NumPy/Pandas

CAC 350

Catch Up

* Green screen

* Please take reading quiz if you haven't already done so

* Please submit assignment from last week if you haven't already done so

Today

* Wrap up a few things from NumPy
* Start working with Pandas

Broadcasting

* Ufuncs allow us to vectorize arrays - meaning we operate on the set or vector rather than one item at a time (hence, no loops)

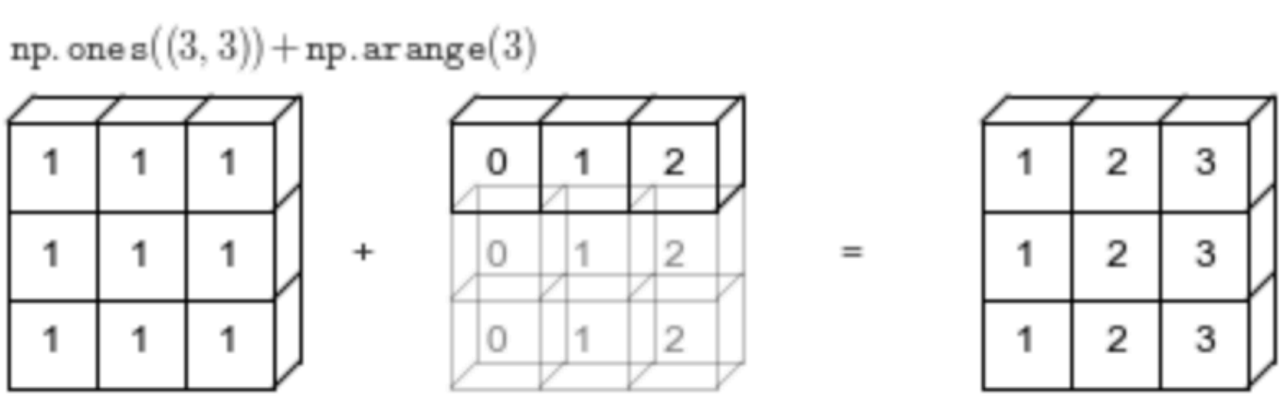
* Similarly, we can use broadcasting to apply binary ufuncs on arrays of different sizes

* The smaller array (or scalar) is broadcast or stretched to match the dimensions of the larger array

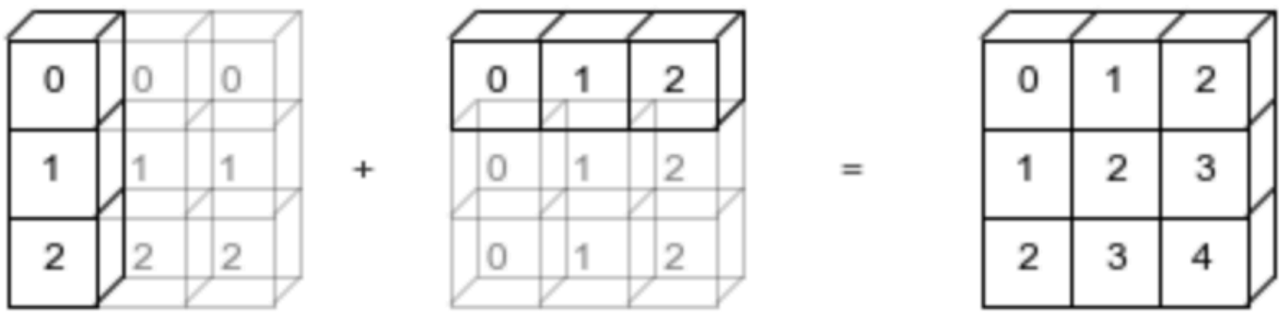
Example

$\texttt{np.arange}(3)+5$

$\texttt{np.ones}((3,3))+\texttt{np.arange}(3)$



$\texttt{np.arange}(3).\texttt{reshape}((3,1))+\texttt{np.arange}(3)$



Broadcasting Rules

1. if the two arrays differ in their number of dimensions, the shape of the one with fewer dimensions is padded with ones on its leading (left) side

2. if the shape of the two arrays does not match in any dimension, the array with shape equal to 1 in that dimension is stretched to match the other shape

3. if in any dimension the sizes disagree and neither is equal to 1, an error is raised

