In [15]:
```
import numpy as np
import pandas as pd
import scipy as sp
import string
```

In [2]:
```
%matplotlib inline
import matplotlib.pyplot as plt
plt.style.use('ggplot')
```

In [3]:
```
%%file hw_data.csv
id,sex,weight,height
1,M,190,77
2,F,120,70
3,F,110,68
4,M,150,72
5,O,120,66
6,M,120,60
7,F,140,70
```

```
Writing hw_data.csv
```

# Python

## 1. Finish creating the following function that takes a list and returns the average value.

In [13]:
```
def average(my_list):
    total = 0
    for item in my_list:
        total += item

    return total/len(my_list)

average([1,2,1,4,3,2,5,9])
```

Out[13]:
```
3.375
```

## 2. Using a Dictionary keep track of the count of numbers (or items) from a list

In [17]:
```
def counts(my_list):
    counts = dict()
    for item in my_list:
        counts[item] = my_list.count(item)

    return counts

counts([1,2,1,4,3,2,5,9])
```

Out[17]:
```
{1: 2, 2: 2, 3: 1, 4: 1, 5: 1, 9: 1}
```

## 3. Using the `counts()` function and the `.split()` function, return a dictionary of most occuring words from the following paragraph. Bonus, remove punctuation from words.

In [94]:
```
paragraph_text = '''
For a minute or two she stood looking at the house, and wondering what to do
The Fish-Footman began by producing from under his arm a great letter, nearl
Then they both bowed low, and their curls got entangled together.
Alice laughed so much at this, that she had to run back into the wood for fe
Alice went timidly up to the door, and knocked.
‘There’s no sort of use in knocking,’ said the Footman, ‘and that for two re
‘Please, then,’ said Alice, ‘how am I to get in?’
‘There might be some sense in your knocking,’ the Footman went on without at
‘I shall sit here,’ the Footman remarked, ‘till tomorrow—’
At this moment the door of the house opened, and a large plate came skimming

for c in string.punctuation:
    paragraph_text = paragraph_text.replace(c,"")

words = paragraph_text.split()
word_count = counts(words)
most_words = {k:v for k,v in word_count.items() if v > 5}
most_words
```

Out[94]:
```
{'a': 15,
 'and': 16,
 'at': 6,
 'door': 6,
 'in': 7,
 'of': 9,
 'she': 6,
 'the': 32,
 'to': 15,
 'was': 8}
```

## 4. Read in a file and write each line from the file to a new file Title-ized

`This is the first line -> This Is The First Line`

Hint: There's a function to do this

In [39]:
```
!printf 'The only winning move is not to play.\n source: warGames movie\n' >

with open('wargames.txt') as f:
    lines = f.readlines()

for x in [0,1]:

    file = 'wg_quote' + str(x) + '.txt'
    wargames = open(file,'w')
    wargames.write(lines[x].title())
    wargames.close()
```

# Numpy

## 1. Given a list, find the average using a numpy function.

In [40]:
```
simple_list = [1,2,1,4,3,2,5,9]

np.average(simple_list)
```

Out[40]: 3.375

## 2. Given two lists of Heights and Weights of individual, calculate the BMI of those individuals, without writing a `for-loop`

In [42]:
```
heights = [174, 173, 173, 175, 171]
weights = [88, 83, 92, 74, 77]
np.divide(weights, heights)
```

Out[42]: array([ 0.50574713,  0.47976879,  0.53179191,  0.42285714,  0.4502924 ])

## 3. Create an array of length 20 filled with random values (between 0 to 1)

In [43]:
```
np.random.random(size=20)
```

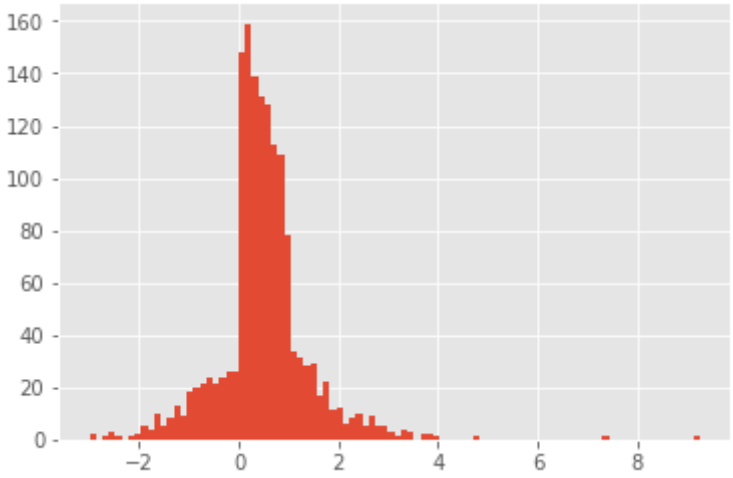
Out[43]:
```
array([ 0.49142164,  0.91854234,  0.01094909,  0.64581187,  0.00593558,
        0.58393419,  0.84781151,  0.65994762,  0.70159116,  0.31911922,
        0.66390357,  0.47594286,  0.92910394,  0.70468871,  0.04010008,
        0.04036646,  0.84768847,  0.62944188,  0.54738822,  0.628266  ])
```

