Programming Gadgets

Erik Fredericks, 2018
Homepage: http://efredericks.net
GitHub: https://github.com/ou-sbselab/SECS-SummerCamp

All links to images available in speaker notes

How this will work

I talk for a little bit
You do a fun demo for a little bit

Key point: **don’t be shy! Ask questions!**

If you are interested...

All class materials are posted to my lab’s GitHub page
https://github.com/ou-sbselab/SECS-SummerCamp

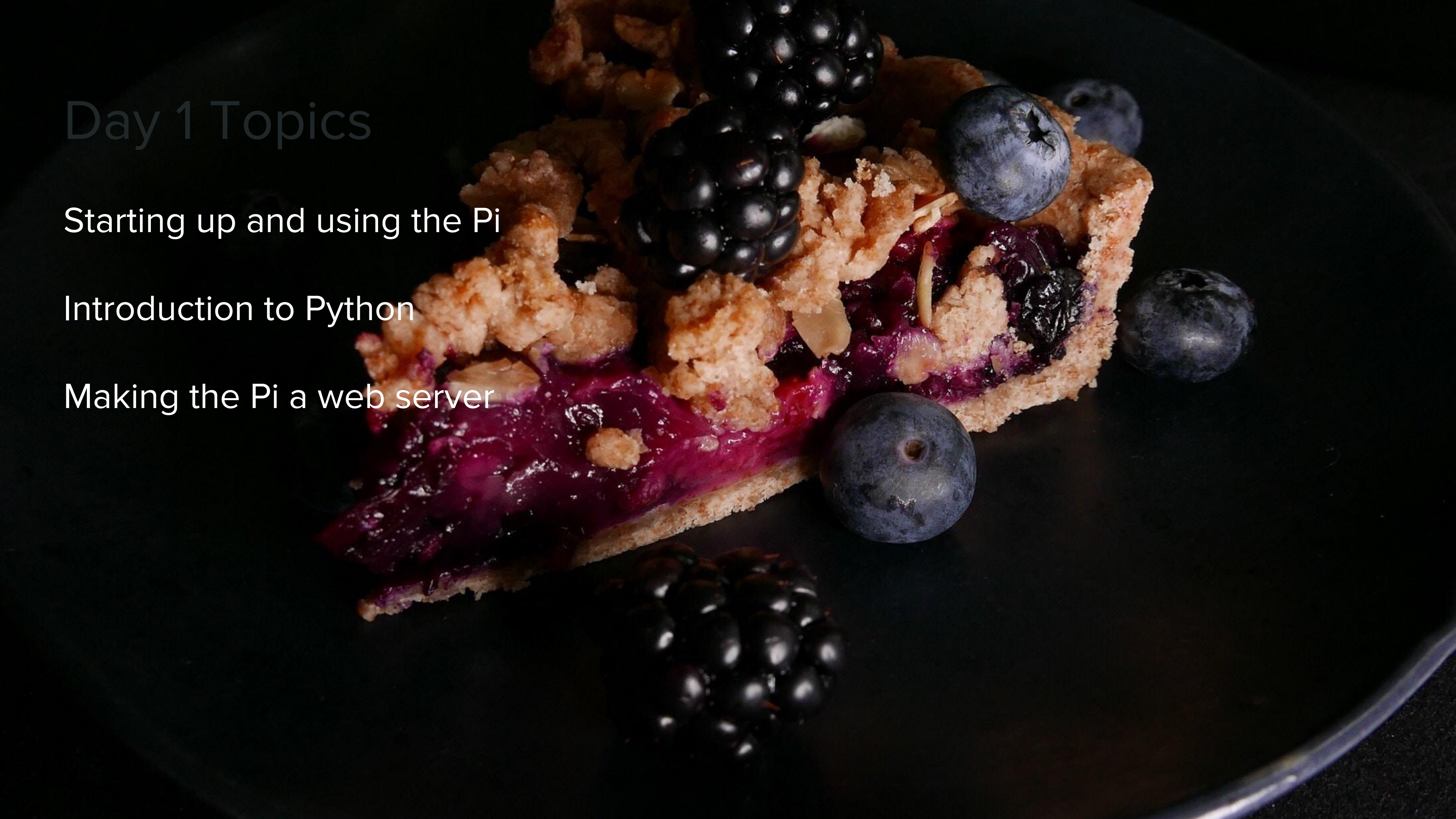
(Your stuff is in programming-gadgets)

Here you can find all the presentations, code, guides, etc.

If you want to work on any of this at home

Raspberry Pi 3B:	~$35.00
Sense Hat:	~$30.00

Or there are numerous kits on Amazon for ~$80 (includes case, cables, etc.)



Day 1 Topics

Starting up and using the Pi

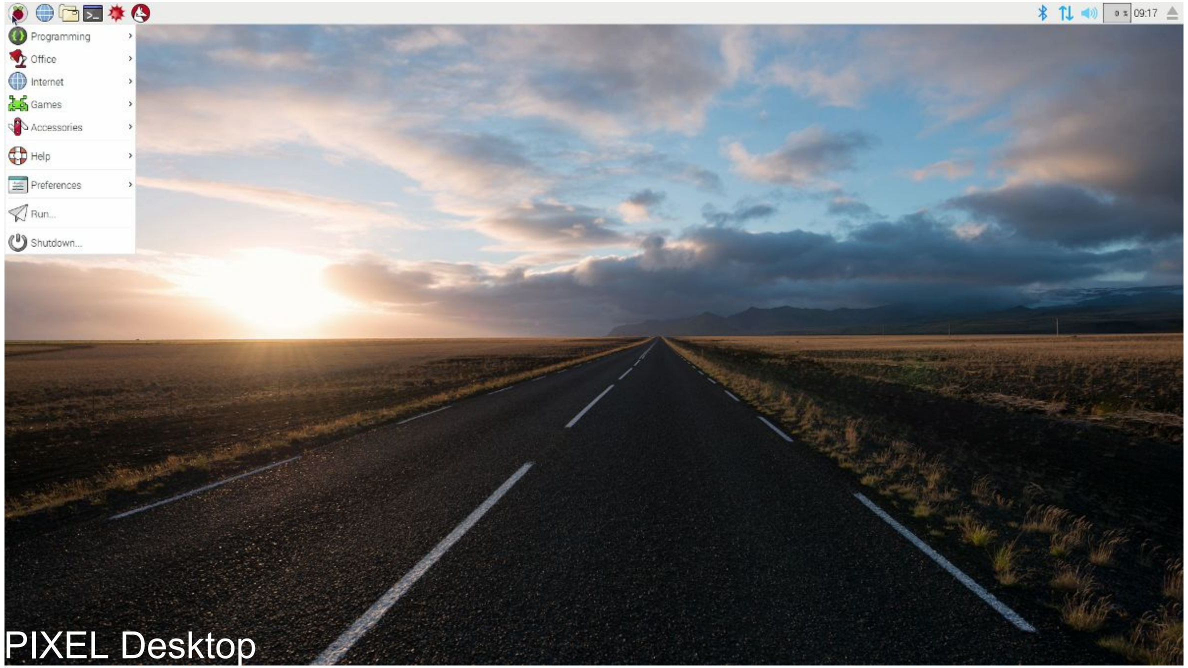
Introduction to Python

Making the Pi a web server

What can these types of systems be used for?

Starting up and using the Raspberry Pi

1) Plug in the HDMI (video) connection
2) Plug the keyboard and mouse into the USB ports
3) Plug in the power
4) …
5) Success!



PIXEL Desktop

Linux??

Alternative operating system

Can run on nearly any computer system

    Including Windows now!

Apple?

Linux??

Raspberry Pi?

Raspbian ➜ variant of Debian Linux

Demo 1 - Navigating PIXEL

- Start Menu
- Opening up a document
- Browsing the internet
- Running IDLE

Internet!

Open a browser (Chromium is the browser)

Try going to google.com

If it takes you to the net registration page, enter your information

My email is fredericks@oakland.edu

About the demos...

All demos are written in **Python**

Easy-to-use programming language

(Great starter language if you're interested in programming)

All files that have a **.py** extension are Python programs

How to run a program called program:

```
$ python program.py
```

Do I need a Raspberry Pi to run Python?

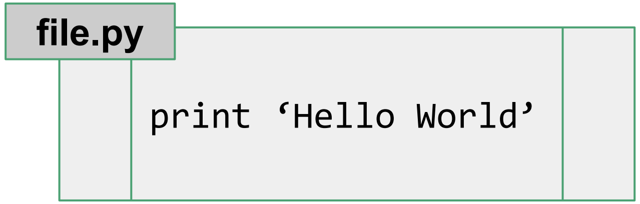
NO

Follow-along code

You can run Python either as a **script** or **interactive**

Script

```
$ python file.py
```



```
print ‘Hello World’
```

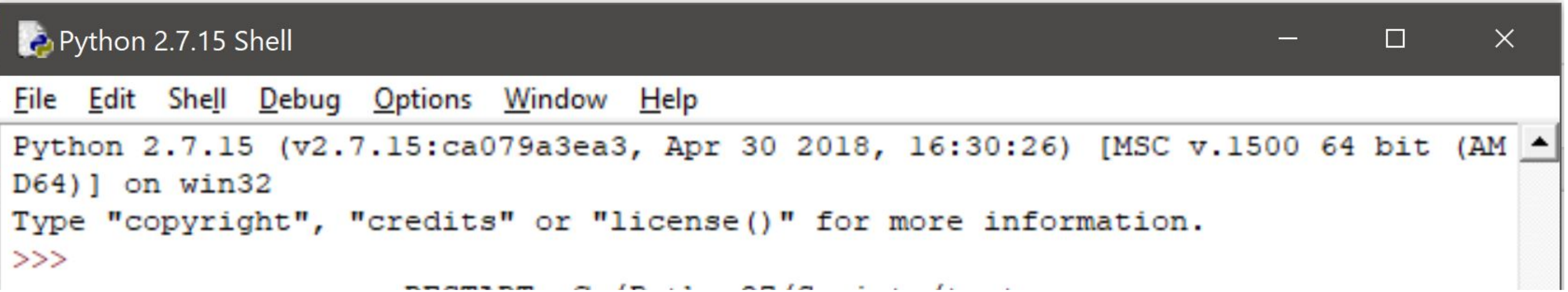
Interactive

```
$ python
>>> print ‘Hello World’
Hello world
>>>
```

Python basics

We're going to use the **IDLE Python IDE**

Start ➜ Programming ➜ IDLE



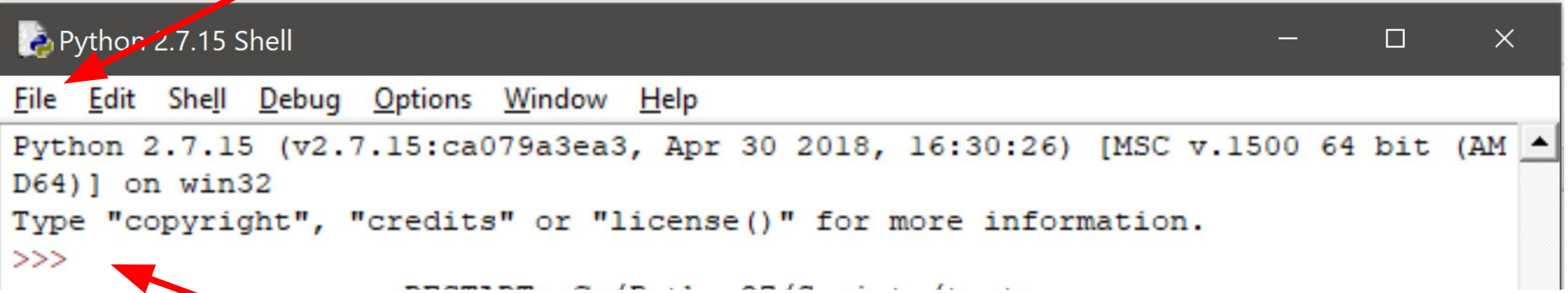
```
Python 2.7.15 (v2.7.15:ca079a3ea3, Apr 30 2018, 16:30:26) [MSC v.1500 64 bit (AM
D64)] on win32
Type "copyright", "credits" or "license()" for more information.
>>>
```

Create script files here

Python basics

We're going to use the **IDLE Python IDE**

Start ➔ Programming ➔ IDLE



```
Python 2.7.15 (v2.7.15:ca079a3ea3, Apr 30 2018, 16:30:26) [MSC v.1500 64 bit (AM
D64)] on win32
Type "copyright", "credits" or "license()" for more information.
>>>
```

Type Python commands here

If you know Python or were here for Adventures in Coding 1:

Get a copy of this file:

https://goo.gl/bfJatC

And this file (your dungeon adventure game):

https://goo.gl/HDy9ae

And hack it so that your Dungeon Adventure uses the joystick to move!

Otherwise follow along with a Python review

(we'll do that thing later as well!)

