
Python DeCal
Week 5

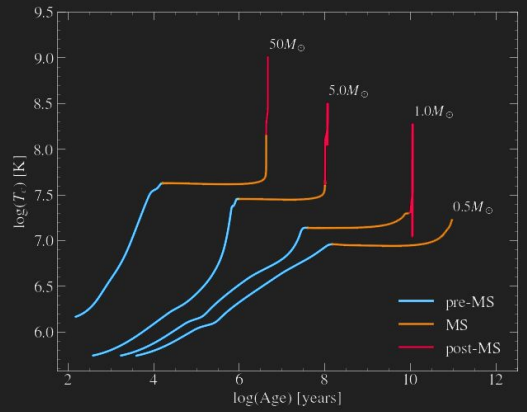
Announcements

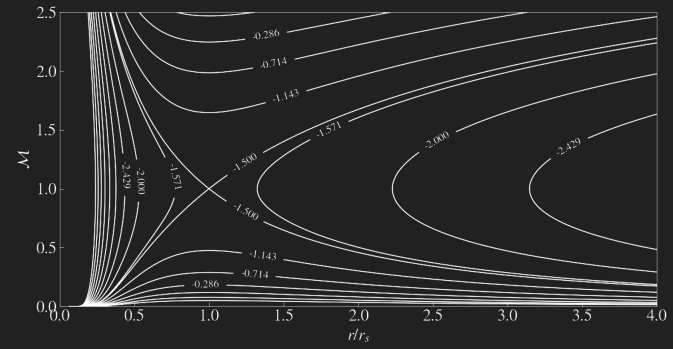
- 3rd Hw was due just now!
- Office Hour
  - Thanks to those who showed up!!
- Attendance!
  - https://tinyurl.com/spring-lol

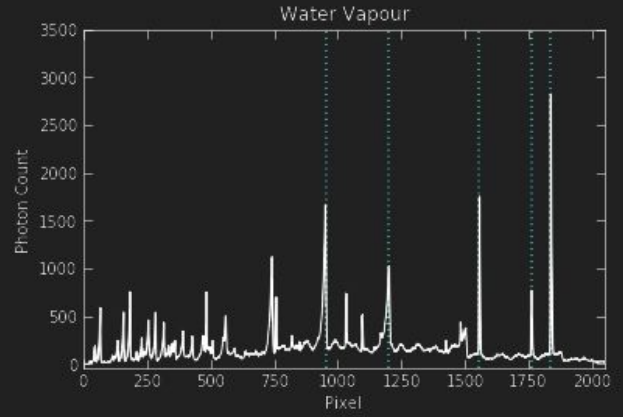
Recap

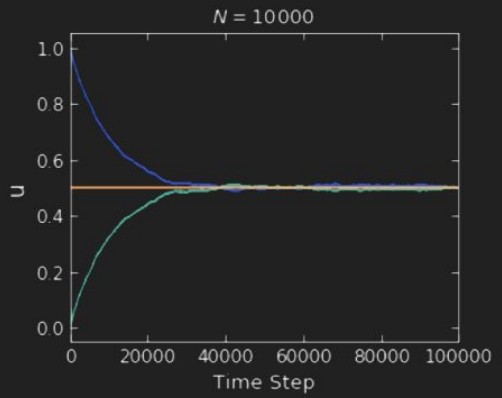
- What does `np.where()` do?
- What is the difference between `np.arange()` and `np.linspace()`

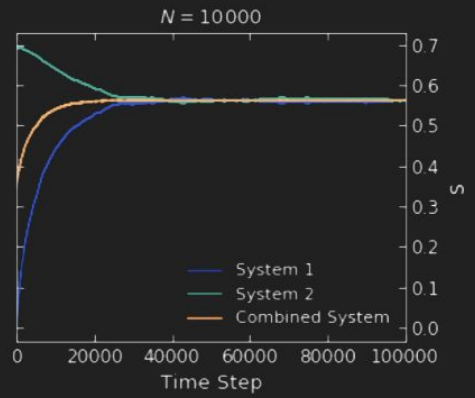
How do we represent data after analysing them...







