# Python DeCal

Week 5

# Announcements

- 3rd Hw is due Wednesday!
- Office Hour
  - Thanks to those who showed up!!
- Attendance!
  - https://forms.gle/bUHiLgePDWo9YvKo8

# Recap

- What is the difference between a numpy **`array`** and a python **`list`**?
- Why would you ever want to use a **`2D array`**? What do they remind you of?

# Little Extra on Numpy

Create array with specific range/number of items

```
numpy.arange([start], stop, [step])

numpy.linspace(start, stop, number of steps)
```

These are super important. Very similar to `range()`

# How do we get Data?

# Load data from a ASCII *.txt file

We can also import data to our jupyter notebook!

```
numpy.loadtxt("file_name", delimiter=None, skiprows=0, usecols=None, unpack=False)
```

| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| North Ameri | United State | 3/1/20 | 69 | 3 | 4.857 | 1 | 1 | 0.143 | 0.208 | 0.009 | 0.015 |
| North Ameri | United State | 3/2/20 | 89 | 20 | 7.714 | 2 | 1 | 0.286 | 0.269 | 0.06 | 0.023 |
| North Ameri | United State | 3/3/20 | 103 | 14 | 7.143 | 6 | 4 | 0.857 | 0.311 | 0.042 | 0.022 |
| North Ameri | United State | 3/4/20 | 125 | 22 | 10.286 | 9 | 3 | 1.286 | 0.378 | 0.066 | 0.031 |
| North Ameri | United State | 3/5/20 | 159 | 34 | 14.286 | 11 | 2 | 1.571 | 0.48 | 0.103 | 0.043 |
| North Ameri | United State | 3/6/20 | 233 | 74 | 24.714 | 12 | 1 | 1.714 | 0.704 | 0.224 | 0.075 |
| North Ameri | United State | 3/7/20 | 338 | 105 | 38.857 | 14 | 2 | 2 | 1.021 | 0.317 | 0.117 |
| North Ameri | United State | 3/8/20 | 433 | 95 | 52 | 17 | 3 | 2.286 | 1.308 | 0.287 | 0.157 |
| North Ameri | United State | 3/9/20 | 554 | 121 | 66.429 | 21 | 4 | 2.714 | 1.674 | 0.366 | 0.201 |
| North Ameri | United State | 3/10/20 | 754 | 200 | 93 | 26 | 5 | 2.857 | 2.278 | 0.604 | 0.281 |
| North Ameri | United State | 3/11/20 | 1025 | 271 | 128.571 | 28 | 2 | 2.714 | 3.097 | 0.819 | 0.388 |
| North Ameri | United State | 3/12/20 | 1312 | 287 | 164.714 | 30 | 2 | 2.714 | 3.964 | 0.867 | 0.498 |
| North Ameri | United State | 3/13/20 | 1663 | 351 | 204.286 | 40 | 10 | 4 | 5.024 | 1.06 | 0.617 |
| North Ameri | United State | 3/14/20 | 2174 | 511 | 262.286 | 47 | 7 | 4.714 | 6.568 | 1.544 | 0.792 |
| North Ameri | United State | 3/15/20 | 2951 | 777 | 359.714 | 57 | 10 | 5.714 | 8.915 | 2.347 | 1.087 |
| North Ameri | United State | 3/16/20 | 3774 | 823 | 460 | 69 | 12 | 6.857 | 11.402 | 2.486 | 1.39 |
| North Ameri | United State | 3/17/20 | 4661 | 887 | 558.143 | 85 | 16 | 8.429 | 14.081 | 2.68 | 1.686 |
| North Ameri | United State | 3/18/20 | 6427 | 1766 | 771.714 | 108 | 23 | 11.429 | 19.417 | 5.335 | 2.331 |
| North Ameri | United State | 3/19/20 | 9415 | 2988 | 1157.571 | 150 | 42 | 17.143 | 28.444 | 9.027 | 3.497 |
| North Ameri | United State | 3/20/20 | 14250 | 4835 | 1798.143 | 150 | 0 | 15.714 | 43.051 | 14.607 | 5.432 |
| North Ameri | United State | 3/21/20 | 19624 | 5374 | 2492.857 | 260 | 110 | 30.429 | 59.287 | 16.236 | 7.531 |
| North Ameri | United State | 3/22/20 | 26747 | 7123 | 3399.429 | 340 | 80 | 40.429 | 80.806 | 21.519 | 10.27 |
| North Ameri | United State | 3/23/20 | 35206 | 8459 | 4490.286 | 471 | 131 | 57.429 | 106.362 | 25.556 | 13.566 |
| North Ameri | United State | 3/24/20 | 46442 | 11236 | 5968.714 | 590 | 119 | 72.143 | 140.307 | 33.945 | 18.032 |
| North Ameri | United State | 3/25/20 | 55231 | 8789 | 6972 | 801 | 211 | 99 | 166.86 | 26.553 | 21.063 |
| North Ameri | United State | 3/26/20 | 69194 | 13963 | 8539.857 | 1050 | 249 | 128.571 | 209.044 | 42.184 | 25.8 |
| North Ameri | United State | 3/27/20 | 85991 | 16797 | 10248.714 | 1296 | 246 | 163.714 | 259.789 | 50.746 | 30.963 |
| North Ameri | United State | 3/28/20 | 104686 | 18695 | 12151.714 | 1707 | 411 | 206.714 | 316.269 | 56.48 | 36.712 |
| North Ameri | United State | 3/29/20 | 124665 | 19979 | 13988.286 | 2191 | 484 | 264.429 | 376.628 | 60.359 | 42.26 |
| North Ameri | United State | 3/30/20 | 143025 | 18360 | 15402.714 | 2509 | 318 | 291.143 | 432.096 | 55.468 | 46.534 |
| North Ameri | United State | 3/31/20 | 164620 | 21595 | 16882.571 | 3170 | 661 | 368.571 | 497.337 | 65.241 | 51.004 |
| North Ameri | United State | 4/1/20 | 189618 | 24998 | 19198.143 | 4079 | 909 | 468.286 | 572.859 | 75.522 | 58 |
| North Ameri | United State | 4/2/20 | 216721 | 27103 | 21075.286 | 5138 | 1059 | 584 | 654.741 | 81.882 | 63.671 |

