Python DeCal

Week 4

Announcements

- 2nd Hw was due just now!
- Office Hour
  - Thanks to those who showed up!!
- Attendance!
  - https://tinyurl.com/naitomea-akumu

Recap

- What is recursion?
- How are dictionaries different from lists?

How to make our code more POWERFUL?

- `import` statements:
  - math, random, numpy…
- You can import packages and libraries so that you don't have to write your own functions
  - How would you calculate the median of a list of a 10 unordered numbers?

How do you import?

- `import package_name as alias:`

- For example, let’s talk about NUMPY today!
    - `import numpy as np`
- Generally put at the very beginning of your code

Numpy Arrays (1D)

- The most used tool in science

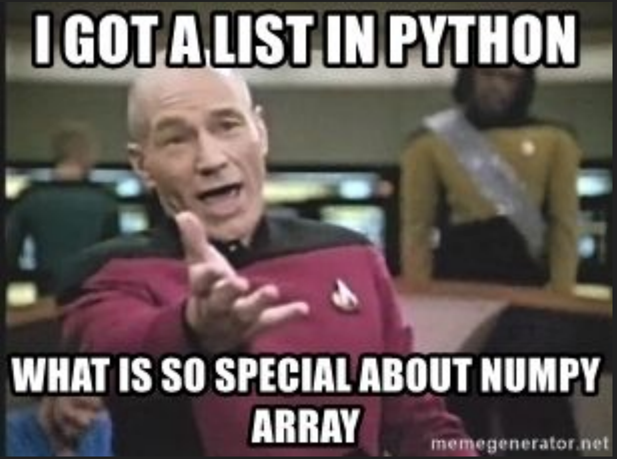
- This is a list

```
l = [0, 1, 2, 3, 4, 5, 6, 7]
```

- This is a numpy array

```
arr = np.array([0, 1, 2, 3, 4, 5, 6, 7])
```

- Same indexing rule as lists



IT IS SUPERIOR!!!!!!!!

Some review: What do the following commands do?

```
list_1 = [0, 1, 2, 3, 4, 5, 6, 7]

list_2 = [0, 1, 2, 3, 4, 5, 6, 7]



3*list1 → ???

list_1 + list_2 → ??????
```

IT IS SUPERIOR!!!!!!!!

HOWEVER, THIS IS WHAT NUMPY ARRAY DOES!

```
    arr1 = np.array([0, 1, 2, 3, 4, 5, 6, 7])

    arr2 = np.array([0, 1, 2, 3, 4, 5, 6, 7])
```

Numpy arrays operate element-wise!

```
    3*arr1 → np.array([0, 3, 6, 9, 12, 15, 18, 21])

    arr1 + arr2 → np.array([0, 2, 4, 6, 8, 10, 12, 14])
```

Notice that the size of the two arrays have to be the same

What else can I do with 1D arrays?

Calculating the mean / average value:

```
    sum(your_list)/len(your_list)

    np.mean(your_array)
```

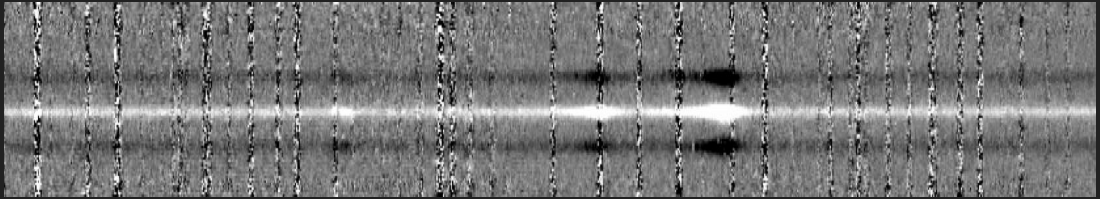
Calculating standard deviation (San Diego!!!)

```
    np.std(your_arr)      np.percentile(your_arr, percentile)
```

Calculating median value:

```
    np.median(your_arr)
```

2D Array!



You can think of it as a matrix, sometimes a table

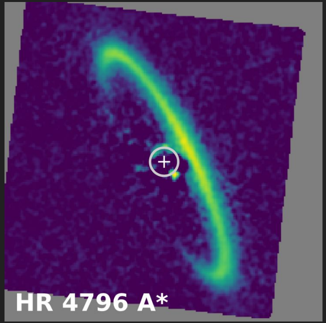
```
    np.array([[1, 2],
               [2, 1]]
```

$$\begin{bmatrix} 1 & 2 \\ 2 & 1 \end{bmatrix}$$

You can store different information in a 2D-array

```
    np.array([[x1, x2, x3, x4, x5],
              [y1, y2, y3, y4, y5]
```

Many astronomical datasets are 2D arrays! (images, spectra)



Slicing & Indexing (you’ll use this almost everyday)

```
arr = np.array([[1, 2, 3],
                 [2, 1, 3],
                 [3, 2, 1]])
```

$$\begin{array}{cc} \text{Axis 0} & \begin{bmatrix}1 & 2 & 3\\ 2 & 1 & 3\\ 3 & 2 & 1\end{bmatrix} \\ & \text{Axis 1} \end{array}$$

Of course, we can have more axes.

Slicing & Indexing (you'll use this almost everyday)

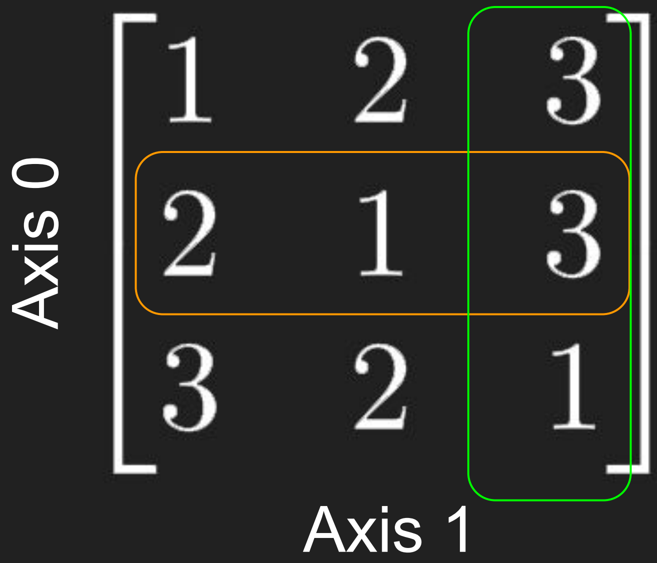
```
>>> arr[1, 2]
    3
```

Index along axis 1

Index along axis 0

Everything along an axis

```
>>> arr[:, 2]
    np.array([3, 3, 1])
```


$$\begin{array}{cc} & \\ \text{Axis 0} & \begin{bmatrix} 1 & 2 & 3 \\ 2 & 1 & 3 \\ 3 & 2 & 1 \end{bmatrix} \\ & \text{Axis 1} \end{array}$$

Now I want the middle row….

“Axes-wise Operations”

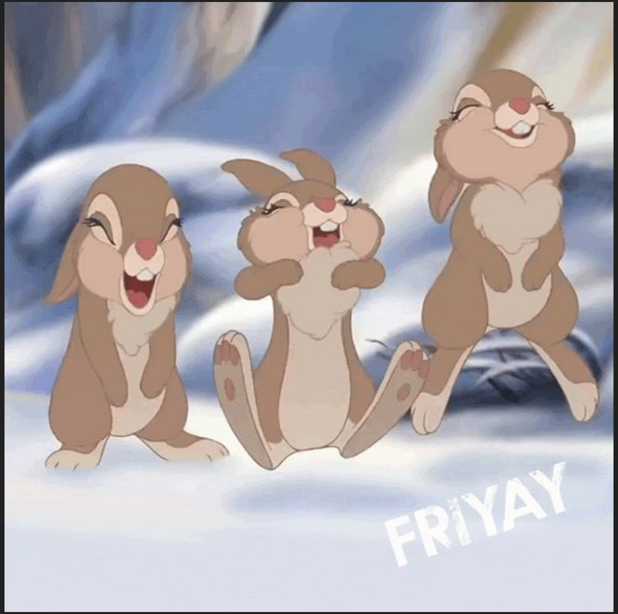
```
>>> np.mean(arr, axis=0)
    np.array([2,1.67,2.33])


>>> np.median(arr,axis=1)
    np.array([2, 2, 2])
```

$$\text{Axis 0}\begin{bmatrix}1 & 2 & 3\\ 2 & 1 & 3\\ 3 & 2 & 1\end{bmatrix}$$

Axis 1

Attendance Form: https://tinyurl.com/extragalactic-exoplanet

FRIDAY



Recap

- Numpy
  - What is it?
  - Numpy array

Breakout Room

- I have a 2D array

```
Example_array = np.array([[1, 2],
                          [2, 1]]
```

- How do I extract the everything along axis 0? Axis 1?

Numpy stats functions

Sometimes we want to directly know some information about the array we have.

- Order Statistics
- Averages and Variances

The Basics

- `numpy.max()`

- `numpy.min()`

Assumes a 1D Array or a List

Order Statistics for Higher Dimensional Arrays

These functions will give us some insight on the range and distribution of the data.

```
numpy.amin outputs the minimum along a specified axis

numpy.amax outputs the maximum along a specified axis
```

Order Statistics Demo

```
numpy.amin outputs the minimum along a specified axis

a = np.array([[1, 2],

              [2, 1]]

np.amin(a)

np.amin(a, axis=0)
```

Order Statistics Demo

```
numpy.amax outputs the maximum along a specified axis

a = np.array([[1, 2],

              [2, 1]]

np.amax(a)

np.amax(a, axis=0)
```

Averages and Variances

`numpy.median`

`numpy.mean (arithmetic mean)`

`numpy.std`

`numpy.var`

Create array with specific range/number of items

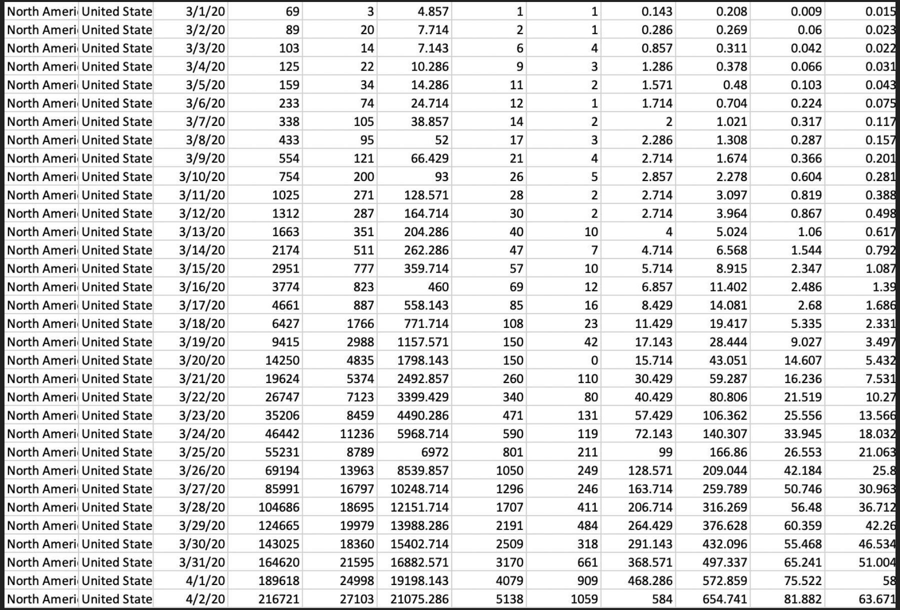
```
numpy.arange([start], stop, [step])

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

How do we get the 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])
```

Homework Time

Correlating

These functions will give us some insight on the correlations

`numpy.amin`

`numpy.amax`

`Do you want the correlations?`

`Idk what they are but they are on the numpy stat functions`

Histograms

These functions will compute the histogram of a given set of data

```
numpy.histogram

numpy.histogram2d

numpy.histogramdd

numpy.bincount

numpy.histogram_bin_edges

numpy.digitize
```