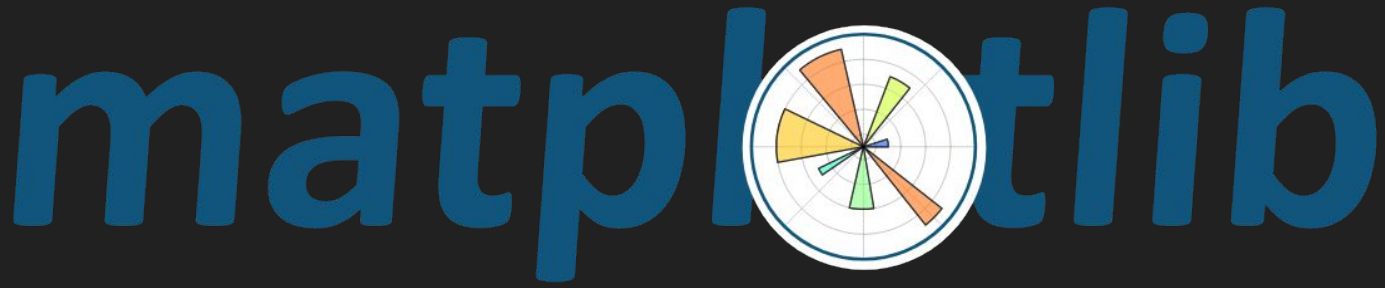




What package to import?

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

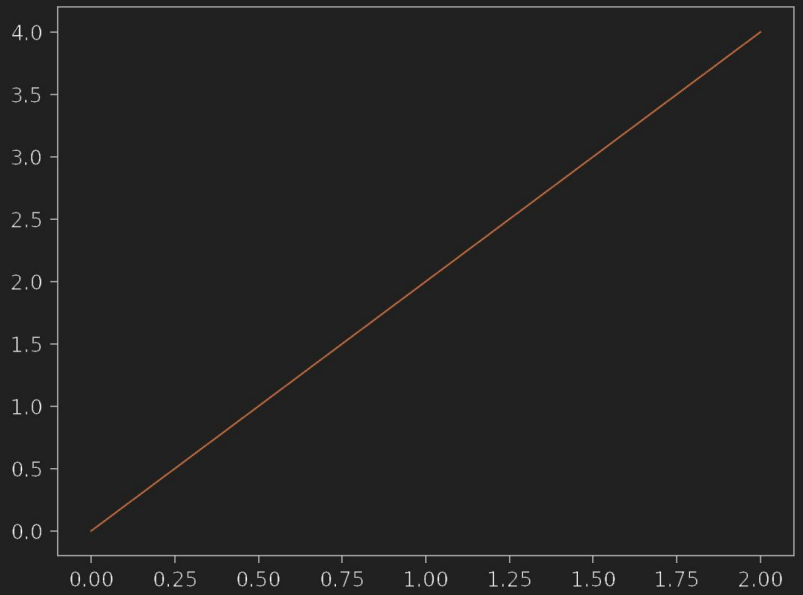
Whatever type of research you do, you will need this!



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

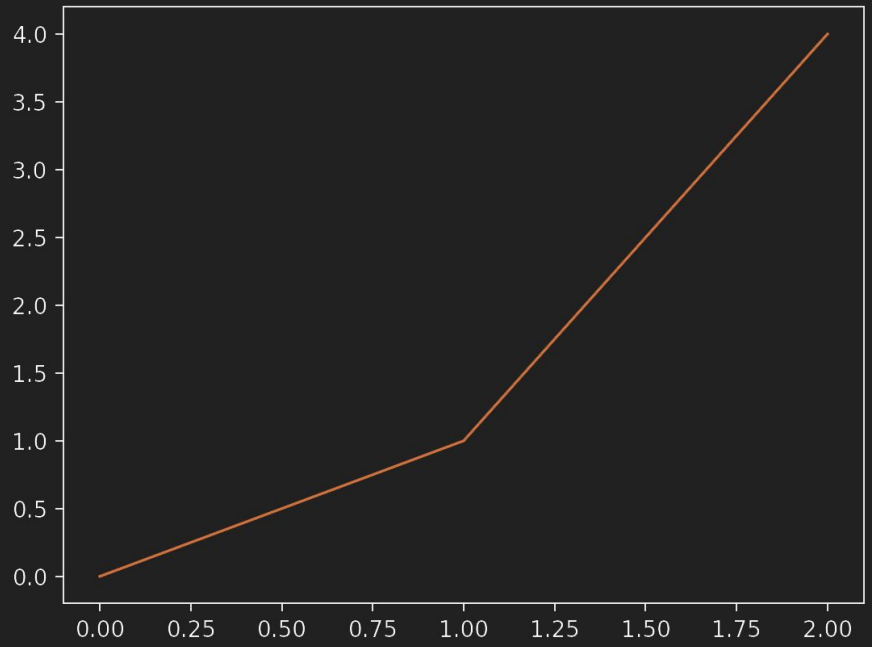
```
plt.plot(arr)

plt.plot(x_arr, y_arr)
```

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

>>> y = 2*x

>>> plt.plot(x, y)

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



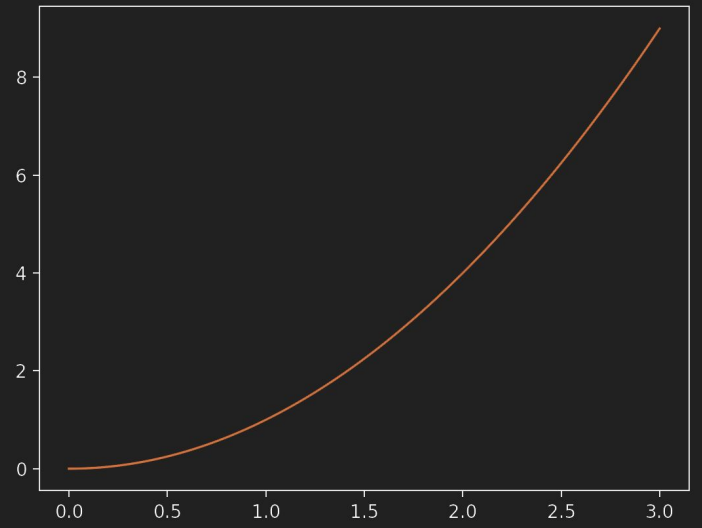
Now, what about a curve?

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



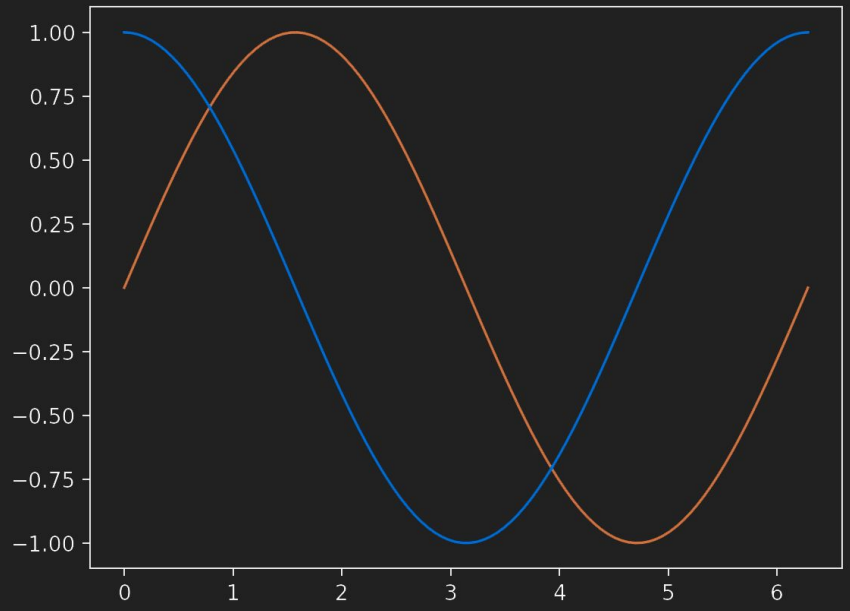
Now, what about a curve?