# Load data from a txt file-Example

```
>>> data = np.loadtxt(“sample_data.txt”, skiprows = 1)

>>> print(data)  →  numpy array with data
```

# Looking for the data?

We can ask numpy to find indexes of specific data using numpy.where

```
numpy.where(condition)
```

Example:

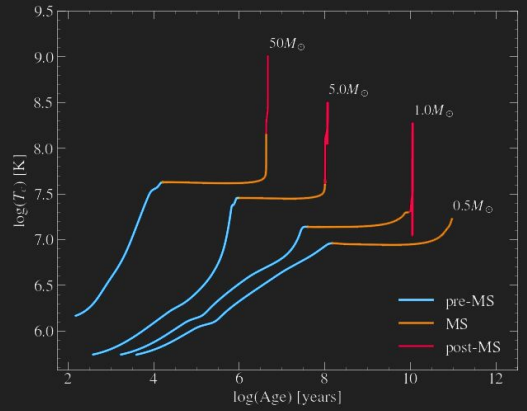
```
>>> a = np.arange(10)

>>> numpy.where(a < 5)

      (array([0, 1, 2, 3, 4]),)
```

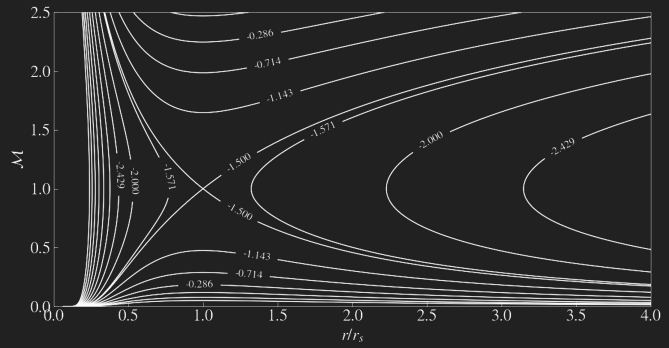
Looks funny, so add on [0] to get a 1D array to work with!

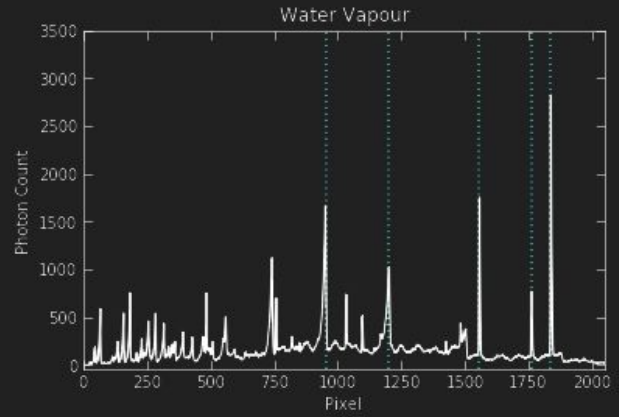
# Looking for the data?-Example

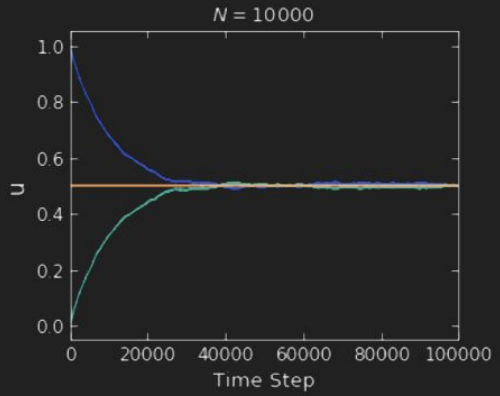
```
>>> a = np.arange(10)

>>> a

array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])

>>> np.where(a < 5, a, 10*a)

array([ 0,  1,  2,  3,  4, 50, 60, 70, 80, 90])
```

