In [3]:
```
import numpy as np
import pandas as pd
import scipy as sp
```

In [4]:
```
%matplotlib inline
import matplotlib.pyplot as plt
plt.style.use('ggplot')
```

In [5]:
```
%%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

In [ ]:
```
## 1. Finish creating the following function that takes a list and ret

Add each element in the list to `total` and return `total`

### DO NOT use a library function nor `sum()`
```

In [12]:
```
def average(my_list):
    total = 0
    for item in my_list:
        #do something with item!
        total = total + item
        avg = total / len(my_list)

    return avg

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

Out[12]:
```
3.375
```

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

In [15]:
```
def counts(my_list):
    counts = dict()
    for item in my_list:
        #do something with item!
        counts[item] = counts.get(item, 0) + 1

    return counts

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

Out[15]:
```
{1: 2, 2: 2, 4: 1, 3: 1, 5: 1, 9: 1}
```

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

In [48]:
```
paragraph_text = '''
For a minute or two she stood looking at the house, and wondering what
The Fish-Footman began by producing from under his arm a great letter,
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
Alice went timidly up to the door, and knocked.
‘There’s no sort of use in knocking,’ said the Footman, ‘and that for
‘Please, then,’ said Alice, ‘how am I to get in?’
‘There might be some sense in your knocking,’ the Footman went on with
‘I shall sit here,’ the Footman remarked, ‘till tomorrow—’
At this moment the door of the house opened, and a large plate came sk

counts(paragraph_text.split())
```

Out[48]:
```
{'For': 2,
 'a': 15,
 'minute': 1,
 'or': 2,
 'two': 2,
 'she': 6,
 'stood': 1,
 'looking': 2,
 'at': 6,
 'the': 32,
 'house,': 1,
 'and': 16,
 'wondering': 1,
 'what': 2,
 'to': 15,
 'do': 1,
 'next,': 1,
 'when': 2,
 'suddenly': 1,
```

4. Read in a file using `open()` and iterated through the file line-by-line write each line from the file to a new file in a `title()`-ized. Create your own file for input

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

Hint: There's a function to do this

In [105]:
```
text = '''
Four score and seven years ago our fathers brought forth on this conti
Now we are engaged in a great civil war, testing whether that nation,
We have come to dedicate a portion of that field, as a final resting p
It is altogether fitting and proper that we should do this.
But, in a larger sense, we can not dedicate—we can not consecrate—we c
The brave men, living and dead, who struggled here, have consecrated i
The world will little note, nor long remember what we say here, but it
It is for us the living, rather, to be dedicated here to the unfinishe
It is rather for us to be here dedicated to the great task remaining b

with open('text.txt', 'r') as f1, open('new_text.txt', 'a') as f2:
    lines = f1.readlines()
    new_lines = list(map(str.title, lines))
    print(new_lines)
```

```
['\n', 'Four Score And Seven Years Ago Our Fathers Brought Forth On T
his Continent, A New Nation, Conceived In Liberty, And Dedicated To T
he Proposition That All Men Are Created Equal.\n', 'Now We Are Engage
d In A Great Civil War, Testing Whether That Nation, Or Any Nation So
Conceived And So Dedicated, Can Long Endure. We Are Met On A Great Ba
ttle-Field Of That War.\n', 'We Have Come To Dedicate A Portion Of Th
at Field, As A Final Resting Place For Those Who Here Gave Their Live
s That That Nation Might Live.\n', 'It Is Altogether Fitting And Prop
er That We Should Do This.\n', 'But, In A Larger Sense, We Can Not De
dicate—We Can Not Consecrate—We Can Not Hallow—This Ground. \n', 'The
Brave Men, Living And Dead, Who Struggled Here, Have Consecrated It,
Far Above Our Poor Power To Add Or Detract.\n', 'The World Will Littl
e Note, Nor Long Remember What We Say Here, But It Can Never Forget W
hat They Did Here.\n', 'It Is For Us The Living, Rather, To Be Dedica
ted Here To The Unfinished Work Which They Who Fought Here Have Thus
Far So Nobly Advanced.\n', 'It Is Rather For Us To Be Here Dedicated
To The Great Task Remaining Before Us—That From These Honored Dead We
Take Increased Devotion To That Cause For Which They Gave The Last Fu
ll Measure Of Devotion—That We Here Highly Resolve That These Dead Sh
all Not Have Died In Vain—That This Nation, Under God, Shall Have A N
ew Birth Of Freedom—And That Government Of The People, By The People,
For The People, Shall Not Perish From The Earth.']
```

Numpy

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

In [92]:
```
simple_list = [1,2,1,4,3,2,5,9]
np.mean(simple_list)
```

Out[92]:
```
3.375
```

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

In [116]:
```
heights = [174, 173, 173, 175, 171]
weights = [88, 83, 92, 74, 77]

