Section 08

Plotting

- **Line(s)**

```
import matplotlib.pyplot as plt
xs = range(100)
ys = [x**2 for x in xs]
plt.title('Lines')
plt.xlabel('X')
plt.ylabel('Y')
plt.plot(xs, ys)
plt.show()
```

If you want to plot multiple lines call plt.plot() again with different X,Y values before the call to plt.show()

```
ys2 = [2*(x**2) for x in xs]
ys3 = [4*(x**2) for x in xs]
plt.plot(xs, ys2)
plt.plot(xs, ys3)
plt.show()
```

You can specify line width, line color, even the line style,

```
plt.plot(xs, ys, linewidth=2, color='green', linestyle='-', marker='s', label=
"y=x^2")
```

To see a list of acceptable line styles and colors visit, http://matplotlib.org/users/pyplot_tutorial.html

- **Scatterplot**

```
import matplotlib.pyplot as plt
plt.title('Scatterplot')
plt.xlabel('X')
plt.ylabel('Y')
x = [0,1,2,3,4,5,6,7,8,9,10]
y = [2,3,4,5,6,7,8,9,11,12,13]
for index in range(len(x)):
plt.scatter(x[index], y[index], c = 'blue')
plt.show()
```

- **Pie Chart**

```
from pylab import *
import matplotlib.pyplot as plt
#creates the figure and sets its size
figure(1, figsize=(7,7))
#centers the figure
ax = axes([.2, .2, .6, .6])
colors = ['red', 'green', 'white', 'yellow']
labels = ['cat', 'dog', 'fish', 'bird']
fracs = [11,24,37,8]
#autopct places the percentages inside their corresponding section
plt.pie(fracs, colors = colors, labels=labels, autopct='%1.1f%%')
plt.title('Pets Owned')
plt.show()
```

For more information visit the following links,
http://msenux.redwoods.edu/math/python/pie.php
http://matplotlib.org/1.2.1/examples/pylab_examples/pie_demo.html

- **Histogram**

```
import matplotlib.pyplot as plt
x = [1,2,3,4]
freqs = [10, 11, 15, 5]
#width of bins
width = .5
plt.xlim([1,5])
plt.title("Histogram")
plt.xlabel('Quarter')
plt.ylabel('Frequency of Robberies')
plt.bar(x, freqs, width, color='m')
plt.show()
```

For more information visit the following links,
http://stackoverflow.com/questions/5926061/plot-histogram-in-python
http://bespokeblog.wordpress.com/2011/07/11/basic-data-plotting-with-matplotlib-part-3-histograms/

Section 08

Pearson r Correlation Coefficient

```
from scipy.stats import pearsonr
pearsonr(x, y)
```

Documentation:

Calculates a Pearson correlation coefficient and the p-value for testing non-correlation.

The Pearson correlation coefficient measures the linear relationship between two datasets. Strictly speaking, Pearson's correlation requires that each dataset be normally distributed. Like other correlation coefficients, this one varies between -1 and +1 with 0 implying no correlation. Correlations of -1 or +1 imply an exact linear relationship. Positive correlations imply that as x increases, so does y. Negative correlations imply that as x increases, y decreases.

The p-value roughly indicates the probability of an uncorrelated system producing datasets that have a Pearson correlation at least as extreme as the one computed from these datasets. The p-values are not entirely reliable but are probably reasonable for datasets larger than 500 or so.

Parameters:

x : 1D array
y : 1D array the same length as x

Returns:

The following tuple,
(Pearson's correlation coefficient, 2-tailed p-value)

Example:

```
pearsonr([1,2,3,4,5], [2,4,6,8,10])
returns → (1.0, 0.0)
#this says there is a perfect linear relationship between x & y and the probability of
#observing such a relationship in a sample of size 5 from a dataset that is actually
#uncorrelated is 0.0

pearsonr([0,7,11,1,-5],[-2,2000,-1000,-11,0])
returns → (0.0082114722023958146, 0.98954494636829993)
#there is no linear relationship whatsoever between x & y and the probability of
#observing such a relationship in a sample of size 5 from a dataset that is actually
#uncorrelated is .9895
```

Note:

Because the p-value is not as reliable it is not necessary to include it in your analysis if you choose to do analysis on correlation.