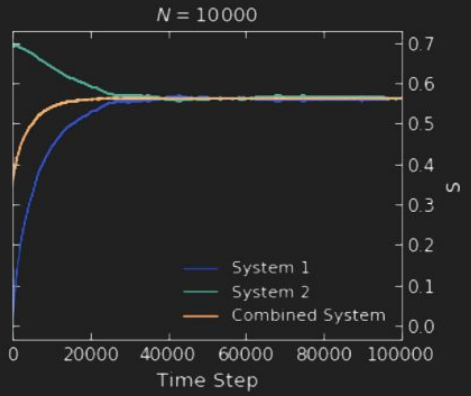
# How do we represent data after analysing them...



# What package to import?

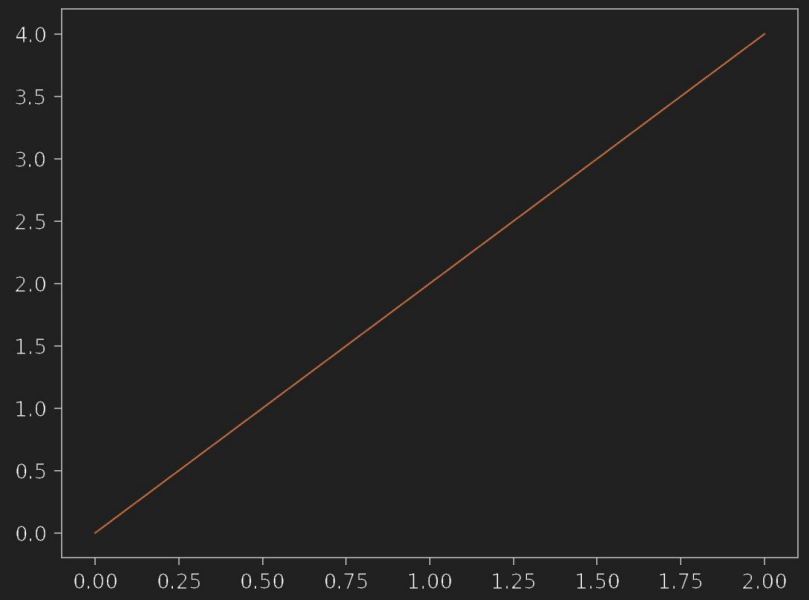
```
import matplotlib.pyplot as plt
```

Whatever type of research you do, you will need this!

matplotlib

# Begin with the simplest…Let's plot a line lol.

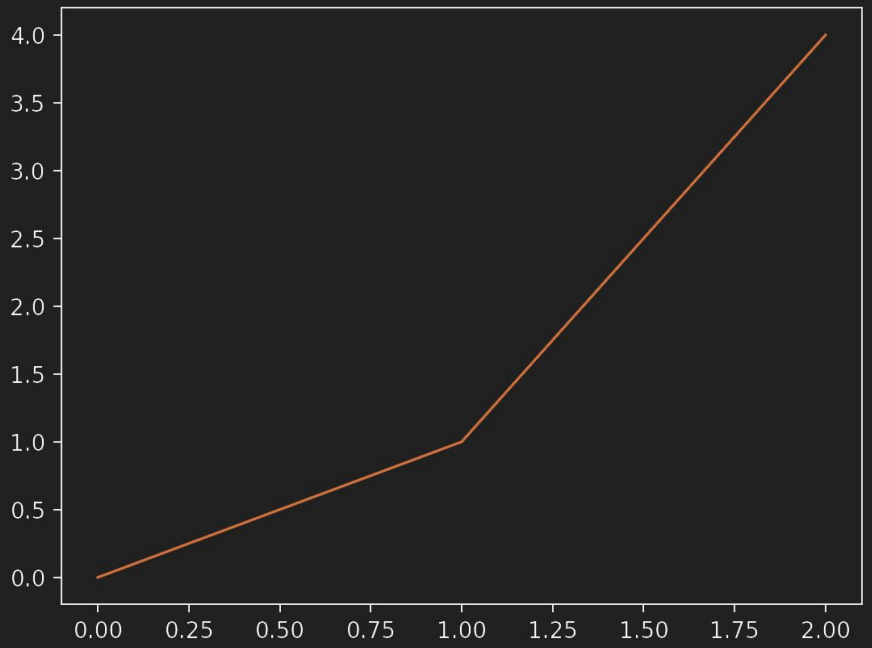
```
plt.plot(arr)

plt.plot(x_arr, y_arr)
```




```
>>> x = np.arange(3)

>>> y = 2*x

>>> plt.plot(x, y)

>>> plt.show()  (jupyter notebook)
```