height = np.array(heights)
weight = np.array(weights)

bmi =  weight / (height/100) **2

print(bmi)
```

```
[29.06592681 27.73229978 30.73941662 24.16326531 26.33288875]
```

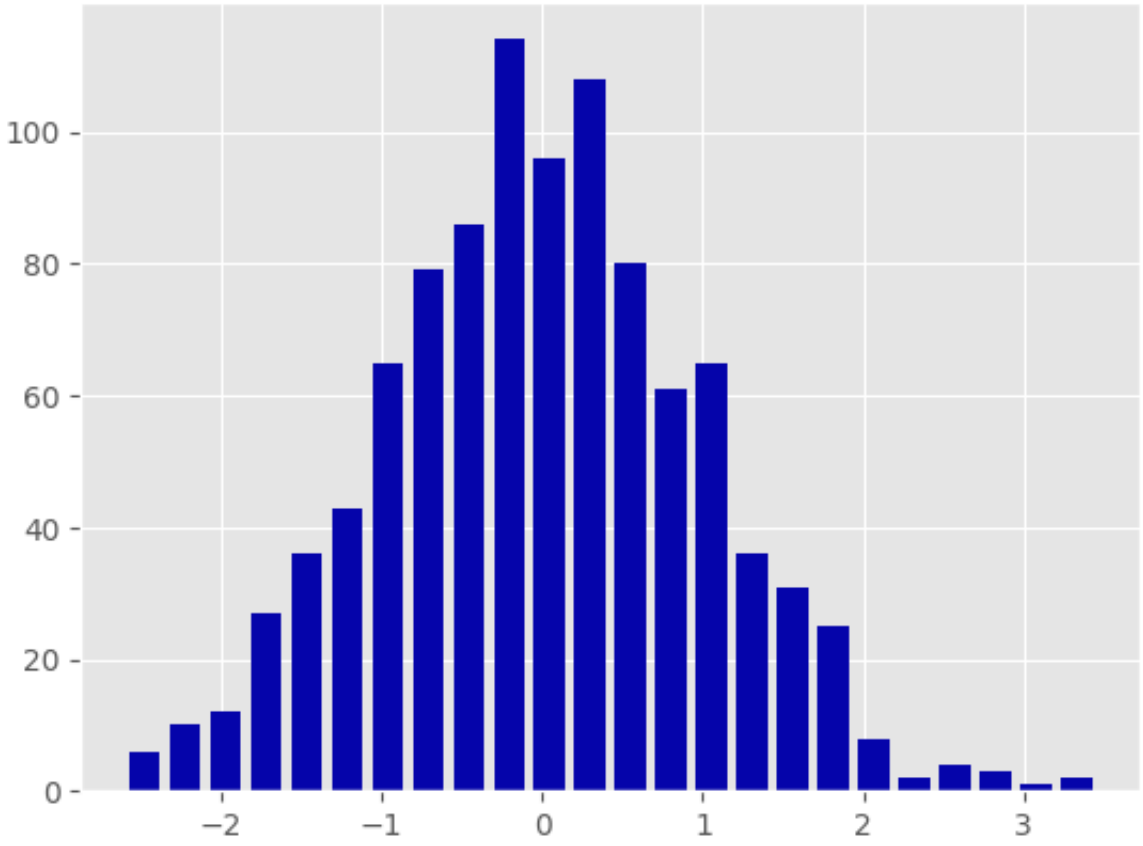
3. Create an array of length 20 filled with random values (between 0 to 1)

In [118]:
```
rarray = np.random.rand(20)
print(rarray)
```

```
[0.08881928 0.97649917 0.67822235 0.78498403 0.46045386 0.62413584
 0.57802122 0.81273883 0.10569177 0.93905747 0.16230417 0.82259147
 0.33721533 0.13502602 0.10550672 0.25041207 0.2984878  0.01536585
 0.60226948 0.45442775]
```

4. Create an array with at least 1000 random numbers from normal distributions (normal). Then, plot a histogram of these values (`plt.hist`).

In [124]:
```
a = np.random.randn(1000)
random_hist = plt.hist(x=a, bins='auto', color='#0504aa', rwidth=0.75)
```



Pandas

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

In [134]:
```
df = pd.read_csv("hw_data.csv", index_col = 'id')
print(df)
df.dtypes
```