## Bonus. 1. Create an array with a large (>1000) length filled with random numbers from different distributions (normal, uniform, etc.). 2. Then, plot a histogram of these

**values.**

In [90]:
```
a = np.concatenate((np.random.exponential(size=500),np.random.random(size=50
plt.hist(a.flatten(),bins='auto')
```

Out[90]:
```
(array([   2.,    0.,    1.,    3.,    1.,    0.,    1.,    2.,    5.,
           4.,   10.,    5.,    8.,   13.,    9.,   18.,   20.,   21.,
          24.,   21.,   24.,   26.,   26.,  148.,  159.,  139.,  131.,
         128.,  113.,  109.,   78.,   34.,   31.,   28.,   29.,   17.,
          22.,   11.,   12.,    6.,    8.,   10.,    5.,    9.,    5.,
           5.,    3.,    1.,    4.,    3.,    0.,    2.,    2.,    1.,
           0.,    0.,    0.,    0.,    0.,    1.,    0.,    0.,    0.,
           0.,    0.,    0.,    0.,    0.,    0.,    0.,    0.,    0.,
           0.,    0.,    0.,    0.,    0.,    0.,    0.,    1.,    0.,
           0.,    0.,    0.,    0.,    0.,    0.,    0.,    0.,    0.,
           0.,    0.,    0.,    1.]),
 array([ -3.00331626e+00,  -2.87292802e+00,  -2.74253978e+00,
         -2.61215154e+00,  -2.48176329e+00,  -2.35137505e+00,
         -2.22098681e+00,  -2.09059857e+00,  -1.96021033e+00,
         -1.82982208e+00,  -1.69943384e+00,  -1.56904560e+00,
         -1.43865736e+00,  -1.30826912e+00,  -1.17788087e+00,
         -1.04749263e+00,  -9.17104390e-01,  -7.86716148e-01,
         -6.56327906e-01,  -5.25939664e-01,  -3.95551422e-01,
         -2.65163180e-01,  -1.34774938e-01,  -4.38669608e-03,
          1.26001546e-01,   2.56389788e-01,   3.86778030e-01,
          5.17166272e-01,   6.47554514e-01,   7.77942756e-01,
          9.08330998e-01,   1.03871924e+00,   1.16910748e+00,
          1.29949572e+00,   1.42988397e+00,   1.56027221e+00,
          1.69066045e+00,   1.82104869e+00,   1.95143693e+00,
          2.08182518e+00,   2.21221342e+00,   2.34260166e+00,
          2.47298990e+00,   2.60337814e+00,   2.73376638e+00,
          2.86415463e+00,   2.99454287e+00,   3.12493111e+00,
          3.25531935e+00,   3.38570759e+00,   3.51609584e+00,
          3.64648408e+00,   3.77687232e+00,   3.90726056e+00,
          4.03764880e+00,   4.16803705e+00,   4.29842529e+00,
          4.42881353e+00,   4.55920177e+00,   4.68959001e+00,
          4.81997826e+00,   4.95036650e+00,   5.08075474e+00,
          5.21114298e+00,   5.34153122e+00,   5.47191947e+00,
          5.60230771e+00,   5.73269595e+00,   5.86308419e+00,
          5.99347243e+00,   6.12386068e+00,   6.25424892e+00,
          6.38463716e+00,   6.51502540e+00,   6.64541364e+00,
          6.77580189e+00,   6.90619013e+00,   7.03657837e+00,
          7.16696661e+00,   7.29735485e+00,   7.42774309e+00,
          7.55813134e+00,   7.68851958e+00,   7.81890782e+00,
          7.94929606e+00,   8.07968430e+00,   8.21007255e+00,
          8.34046079e+00,   8.47084903e+00,   8.60123727e+00,
          8.73162551e+00,   8.86201376e+00,   8.99240200e+00,
          9.12279024e+00,   9.25317848e+00]),
 <a list of 94 Patch objects>)
```




# Pandas

## 1. Read in a CSV () and display all the columns and their respective data types

In [51]:
```
df = pd.read_csv('hw_data.csv', index_col='id')
df.dtypes
```

Out[51]:
```
sex       object
weight     int64
height     int64
dtype: object
```

In [52]:
```
df.head()
```

Out[52]:

| | sex | weight | height |
|---|---|---|---|
| **id** | | | |
| **1** | M | 190 | 77 |
| **2** | F | 120 | 70 |
| **3** | F | 110 | 68 |
| **4** | M | 150 | 72 |
| **5** | O | 120 | 66 |

## 2. Find the average weight

In [53]:
```
df.describe()
```

Out[53]:

| | weight | height |
|---|---|---|
| **count** | 7.000000 | 7.000000 |
| **mean** | 135.714286 | 69.000000 |
| **std** | 27.602622 | 5.259911 |
| **min** | 110.000000 | 60.000000 |
| **25%** | 120.000000 | 67.000000 |
| **50%** | 120.000000 | 70.000000 |
| **75%** | 145.000000 | 71.000000 |
| **max** | 190.000000 | 77.000000 |

## 3. Find the Value Counts on column `sex`

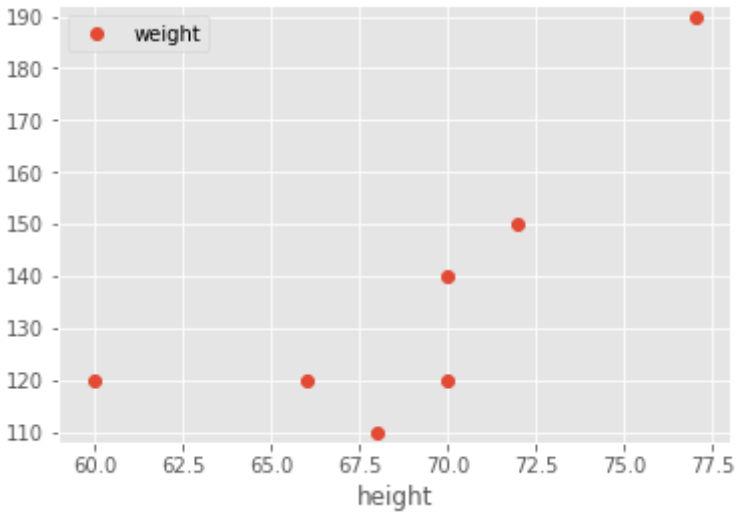
In [56]:
```
df.sex.value_counts()
```

Out[56]:
```
M    3
F    3
O    1
Name: sex, dtype: int64
```

## 4. Plot Height vs. Weight

In [74]:
```
df.plot(x='height'
        ,y='weight'
        ,style= 'o'
        ,xlim=[df.height.min()-1,df.height.max()+1]
        ,ylim=[df.weight.min()-2,df.weight.max()+2])
```

Out[74]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1134f29e8>
```



## 5. Calculate BMI and save as a new column

In [78]:
```
df['BMI'] = df['weight']/df['height']
df
```

Out[78]:

| | sex | weight | height | BMI |
|---|---|---|---|---|
| **id** | | | | |
| **1** | M | 190 | 77 | 2.467532 |
| **2** | F | 120 | 70 | 1.714286 |
| **3** | F | 110 | 68 | 1.617647 |
| **4** | M | 150 | 72 | 2.083333 |
| **5** | O | 120 | 66 | 1.818182 |
| **6** | M | 120 | 60 | 2.000000 |
| **7** | F | 140 | 70 | 2.000000 |

## 6. Save sheet as a new CSV file `hw_dataB.csv`

In [79]:
```
df.to_csv('hw_dataB.csv')
```

## Run the following

In [80]:
```
!cat hw_dataB.csv
```

```
id,sex,weight,height,BMI
1,M,190,77,2.4675324675324677
2,F,120,70,1.7142857142857142
3,F,110,68,1.6176470588235294
4,M,150,72,2.0833333333333335
5,O,120,66,1.8181818181818181
6,M,120,60,2.0
7,F,140,70,2.0
```

In [ ]:
```

```