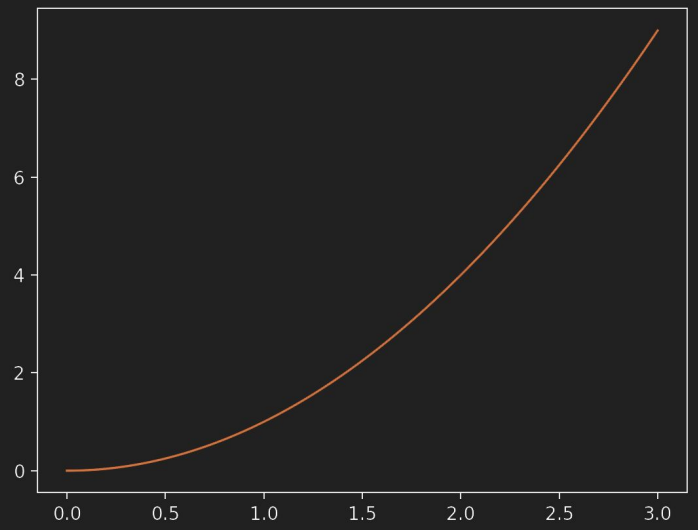
# Now, what about a curve?

```
>>> x = np.arange(3)
>>> y = x**2
>>> plt.plot(x, y)
```



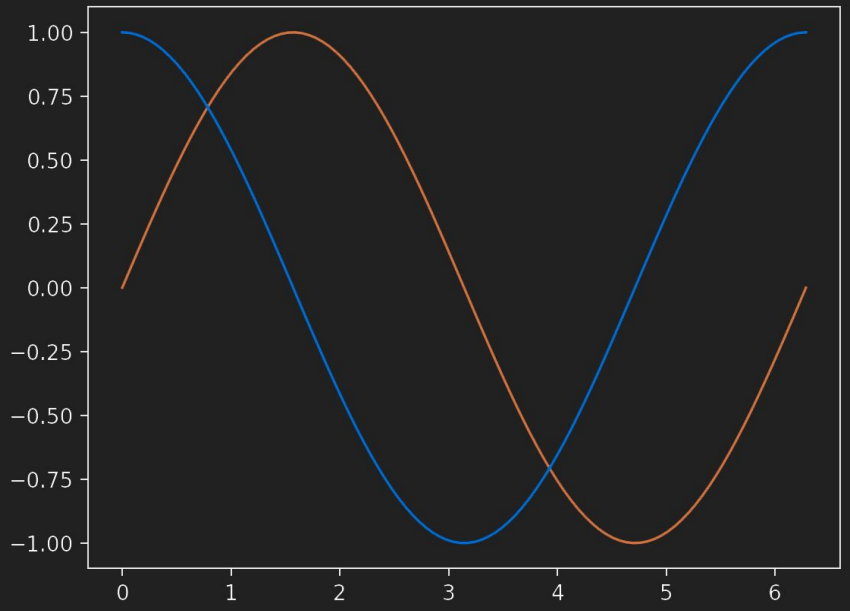
# Now, what about a curve?

```
>>> x = np.linspace(0,3,100)
>>> y = x**2
>>> plt.plot(x, y)
```



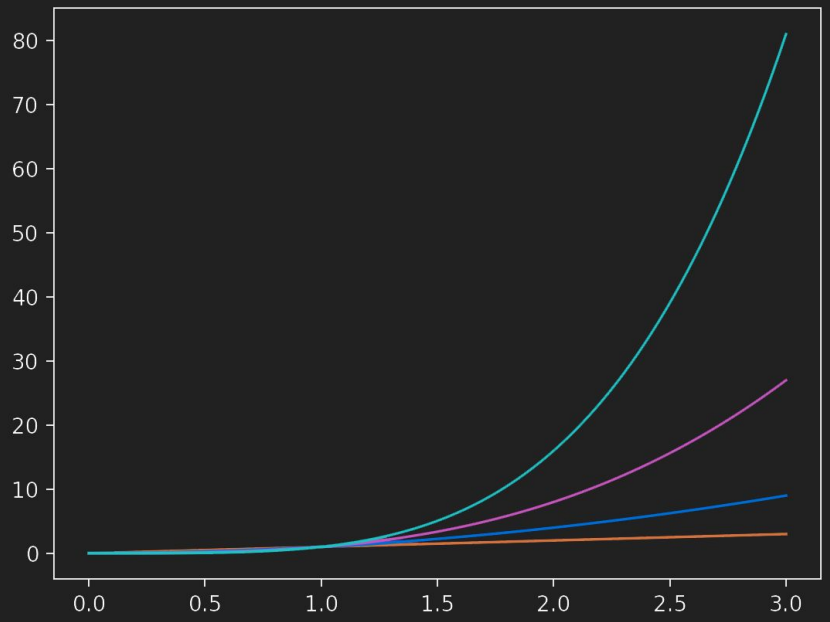
# Multiple curves?????

```
x = np.linspace(0,2*np.pi,100)

plt.plot(x, np.sin(x))

plt.plot(x, np.cos(x))

plt.show()
```