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



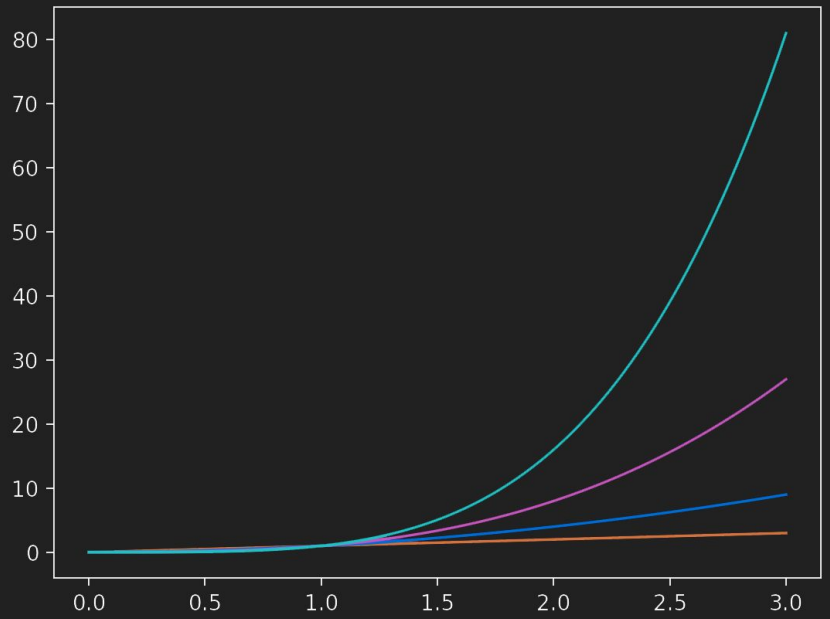
Multiple curves?????

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

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

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

plt.show()
```



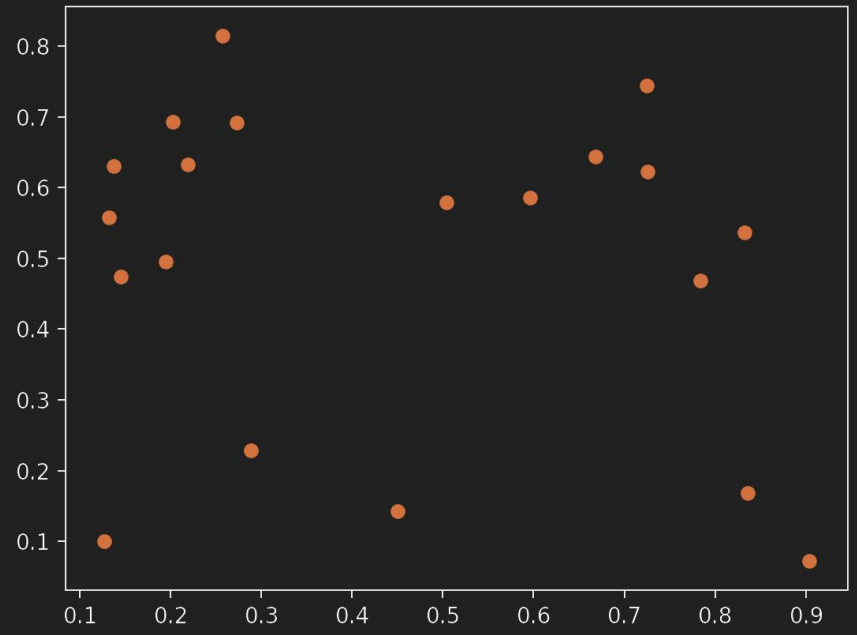
Say… I want four curves…

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

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

    plt.plot(x, x**p)

plt.show()
```



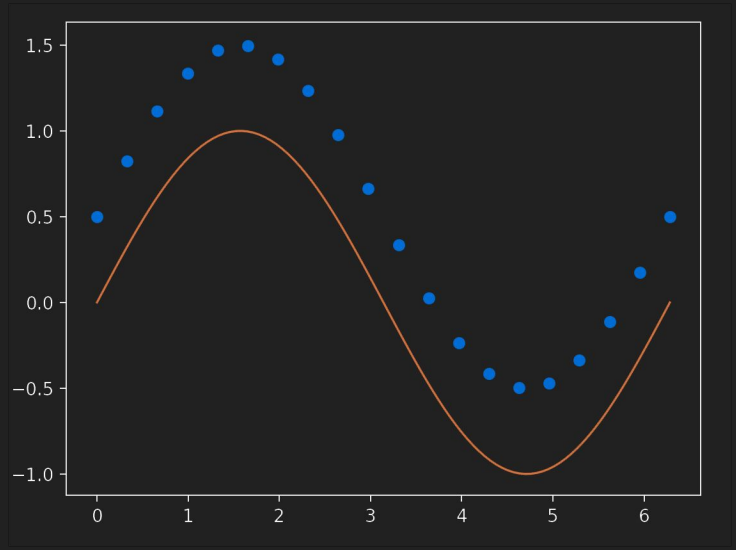
Plotting points? Scatter plots!!!

```
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!”

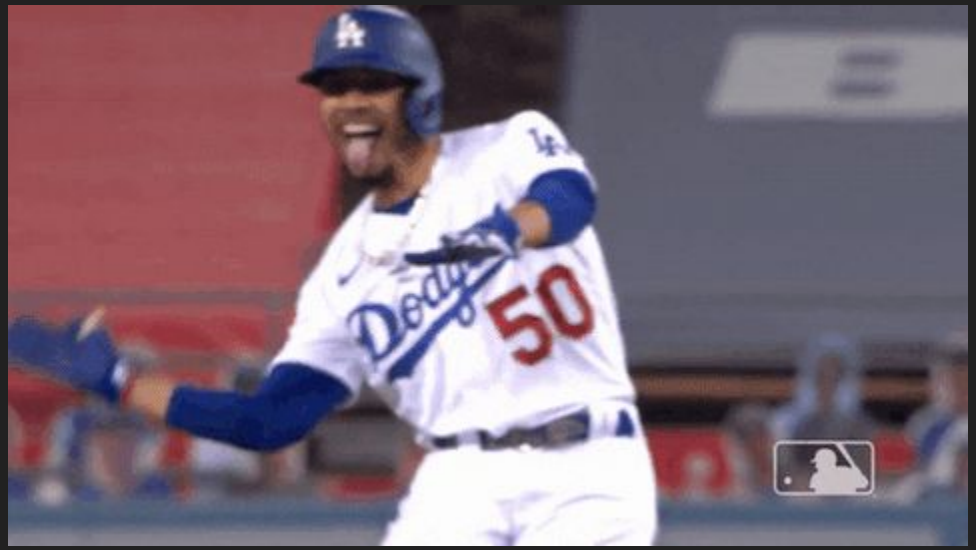
```
plt.scatter(x, y, c='red', s=1, alpha=0.8, marker='*')
```


```
plt.xlabel("x")

plt.ylabel("y")
```

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

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

FRIDAY



https://forms.gle/C2HaD3bDixQXp3tx6

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)

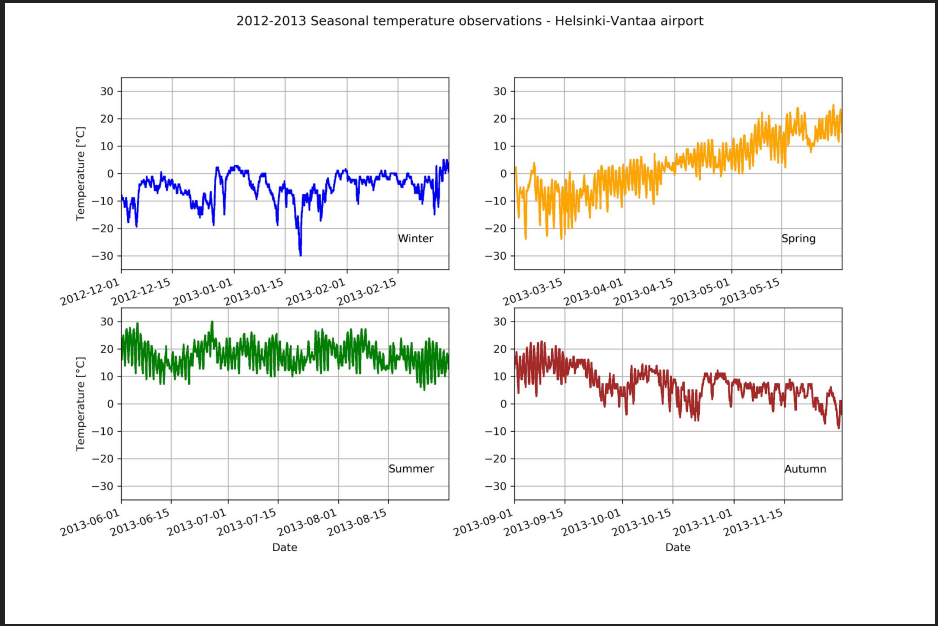
```
plt.plot(x, f(x))

plt.show()


plt.plot(x, g(x))

plt.show()
```

More Advanced Plotting

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

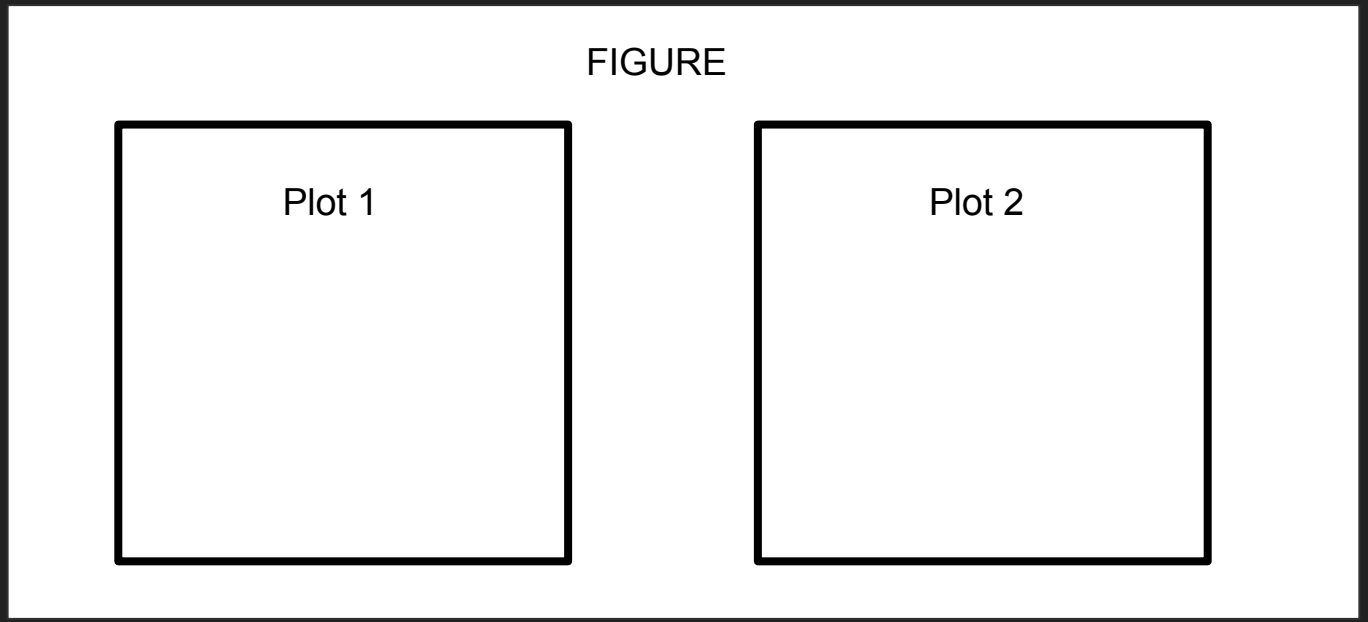
Subplots

- 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

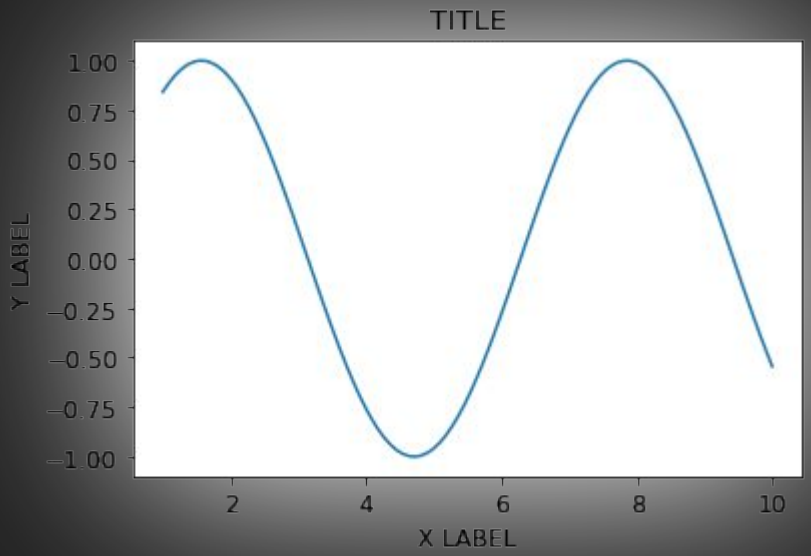
```
fig, ax = plt.subplots(1,1)

ax.plot(x, f(x))

plt.show()
```

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

Subplot Syntax

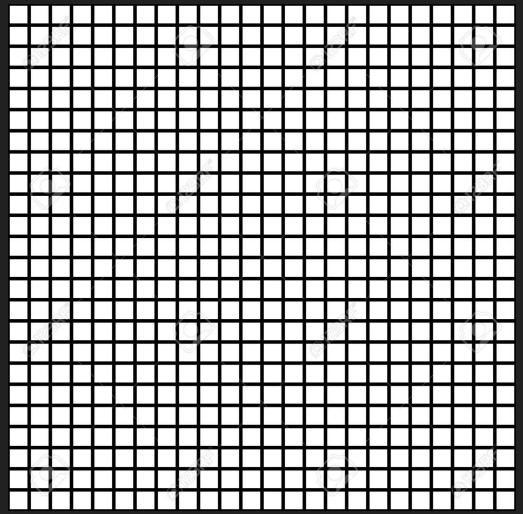
```
x = np.linspace(1,10,100)

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



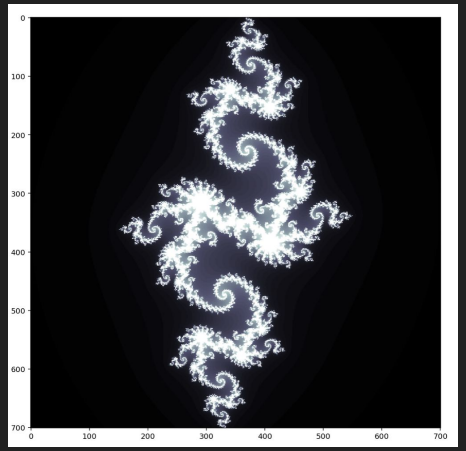
3D Plots

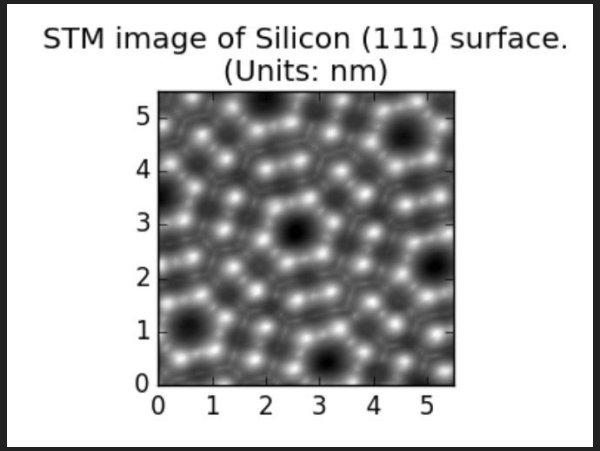
- Slightly different, now we use `plt.imshow(matrix)`
- This allows us to plot a 2x2 array with values, like a grid

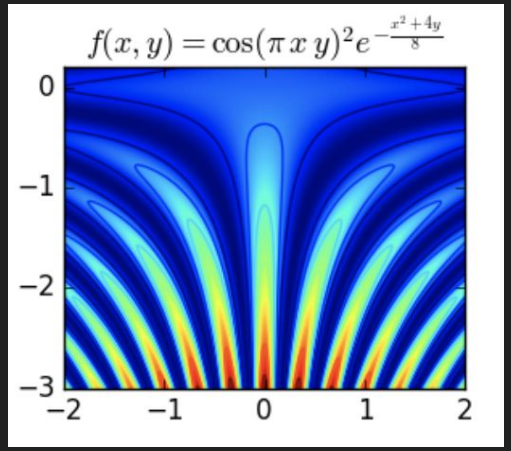


3D Plots

- You can make some really cool stuff like this







Ultimate Plotting Guide

An interesting way you can use this: