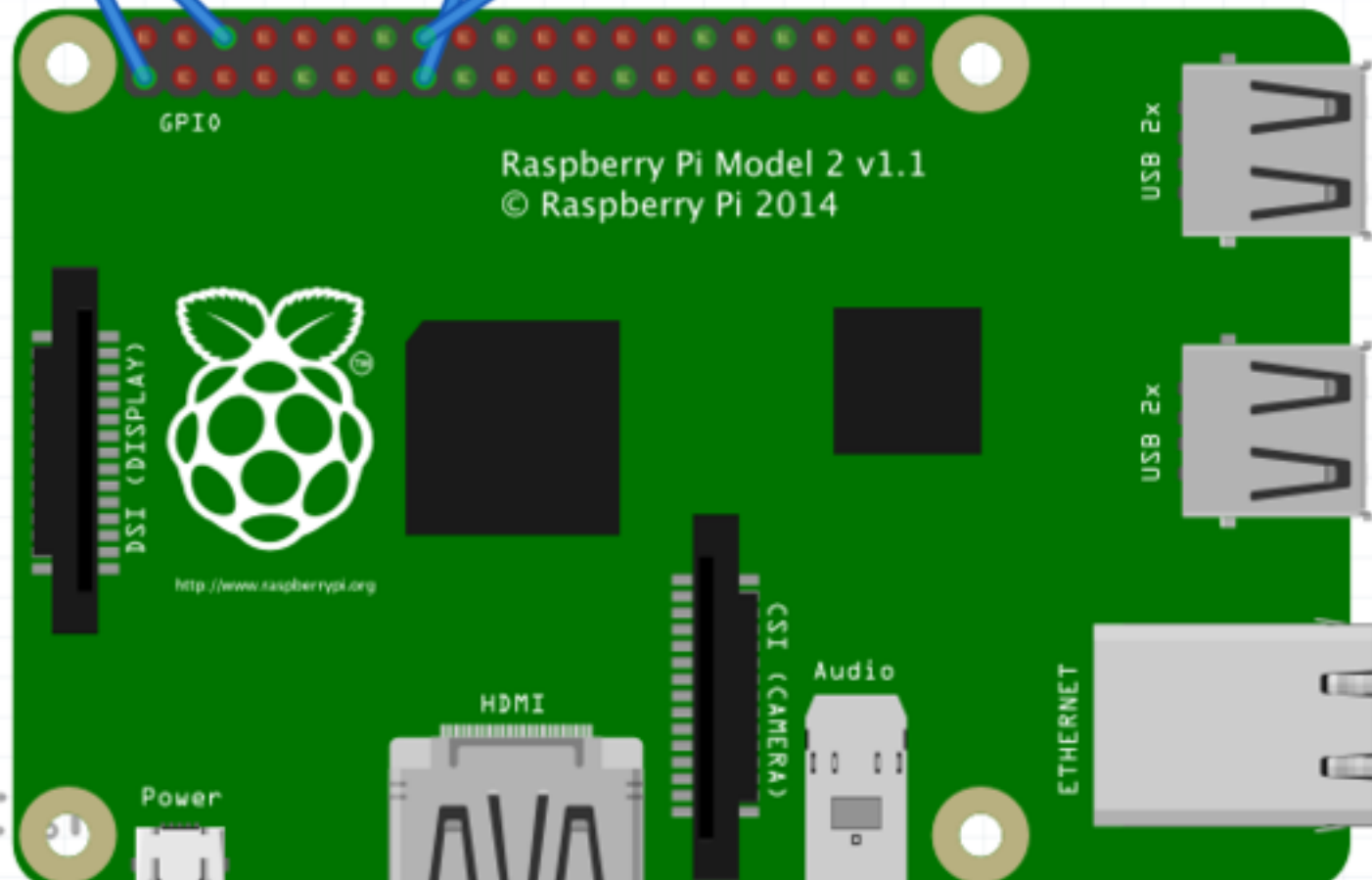
#set up pins
GPIO.setmode(GPIO.BOARD)
GPIO.setup(14, GPIO.IN)
GPIO.setup(15, GPIO.OUT)

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





3v3 GND GPIO 22 GPIO 23





In 'logger.py'

Challenge

Improve wording

Make a program that blinks the LED while the button is released, and stops blinking when the button is pressed.

Blinker solution

Make pretty

```
#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.BOARD)

#set pin 7 to output mode

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)
```



In a new file
called 'temp.py'

7 segment displays

Need better title

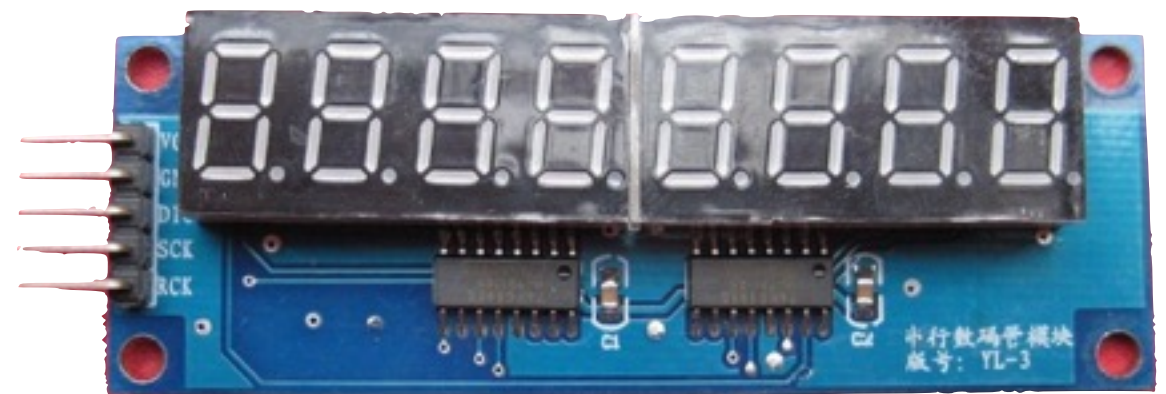
Fritzing diagram

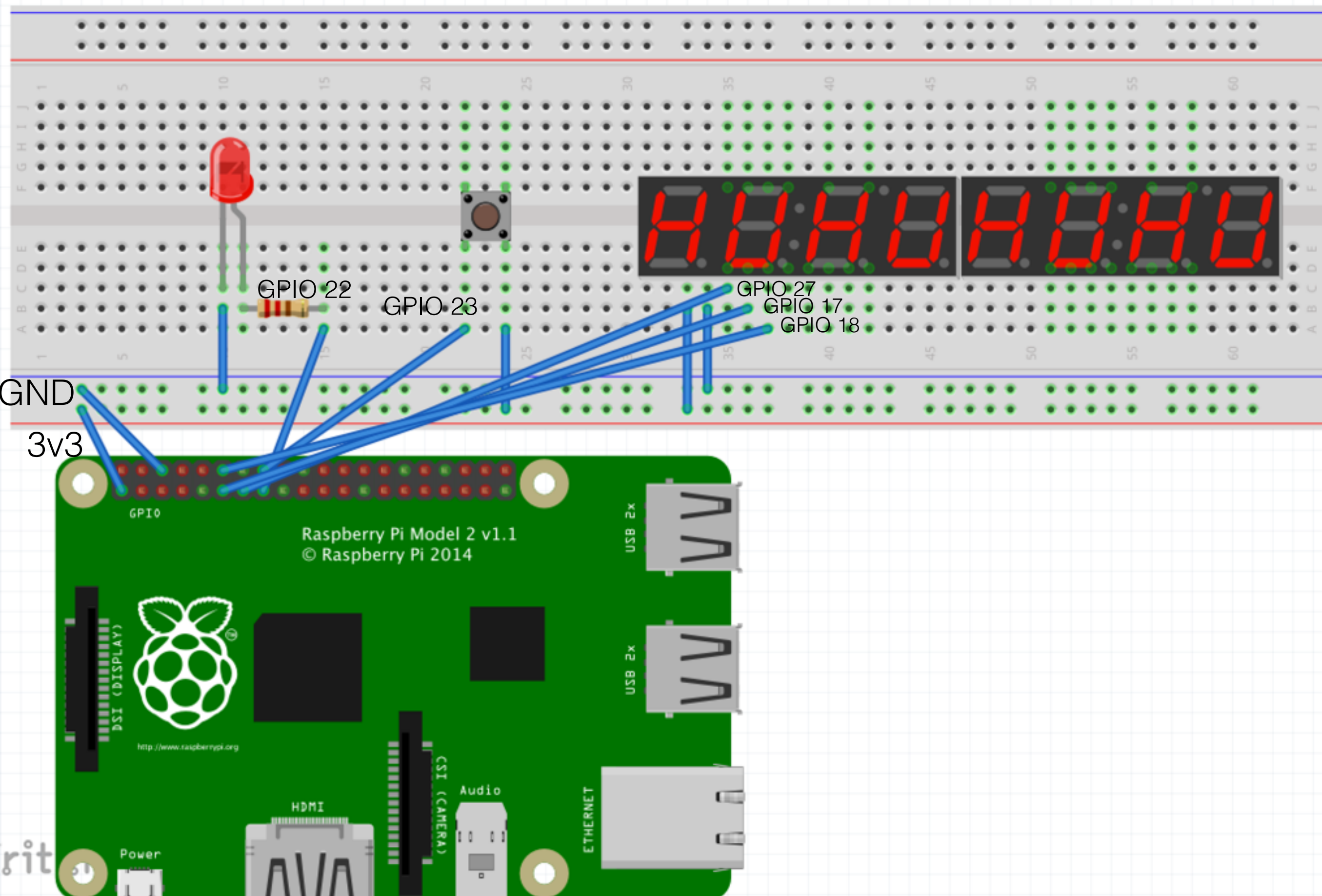
- 7 segment displays operate through a custom module called PiSlice
- Set display by using `PiSlice.display_number`

```
import PiSlice
```

```
PiSlice.init()
```

```
PiSlice.number = 12345678
```







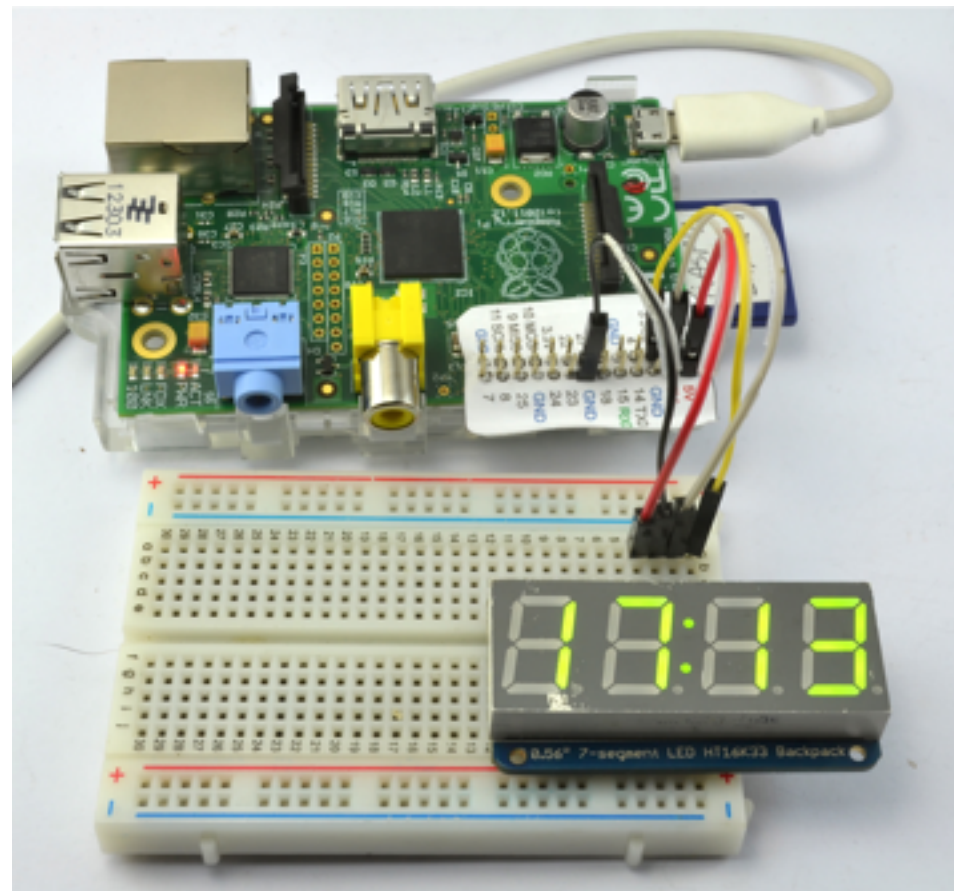
In 'logger.py'

Challenge

investigate debouncing further

Wording?

Add up the number of times you press the button and display it on the 7 segment displays incrementing each time you press the button.



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
```

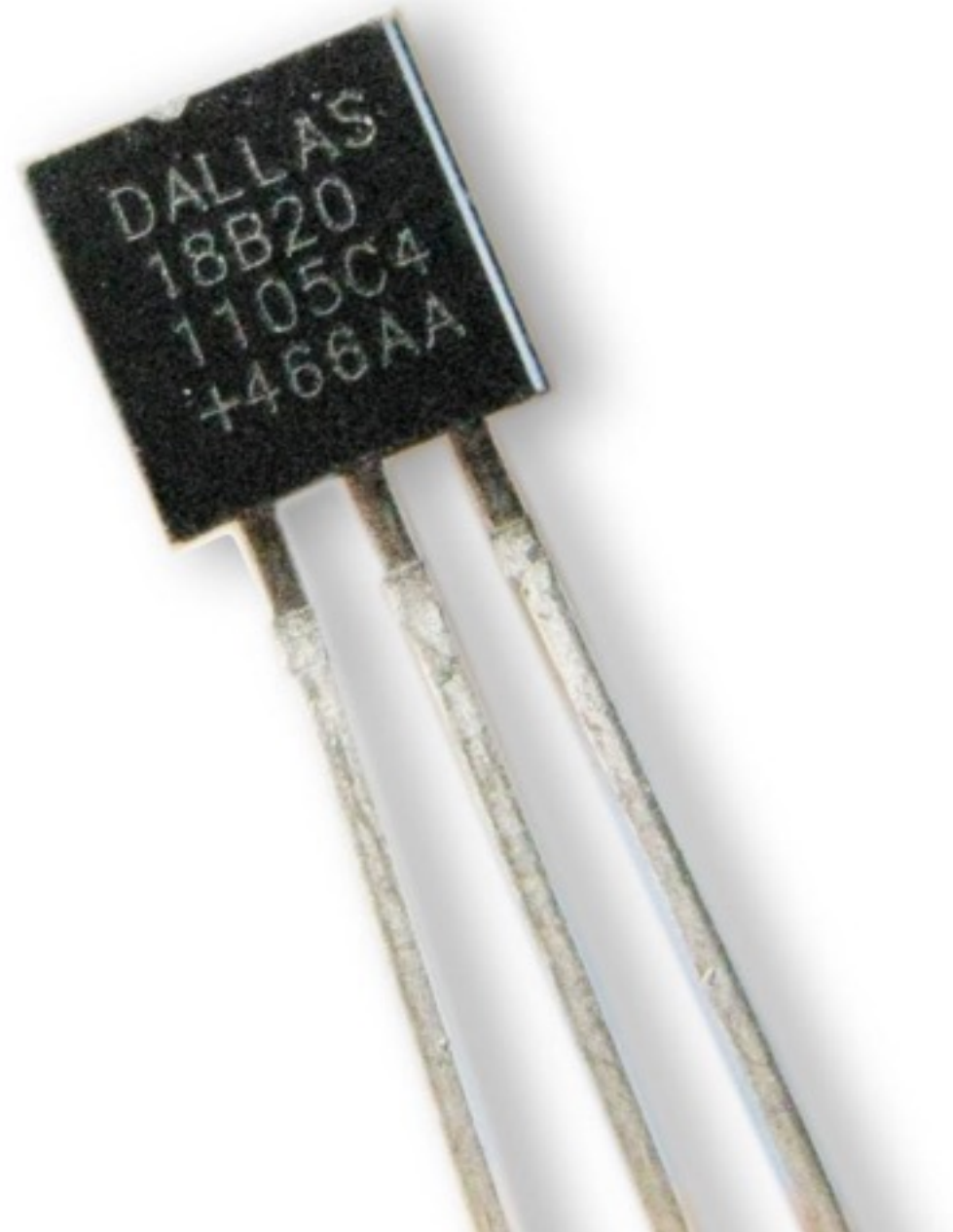


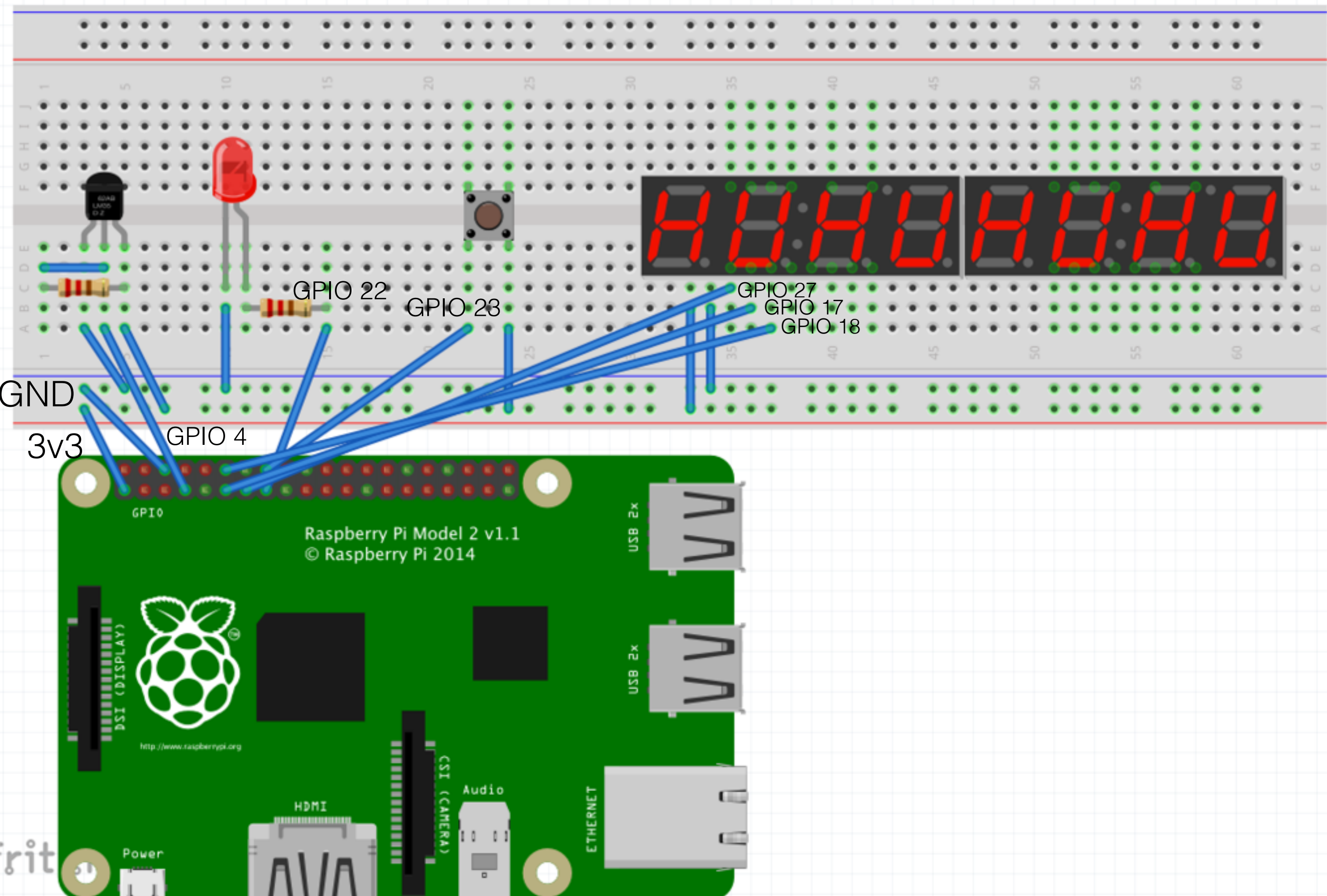
In 'temp.py'

Sensing temperature

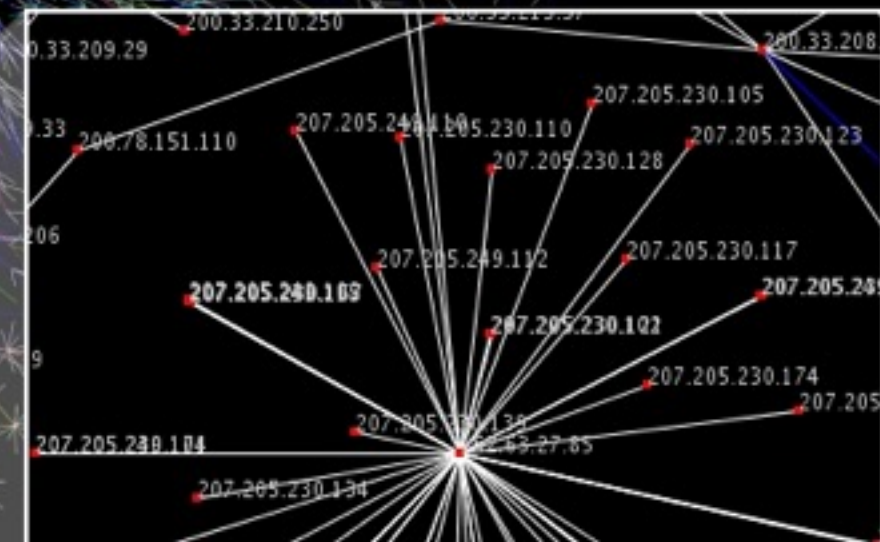
```
import PiSlice

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





Getting your Raspberry Pi project online



Email

- 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
- The password is LetsCode





In a new file
called 'mail.py'

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()
```





In 'logger.py'

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.