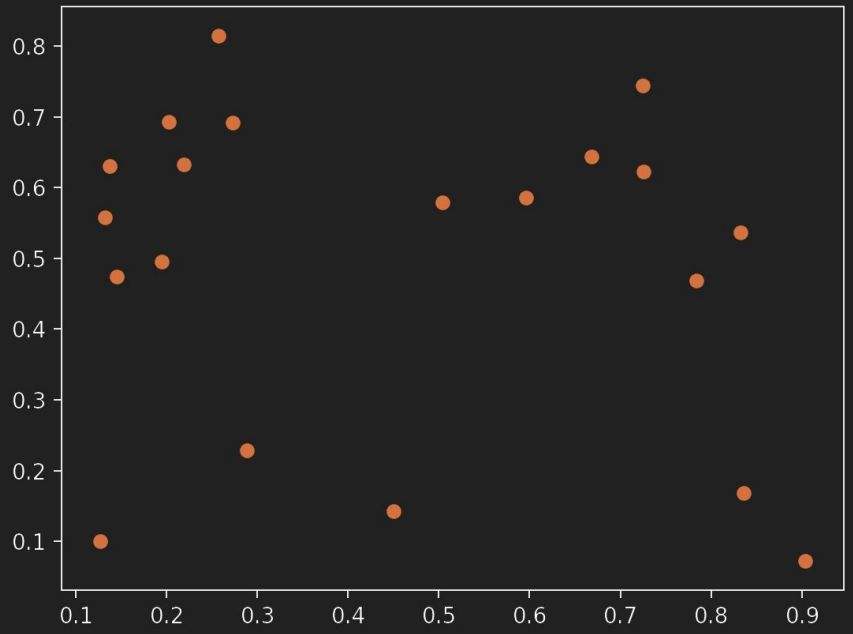
# Say… I want four curves…

```
x = np.linspace(0,3,100)

for p in [1,2,3,4]:

    plt.plot(x, x**p)

plt.show()
```



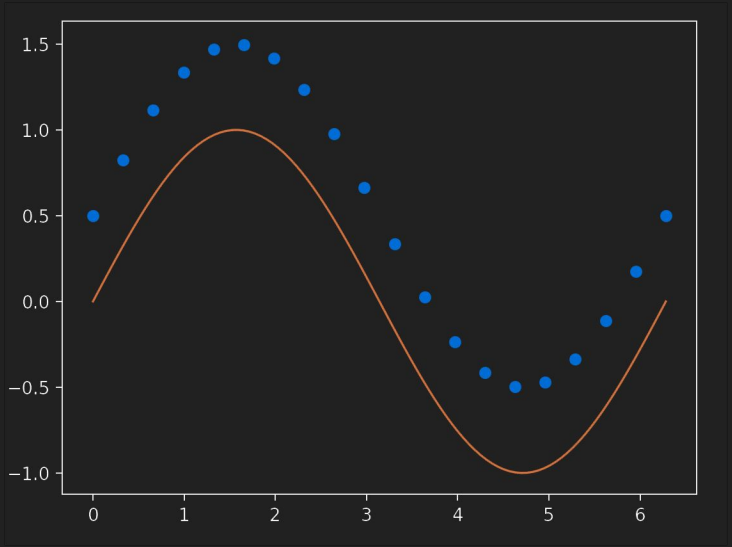
# Plotting points? Scatter plots!!!

```
x = np.random.rand(20)

y = np.random.rand(20)

plt.scatter(x,y)

plt.show()
```



# Different types of plot in one?

```
XX = np.linspace(0,2*np.pi,100)

YY = np.sin(XX)

x = np.linspace(0,2*np.pi,20)

y = np.sin(x) + 0.5

plt.plot(XX, YY)

plt.scatter(x,y)

plt.show()
```



# “ASTRONOMERS HAVE THE BEST PLOTS!”

```
plt.scatter(x, y, c=‘red’, s=1, alpha=0.8, marker=‘*’)
```

colour

size

opacity
[0, 1]

Marker style

```
plt.xlabel(“x”)

plt.ylabel(“y”)
```

AXES -- more advanced , more flexible (ask us)

COLOUR-BLIND FRIENDLY PLOTS: https://personal.sron.nl/~pault

# Wednesday


https://forms.gle/M6Kh8bFcNkZJjB5N7

# Recap

- How to plot lines (functions)
- How to plot scatter plots
- Overlaying on one single figure

# Breakout Room Question

What would we see if I wrote the following code?