```
   sex  weight  height
id
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
```

Out[134]:
```
sex       object
weight     int64
height     int64
dtype: object
```

2. Find the average weight

In [143]:
```
df.iloc[:,1:2].mean()
```

Out[143]:
```
weight    135.714286
dtype: float64
```

3. Find the Value Counts on column `sex`

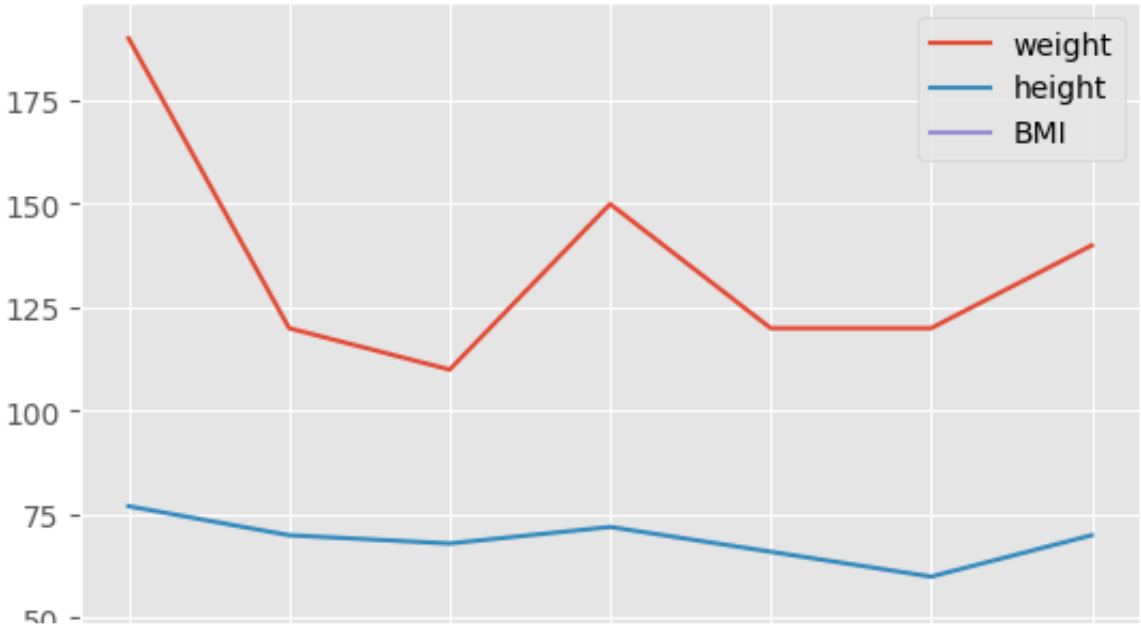
In [149]:
```
df['sex'].value_counts()
```

Out[149]:
```
M    3
F    3
O    1
Name: sex, dtype: int64
```

4. Plot Height vs. Weight

In [159]:
```
df.plot()
```

Out[159]: <AxesSubplot: xlabel='id'>



5. Calculate BMI and save as a new column

In [156]:
```
df['BMI'] = df.weight * 703 / df.height **2
print(df)
```

```
   sex  weight  height        BMI
id
1    M     190      77  22.528251
2    F     120      70  17.216327
3    F     110      68  16.723616
4    M     150      72  20.341435
5    O     120      66  19.366391
6    M     120      60  23.433333
7    F     140      70  20.085714
```

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

In [157]:
```
df.to_csv('hw_dataB.csv')
df2 = pd.read_csv("hw_dataB.csv", index_col = 'id')
print(df2)
```

```
   sex  weight  height        BMI
id
1    M     190      77  22.528251
2    F     120      70  17.216327
3    F     110      68  16.723616
4    M     150      72  20.341435
5    O     120      66  19.366391
6    M     120      60  23.433333
7    F     140      70  20.085714
```

Run the following (Mac)

In [158]:
```
!cat hw_dataB.csv
```

```
id,sex,weight,height,BMI
1,M,190,77,22.528250969809413
2,F,120,70,17.216326530612246
3,F,110,68,16.723615916955016
4,M,150,72,20.341435185185187
5,O,120,66,19.366391184573004
6,M,120,60,23.433333333333334
7,F,140,70,20.085714285714285
```

Run the following (Windows)

In [ ]:
```
!type hw_dataB.csv
```