Hello World (your first Python program)

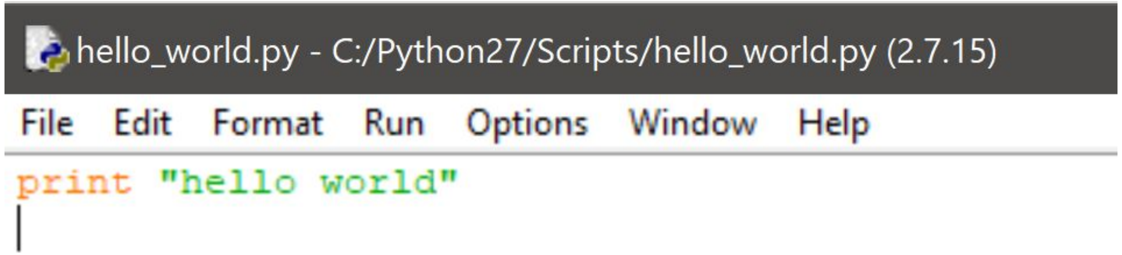
```
>>> print(“hello world”)
hello world
>>>
```

Hello World (your first Python program)

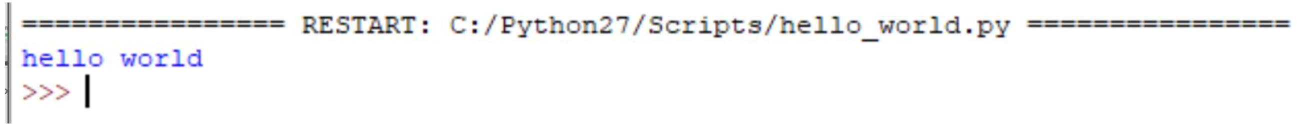
```
>>> print “hello world”
hello world
>>>
```

File → New File
(save as hello_world.py)

Run → Run Module (F5)


hello_world.py - C:/Python27/Scripts/hello_world.py (2.7.15)

```
print "hello world"
```

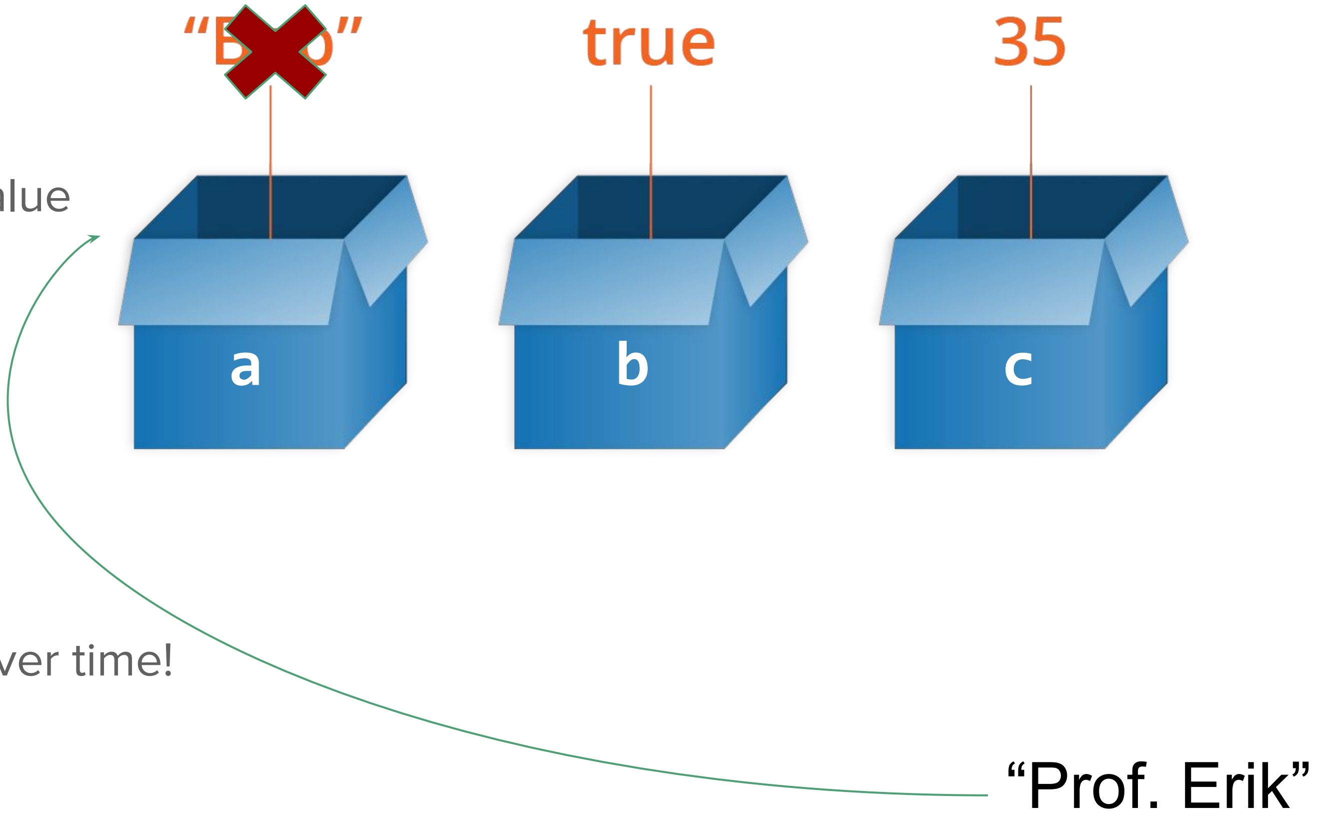


```
================ RESTART: C:/Python27/Scripts/hello_world.py ================
hello world
>>>
```

Variables

An object that **holds** a value

This value can change over time!



Variable names

RULES!

Letters, numbers, and underscores (_) all OK!

NO SPACES

Variables can’t start with a number!

What can a variable hold?

Anything really!

- Numbers
- Letters (strings)
- Objects
- ...

Variable basics

Variables have **no type** until you assign them values

```
my_shiny_new_variable = 10

my_shiny_new_variable = “HELLO”

print my_shiny_new_variable
```

Type ➜ is it a number? a string? an object?

Why do we need variables?

What if we want to print variables *and other things*?

This is called concatenation

Combining two **strings**

```
variable_a = “Hello there”
variable_b = “SECS summer camp!”

print variable_a + “ “ + variable_b
```

???

Time to do a thing!

1) In a new script file (File ➜ New File), create a script that:

   a) Has a variable that stores your FIRST NAME
   b) Prints a statement that says:

```
Hello <FIRST NAME>, welcome to OU!
```

(Make sure to replace <FIRST NAME> with your first name)

How to do the thing!

```
my_name = “Prof. Erik”
print “Hello “ + my_name + “, welcome to OU!”
```

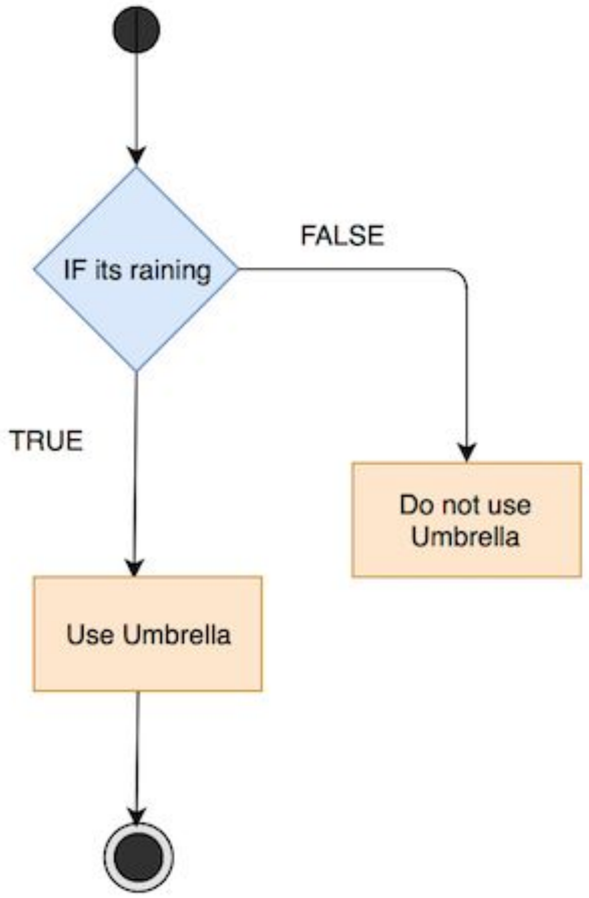
Other ways!

```
output = “Hello %s, welcome to OU!” % (my_name)
```

```
first = “Hello “
second = “, welcome to OU!”
print first + my_name + second
```

Decisions (IF-statements)

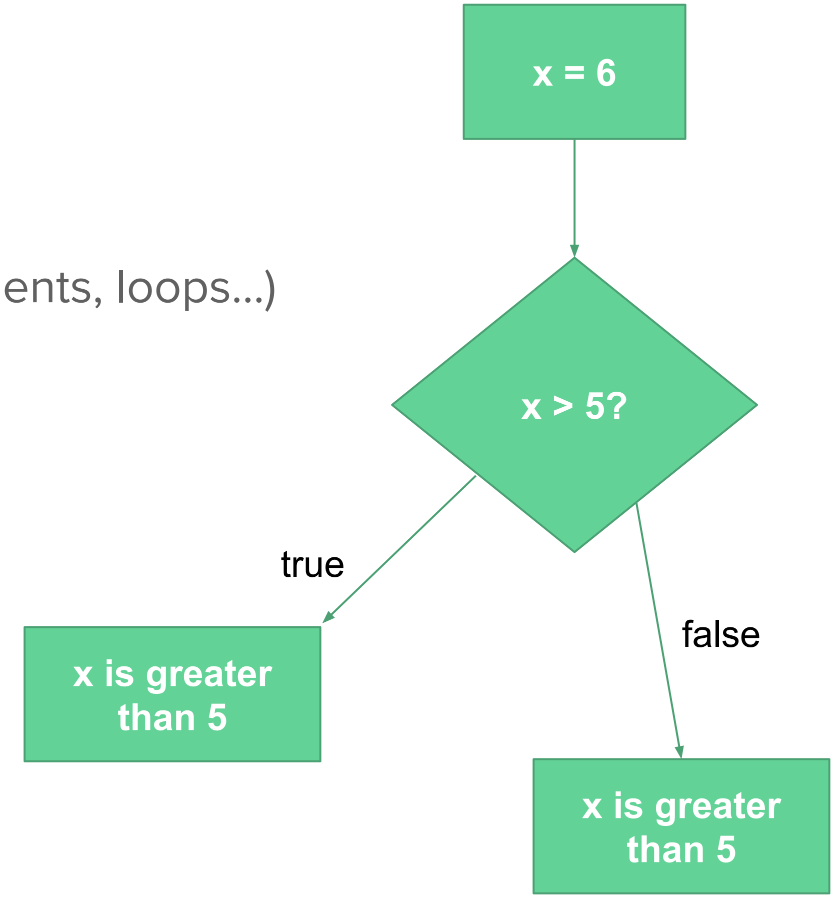
Sometimes you want your program to do things differently



Indenting

Blocks of code work through indents (if statements, loops...)

```
if (x > 5):
  print (‘x is greater than 5’)
else:
  print (‘the else case!’)

------------------------------
```



Are these the same?

```
if (x > 5):
  print (‘x is greater than 5’)
else:
  print (‘the else case!’)

-----------------------------------------------------------------

if (x > 5):
                          print (‘x is greater than 5’)
else:
                          print (‘is this ok?’)
```

Making things happen over (and over (and over)) - loop

```
x = 0
# Let’s get x to 5
x = x + 1
x = x + 1
x = x + 1
x = x + 1
x = x + 1
```

Loops!

```
x = 0
# for loop
for i in range(0,5):
  x = x + 1
```

range?

`range` is a function that returns a list of numbers

When you say: `range(0, 4)`
  You get back: `[0, 1, 2, 3]`

And when we say: `for i in range(0, 4):`

```
  What is happening is:

i = 0
i = 1
i = 2
i = 3
```

Loop practice!

There are lots of different loop types, but a very common one is the **for** loop

Practice with it!

- Make a **for loop** that prints out the loop variable **twenty times**

# For loop reference from earlier
```
for i in range(0,5):
  x = x + 1
```

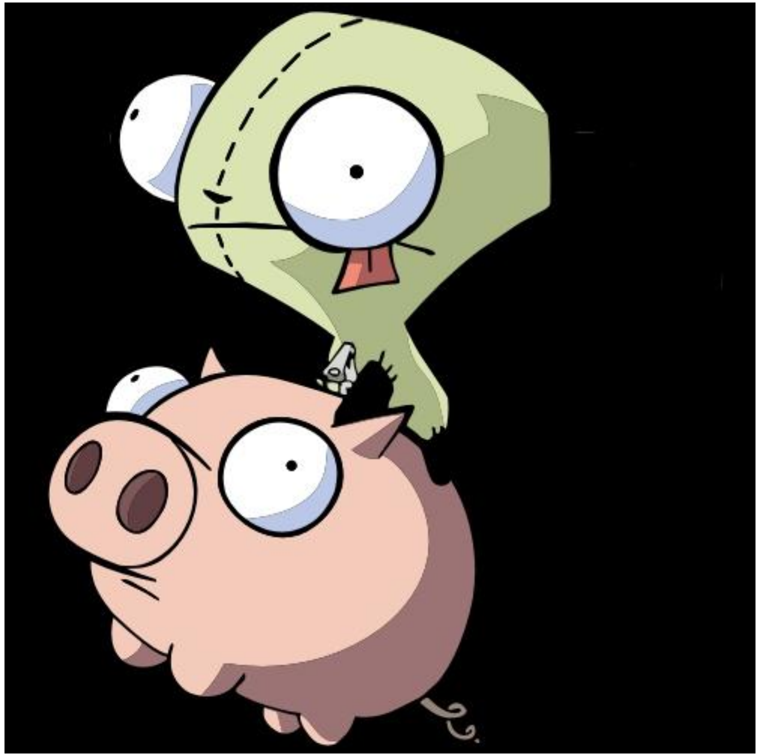
Math!

```
>>> print (3 + 2)

>>> print (5 * 3.5)

>>> a = 5
>>> b = 10
>>> c = a / b
>>> print (c)
```

Game #1: Dice roll

Time to learn how to do random things!

Why is randomness important for programming?



First, import `random`

Our first time **importing** a module!

`import random` → Add extra Python functionality

Random doesn’t exist in our programs until we **import** it

Random number generation

Create a new file: `dice.py`

```
import random
print random.random()
```

Run it a few times and see what happens

Random integers

Last time you generated a `floating` point number (hint: it has lots of decimal points)

Now we want an `integer` (hint: it has no decimal points)

1) Remove `random.random()` from your code
2) Replace it with `random.randint(1,10)`

Run it a few times again

How would you change this to be a die roll?

A quick aside ➜ random.seed

Sometimes you may want to **seed** your random function

This means that your random function will always return the same values each time you run your program

Game #2: Guess the dice roll

For this, let’s extend the previous game. Instead of just rolling dice, let’s roll a user-provided number of dice, and then have the player guess what the number is!

How this will work:

1) Ask the player how many dice to roll

```
variable = raw_input("What is your variable? ")
```

2) Roll the dice and store the total value
3) Tell the player the number range to guess on
4) For each guess:
    a) Tell the player if they are higher or lower than the real value
    b) Keep track of the number of guesses

Debugging

Debugging

Sometimes we have a problem with our code!

One of the easiest and most common ways to fix bugs is **trace debugging**

This means adding **print** statements to see what is going wrong

(If you have a typo, usually the Python interpreter will tell you)

Debugging the dice.py file!

The Hat

Sense Hat

8x8 RGB LED Matrix

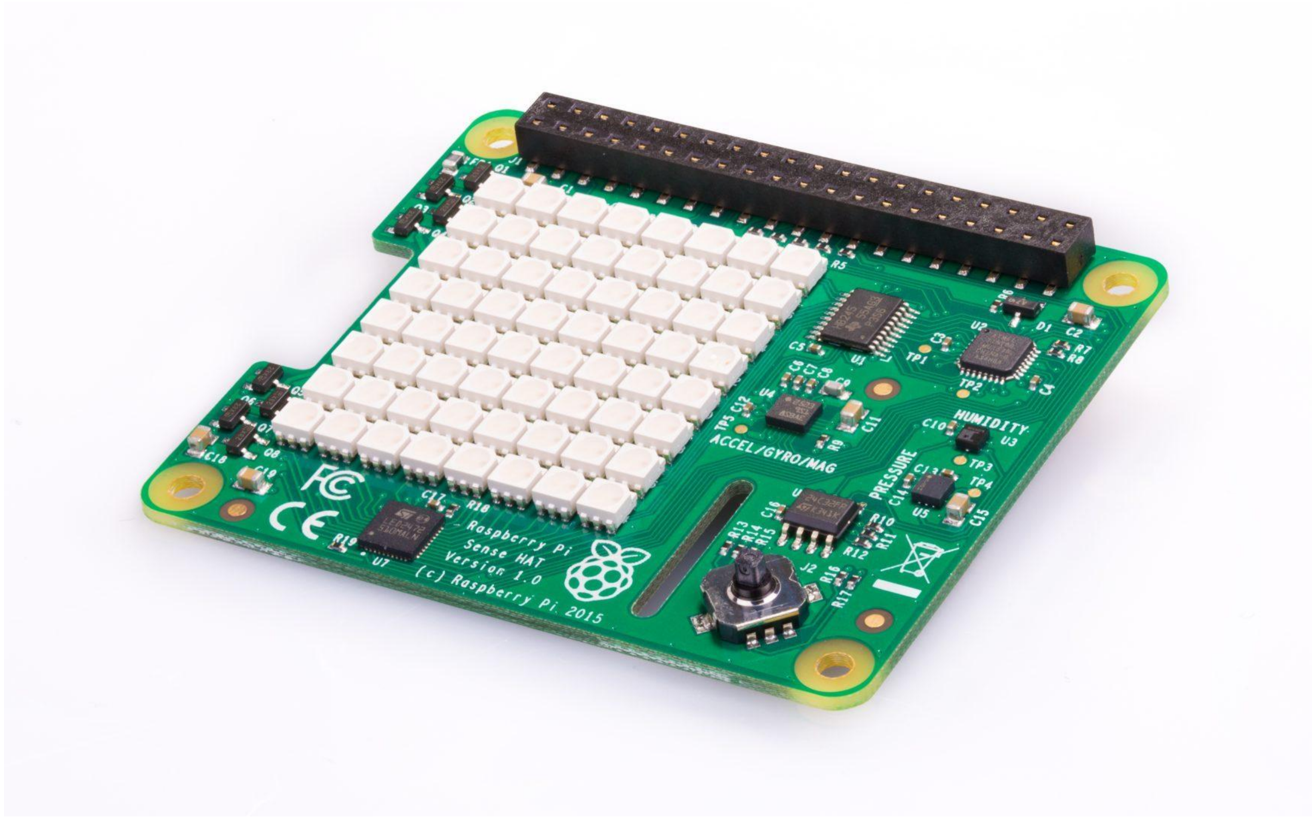
Joystick

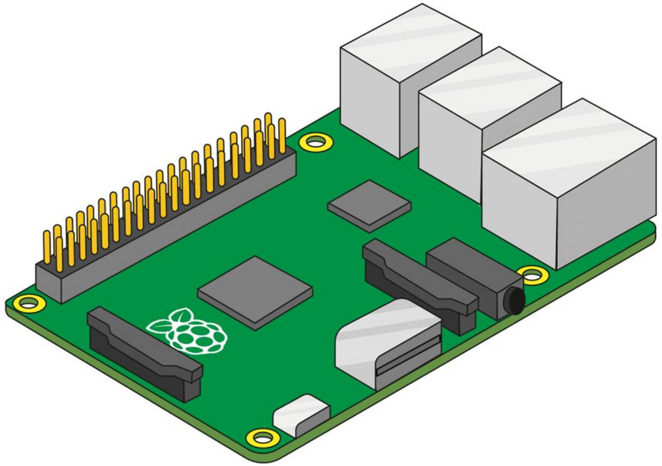
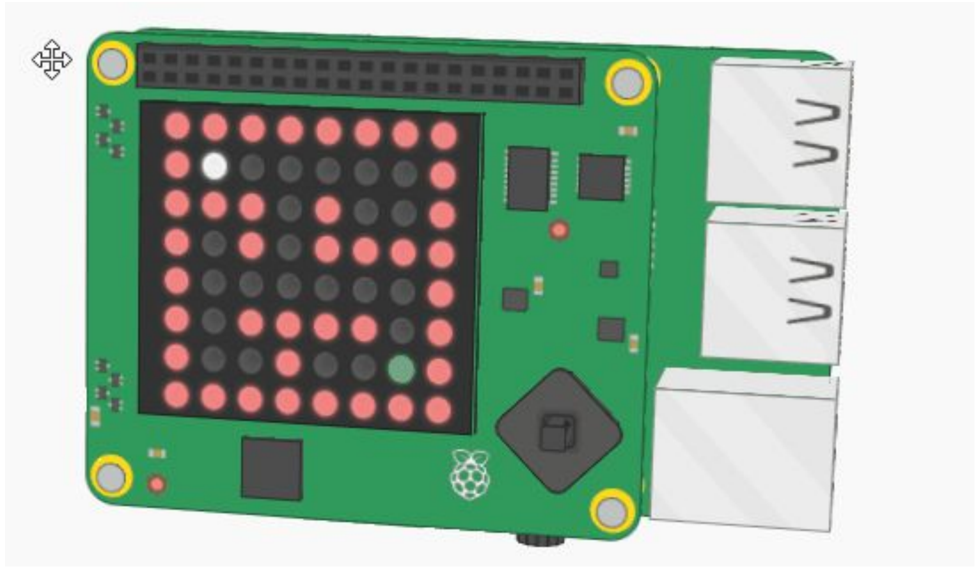
Accelerometer

Temperature

Barometric pressure

Magnetometer



https://goo.gl/CJj9ug

Welcome back

Sensors

Acceleration

Humidity

Pressure

Temperature

Home Server

The Pi can act as a home web server for you!

All you need is a server ➜ Flask

Open up a terminal

```
pip install flask
```

Flask
web development,
one drop at a time

But wait!

We need to do this through the terminal!

Open up a terminal (Start ➔ Accessories ➔ Terminal)

You'll still use IDLE for writing the code

- But you'll run python via the terminal for this

Web

We'll serve our web files via Python

Create **hello-web.py**

```
from flask import Flask
app = Flask(__name__)

@app.route("/")
def hello():
    return "hello world!"
```

```
(In terminal)
FLASK_APP=hello-web.py flask run

(Browse to 127.0.0.1:5000)
```

Let's link our weather station to Flask

Python -- create weather.py (inside weather folder)

```
from flask import Flask
from flask import render_template
app = Flask(__name__)

@app.route('/weather/')
@app.route('/weather/<name>')
def weather(name=None):
    return render_template('weather.html', name=name)
```

Create folder structure

Make a directory:

```
    templates
```

Create `weather.html` (inside templates)

```
<!doctype html>
<title>Super-local weather</title>

{% if name %}
<h1>Hello {% name %}</h1>
{% else %}
<h1>Hello anonymous user!</h1>
</html>
```

Try it out!

Run your app:

```
FLASK_APP=weather.py flask run
```

In the browser:

```
127.0.0.1:5000
127.0.0.1:5000/YOUR NAME
```

Show off your weather data! (Update `weather.html`)

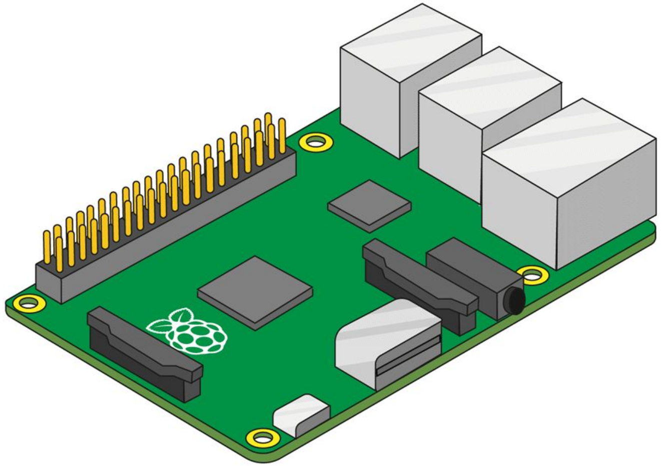
```
...

<div style="border: 1px solid #666; width: 50%">
<ul>
  <li>Temperature: {{ temperature }}</li>
  <li>Humidity: {{ humidity }}</li>
  <li>Pressure: {{ pressure }}</li>
</ul>
</div>

</html>
```

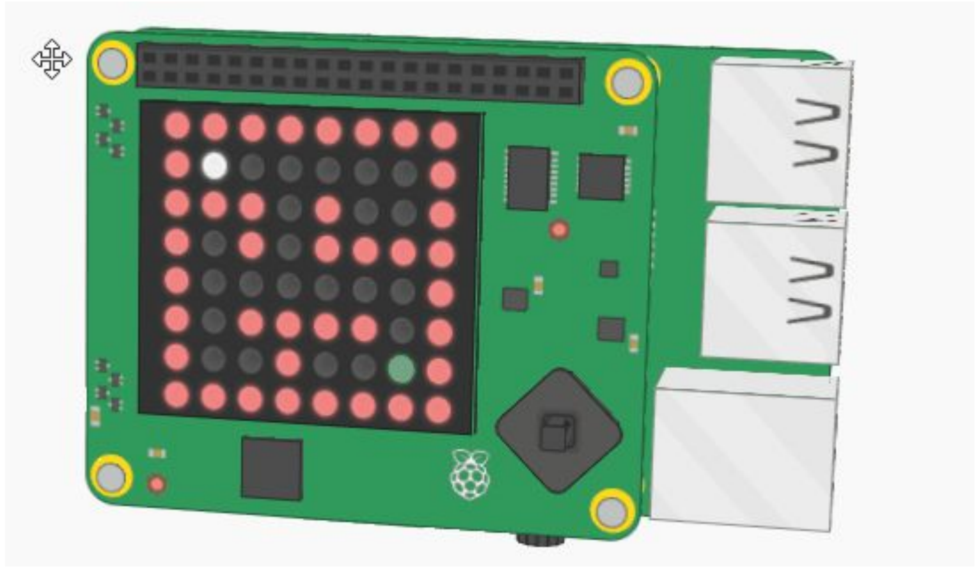
And then copy over your code from your other file for recording various sensor data.

Look at your old sensors file and remember:

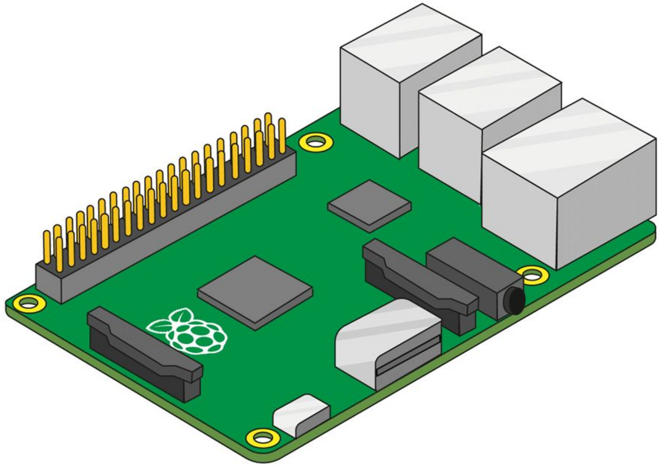
- You need to handle the Sense hat
- You'll need to call the variables appropriately
- You can extend the `render_template` function to include other variables

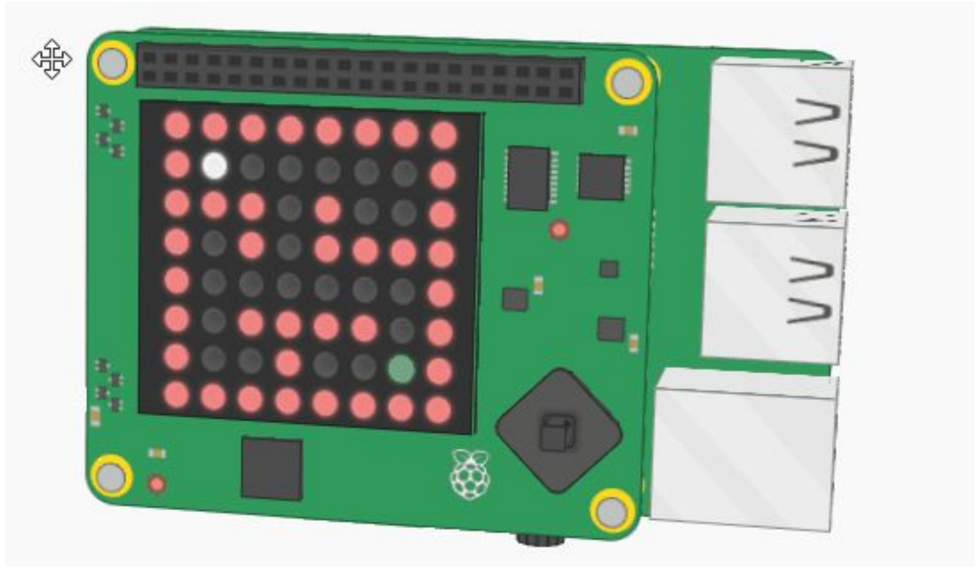
Hello World



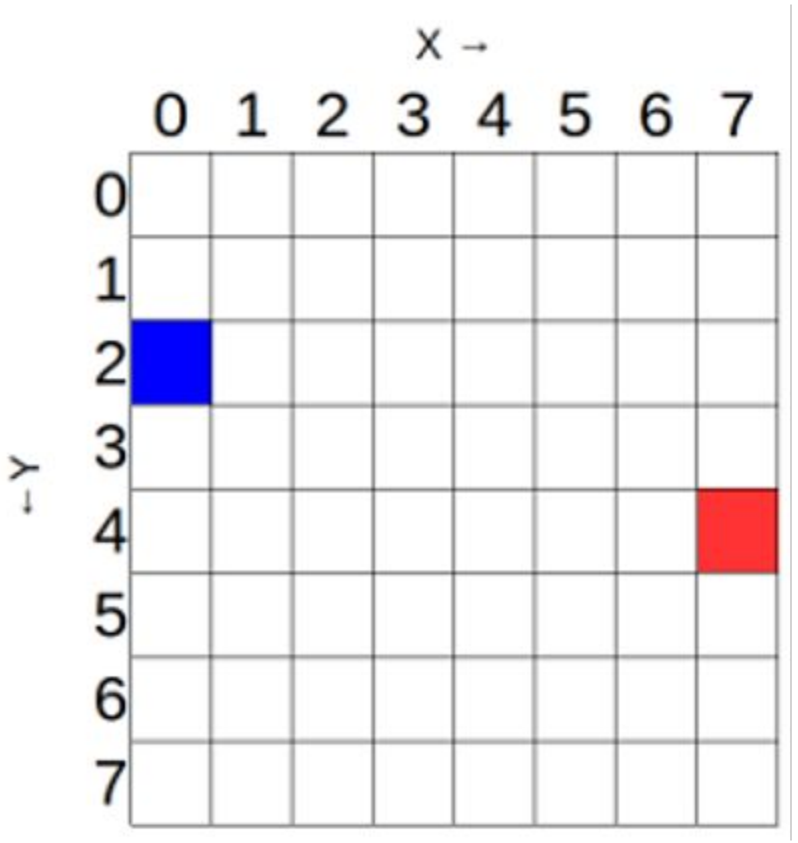


Letters

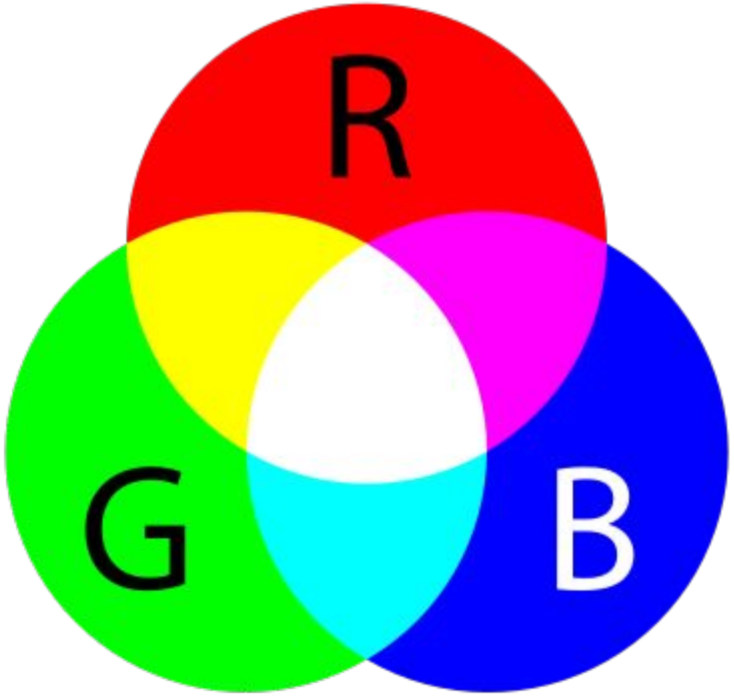




LED Matrix



Colors

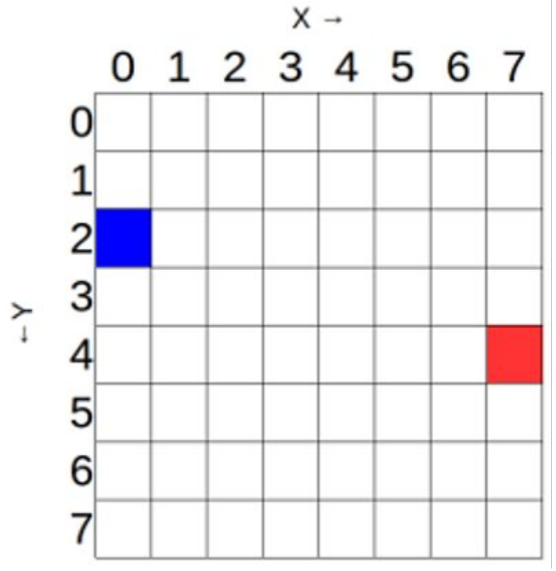


```
R: [0, 255]
G: [0, 255]
B: [0, 255]
```

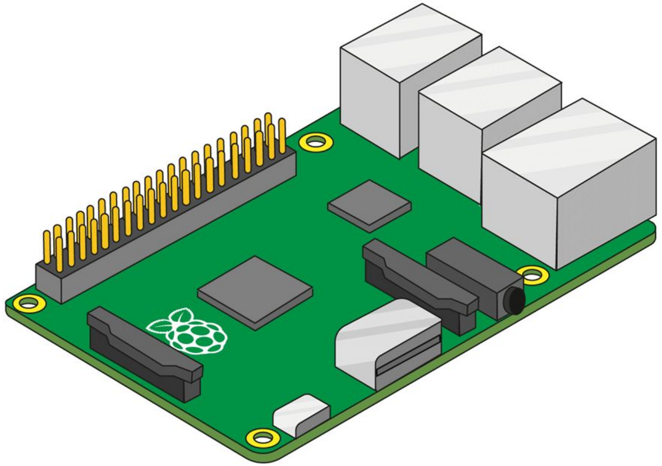
https://www.rapidtables.com/web/color/RGB_Color.html

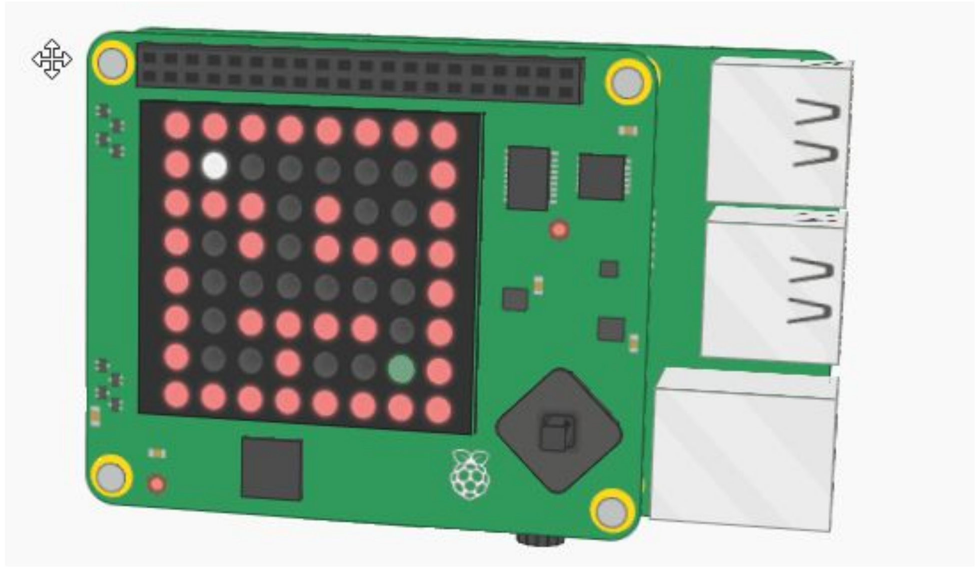
LED Matrix

```
from sense_hat import SenseHat
sense = SenseHat()

red   = (255,0,0)
blue  = (0,0,255)
green = (0,255,0)

sense.set_pixel(0, 2, blue)
sense.set_pixel(7, 4, red)
```

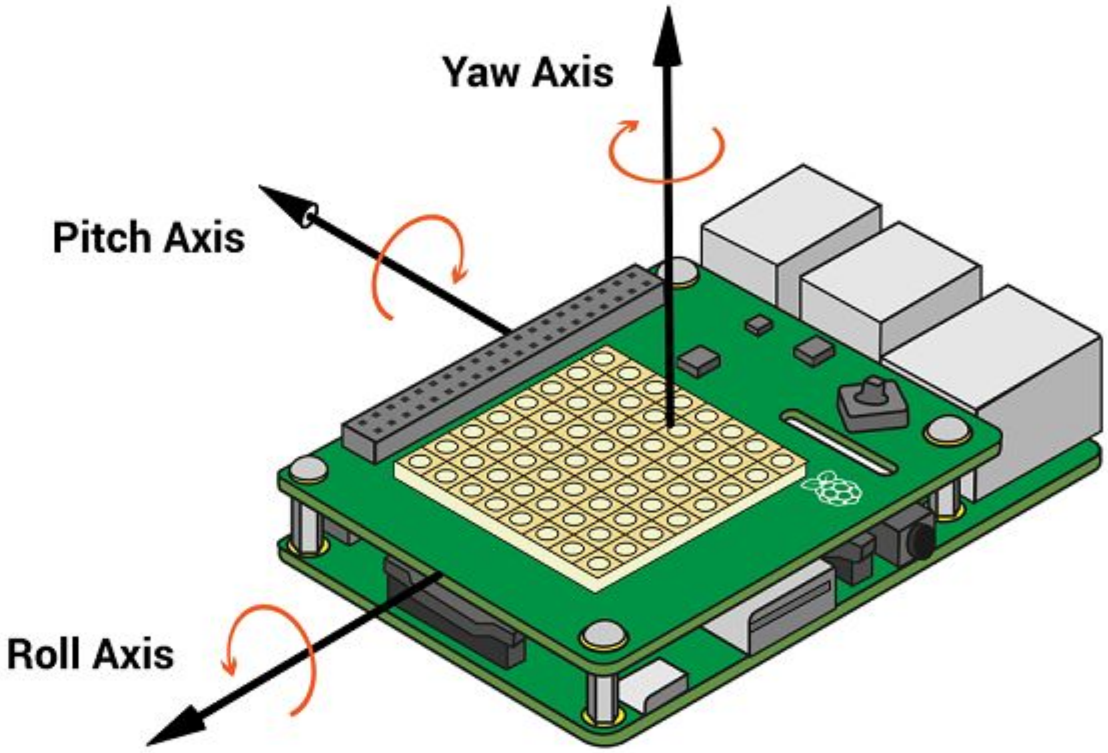


Joystick!





Marble Maze


Yaw Axis
Pitch Axis
Roll Axis

Adventure Game with the Hat

Get a copy of this file:

https://goo.gl/bfJatC

And this file (your dungeon adventure game):

https://goo.gl/HDy9ae

And hack it so that your Dungeon Adventure uses the joystick to move!