(assuming we already wrote x, f(x), and g(x) earlier in the code)

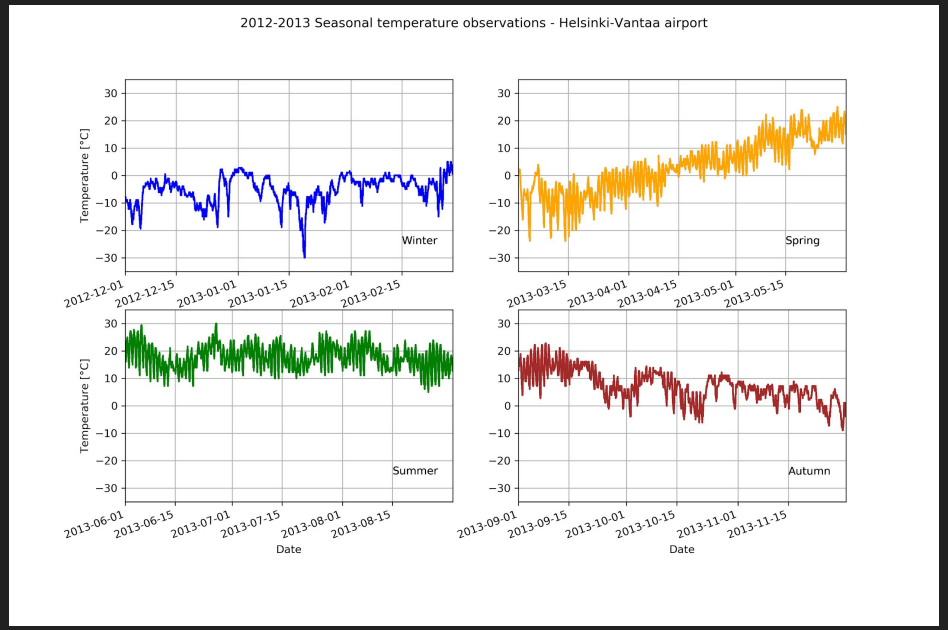
```
plt.plot(x, f(x))

plt.show()



plt.plot(x, g(x))

plt.show()
```

# More Advanced Plotting

- Subplots
- 3 dimensional plots
- Contour plots
- Histograms
- Polar graphs
- Etc…

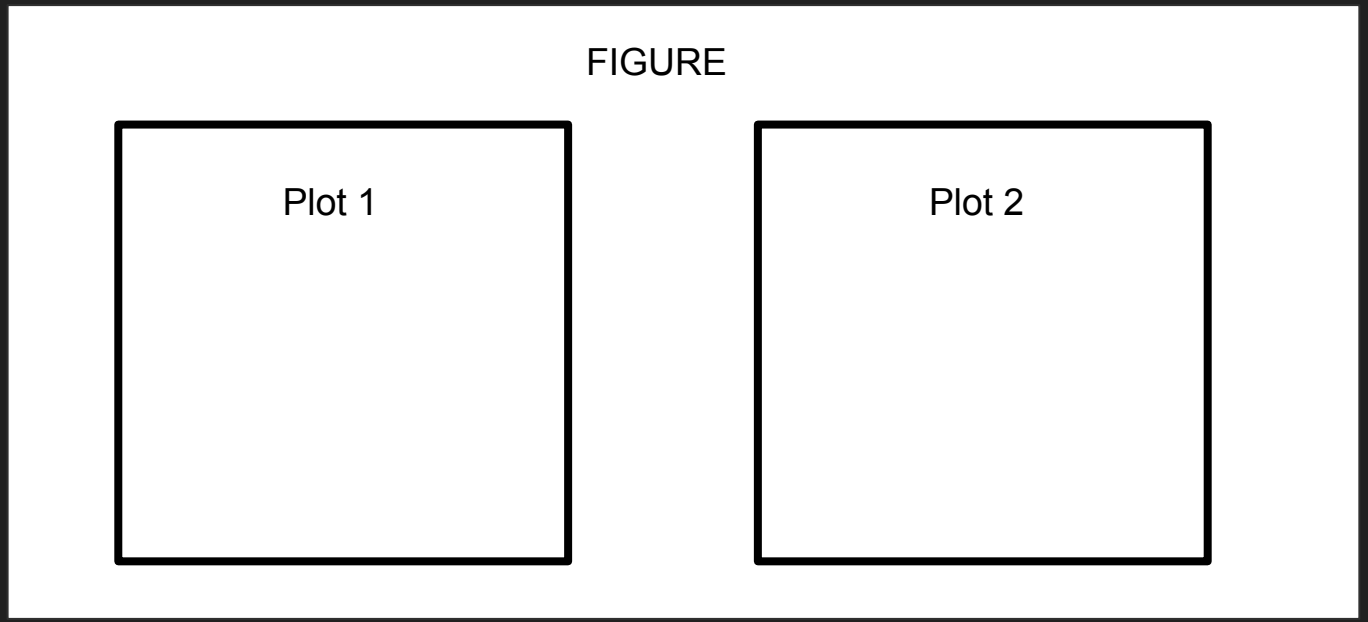
# Subplots

- In order to make these we need to understand what is happening behind the scenes



