**Numpy notes –** NumPy is great at storing and manipulating numerical data in arrays.

Let's take a look at an example. Twice Charred in a fictional (mostly) movie review site where four good friends and movie reviewers, Lorie, Marty, Tori, and Kurtz watch movies and give them ratings on a scale of 0 to 100.

In [1]:

*# Before we do anything, we need to import NumPy*

**import** **numpy** **as** **np**

When the gang rates a movie, we can store their ratings in a NumPy array movie\_ratings:

In [2]:

movie\_ratings = np.array([63.0, 54.0, 70.0, 50.0])

But they see more than one movie, so we have to create a 2-dimensional array where each row is their ratings for a specific movie.

In [3]:

movie\_ratings = np.array([[63.0, 54.0, 70.0, 50.0],

[94.0, 85.0, 89.0, 95.0],

[64.0, 90.0, 73.0, 85.0]])

Some fans prefer to have the movies rated on a five star scale, so we can use NumPy to easily divide each element by 20.

In [4]:

movie\_ratings\_stars = movie\_ratings / 20

Now let's say the ratings are always in the same order (Lorie, Marty, Tori, Kurtz) if we wanted to create an array that only had Tori's ratings, we could select that from our movie\_ratings array.

In [5]:

tori\_ratings = movie\_ratings[:, 2]

tori\_ratings

Out[5]:

array([ 70., 89., 73.])

Now, say we find that we have very similar taste to Marty, so we only want to see movies that he gives a good rating to, we can use logic to select those movies.

Let's select all of Marty's ratings that are over 80:

In [6]:

marty\_ratings = movie\_ratings[:, 1]

marty\_ratings[marty\_ratings > 80]

Out[6]:

array([ 85., 90.])

## Introduction to Statistics with NumPy

After the river in your town flooded during a recent hurricane, you've become interested in collecting data about the its height. Every day for the past month, you walk to the river, measure the height of the water, and enter this information into a notebook.

Let's look at how you can use NumPy functions to analyze your dataset.

First, we'll import the NumPy module, so we can use its statistical calculation functions.

In [10]:

**import** **numpy** **as** **np**

water\_height = np.array([4.01, 4.03, 4.27, 4.29, 4.19,

4.15, 4.16, 4.23, 4.29, 4.19,

4.00, 4.22, 4.25, 4.19, 4.10,

4.14, 4.03, 4.23, 4.08, 14.20,

14.03, 11.20, 8.19, 6.18, 4.04,

4.08, 4.11, 4.23, 3.99, 4.23])

Let's use the function np.mean() to find the average water height:

In [11]:

np.mean(water\_height)

Out[11]:

5.2510000000000003

But wait! We should sort our data to see if there could be any measurements to throw our data off, or represent a deviation from the mean:

In [12]:

np.sort(water\_height)

Out[12]:

array([ 3.99, 4. , 4.01, 4.03, 4.03, 4.04, 4.08, 4.08,

4.1 , 4.11, 4.14, 4.15, 4.16, 4.19, 4.19, 4.19,

4.22, 4.23, 4.23, 4.23, 4.23, 4.25, 4.27, 4.29,

4.29, 6.18, 8.19, 11.2 , 14.03, 14.2 ])

Looks like that thunderstorm might have impacted the average height! Let's measure the median to see if its more representative of the dataset:

In [13]:

np.median(water\_height)

Out[13]:

4.1900000000000004

While the median tells us where half of our data lies, let's look at a value closer to the end of the dataset. We can use percentiles to use a data points position and get its value:

In [14]:

np.percentile(water\_height, 75)

Out[14]:

4.2649999999999997

So far, we've gotten a good idea about specific values. But what about the spread of our data? Let's calculate the standard deviation to understand how similar or how different each data point is:

In [15]:

np.std(water\_height)

Out[15]:

2.784585367099861

Great! Just using a few simple functions we've been able to quickly calculate several important measurements and can begin analyzing our dataset.