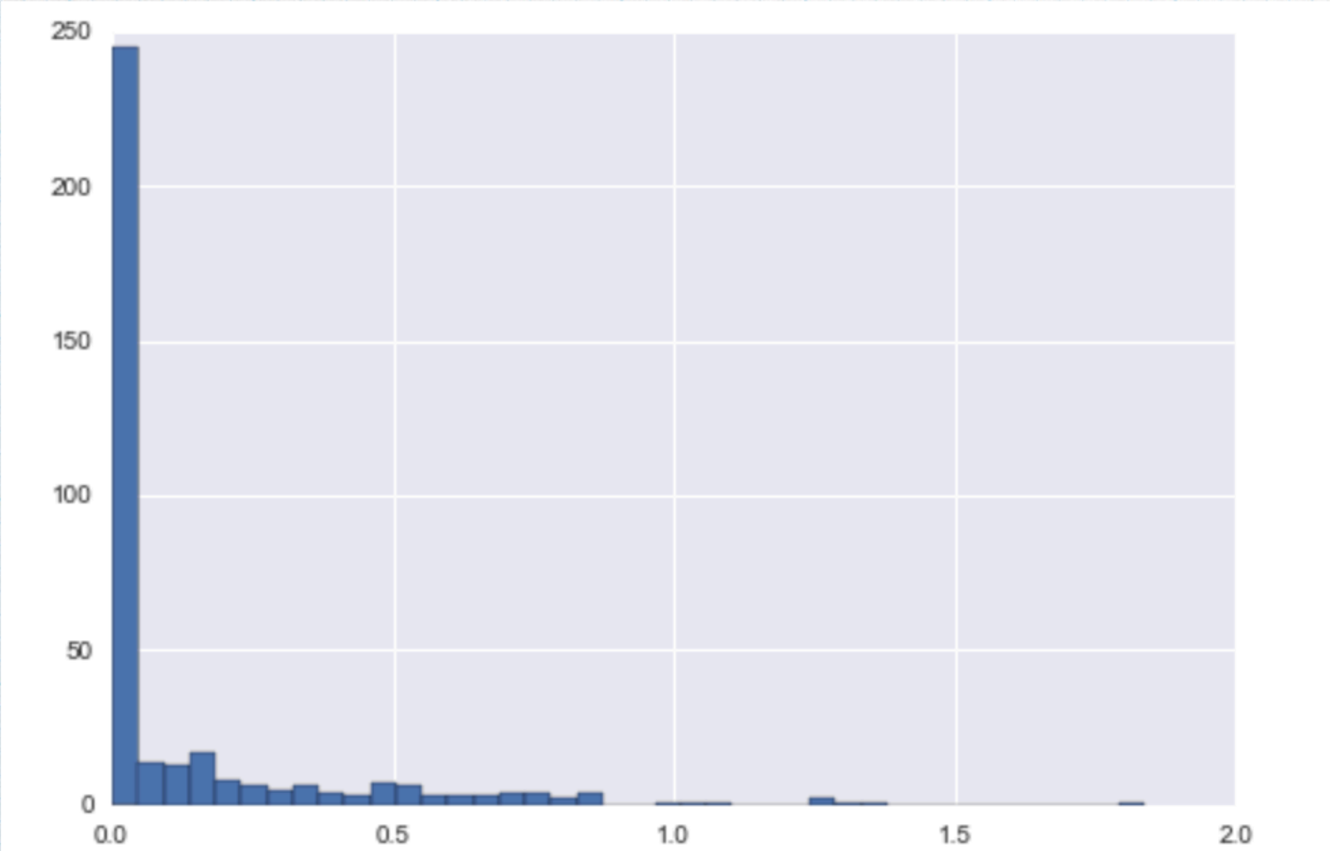
jupyter

Why Use Broadcasting?

* Centering an array
* Plotting a 2D function

Comparisons, Masks, and Boolean Logic

* Daily rainfall in inches



* How many rainy days were there in a year?
* What is the avg precipitation on those days?
* How many days were there with more than half an inch of rain?

Aggregate Functions with Boolean Expressions

In [25]:
```
print("Number days without rain:      ", np.sum(inches == 0))
print("Number days with rain:         ", np.sum(inches != 0))
print("Days with more than 0.5 inches:", np.sum(inches > 0.5))
print("Rainy days with < 0.2 inches  :", np.sum((inches > 0) &
                                                (inches < 0.2)))
```

```
Number days without rain:       215
Number days with rain:          150
Days with more than 0.5 inches: 37
Rainy days with < 0.2 inches  : 75
```

Masking

* Actually selecting the values we want from the boolean array

```
In [29]: # construct a mask of all rainy days
         rainy = (inches > 0)

         # construct a mask of all summer days (June 21st is the 172nd day)
         days = np.arange(365)
         summer = (days > 172) & (days < 262)

         print("Median precip on rainy days in 2014 (inches):   ",
               np.median(inches[rainy]))
         print("Median precip on summer days in 2014 (inches):  ",
               np.median(inches[summer]))
         print("Maximum precip on summer days in 2014 (inches): ",
               np.max(inches[summer]))
         print("Median precip on non-summer rainy days (inches):",
               np.median(inches[rainy & ~summer]))
```

```
Median precip on rainy days in 2014 (inches):    0.194881889764
Median precip on summer days in 2014 (inches):   0.0
Maximum precip on summer days in 2014 (inches):  0.850393700787
Median precip on non-summer rainy days (inches): 0.200787401575
```

Pandas

* Pandas objects are enhanced NumPy arrays
  * Series
  * DataFrame
  * Index

Series

* One-dimensional array of indexed data

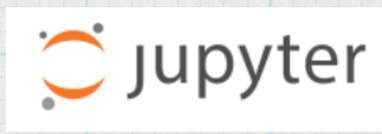
```
In [2]: data = pd.Series([0.25, 0.5, 0.75, 1.0])
        data

Out[2]: 0    0.25
        1    0.50
        2    0.75
        3    1.00
        dtype: float64
```

* So, what's the difference between a NumPy array and a Pandas Series?
* More like a dictionary but better, we can slice

jupyter

DataFrame

* Essentially a two-dimensional array with both flexible row indices and column names
* Can construct multiple way:
  * Single series
  * Two series
  * List of dictionaries
  * 2D NumPy Array

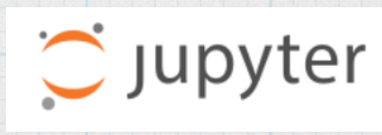


Index

* Immutable array - avoids side effects
* Can be used to slice data frame

Let's Look at Data...

Grab the data set from Moodle



For Next Time

* Homework assignment
* Keep skimming Chapter 3: Pandas
* Start skimming Chapter 4: Visualization