# Structure

- Matplotlib makes a figure on the screen, then adds a plot(s) onto the figure
- Like a collage of sub pictures



# Approaches

State - Machine Approach

```
plt.figure()

plt.plot(x, f(x))

plt.show()
```

Allows us to make single plots quickly without worrying about syntax as much

Object Oriented Approach

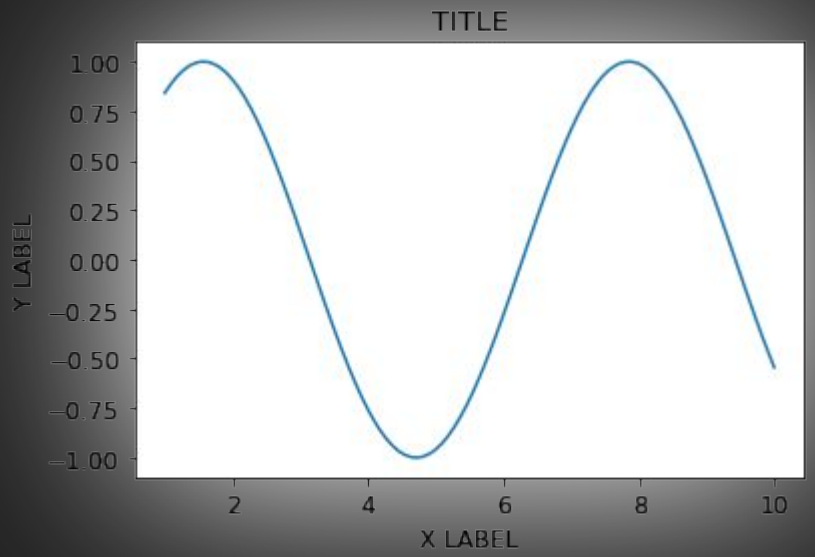
```
fig, ax = plt.subplots(1,1)

ax.plot(x, f(x))

plt.show()
```

Allows us to make multiple plots efficiently once created, slightly different syntax

# Subplot Syntax

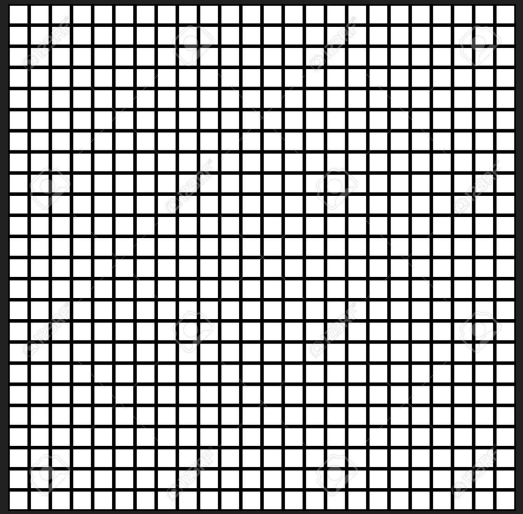
```
x = np.linspace(1,10,100)

y = np.sin(x)

fig, ax = plt.subplots(1,1)

ax.plot(x, y)

ax.set_title('TITLE')

ax.set_xlabel('X LABEL')

ax.set_ylabel('Y LABEL')

plt.show()
```

Creates an axis object



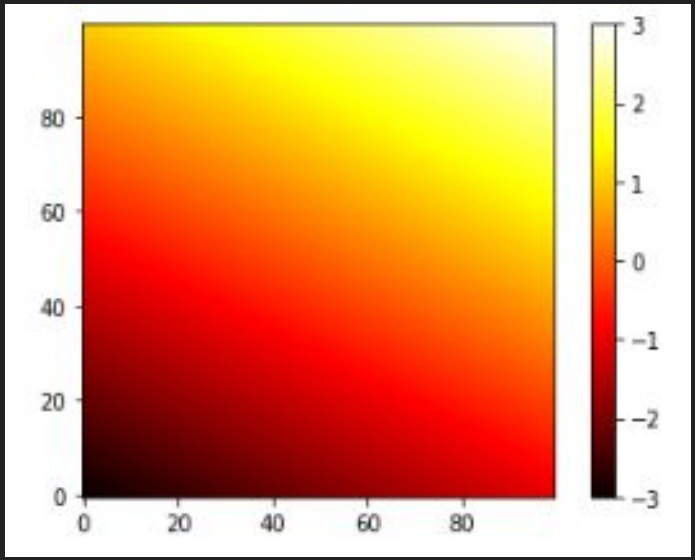
# 3D Plots

- Slightly different, now we use `plt.imshow(matrix)`
- This allows us to plot a 2x2 array with values, like a grid



# Plotting 3D Functions with `plt.imshow()`

```
x, y = np.meshgrid(np.linspace(-1, 1, 100),
                   np.linspace(-2, 2, 100))

f  = x + y

plt.figure()

plt.imshow(f, cmap= ____, origin = ____, ....)

plt.colorbar()

plt.show()
```

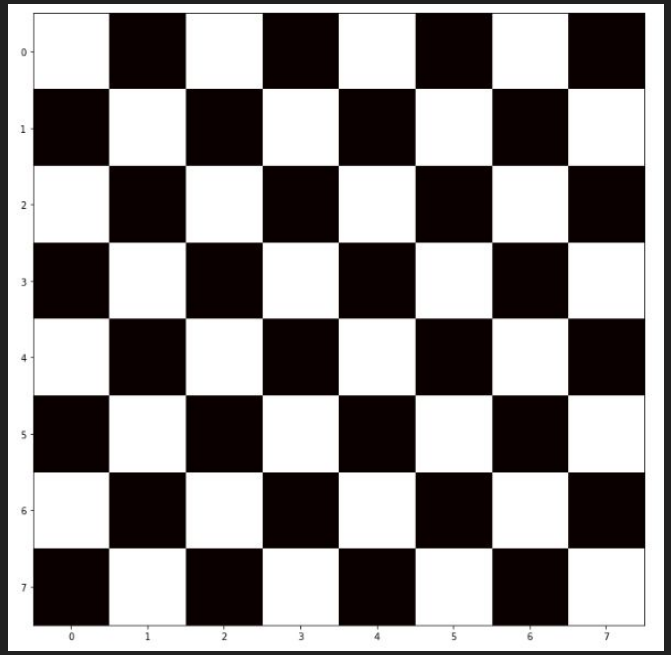


# Plotting 3D Data with `plt.imshow()`

```
checkerboard_data = np.loadtxt('DataFile.csv', delimiter=',', unpack=False)

plt.figure(figsize=(12,12))

plt.imshow(checkerboard_data)

plt.show()
```

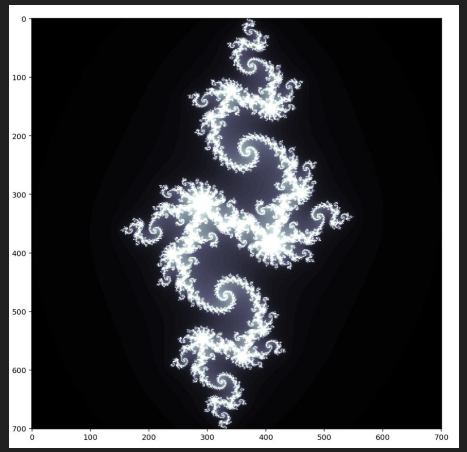
```
1, 0, 1, 0, 1, 0, 1, 0
0, 1, 0, 1, 0, 1, 0, 1
1, 0, 1, 0, 1, 0, 1, 0
0, 1, 0, 1, 0, 1, 0, 1
1, 0, 1, 0, 1, 0, 1, 0
0, 1, 0, 1, 0, 1, 0, 1
1, 0, 1, 0, 1, 0, 1, 0
0, 1, 0, 1, 0, 1, 0, 1
```

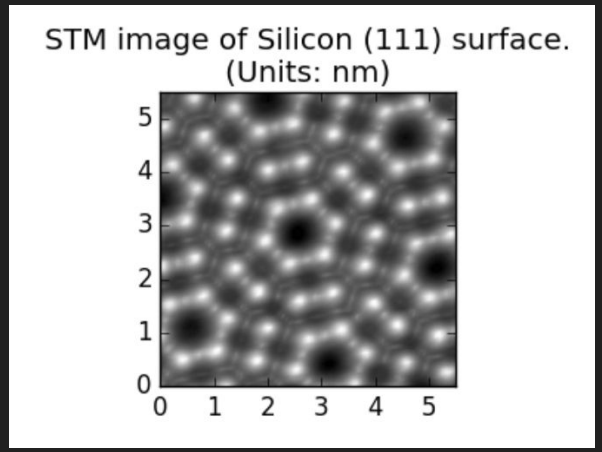
**DataFile**

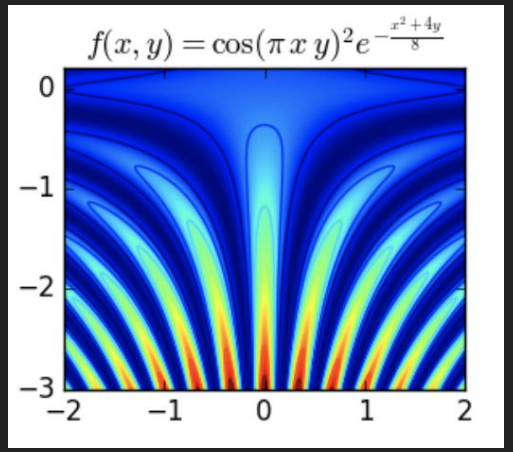


# 3D Plots

- You can make some really cool stuff like this



# Ultimate Plotting Guide

# An interesting way you can use this: