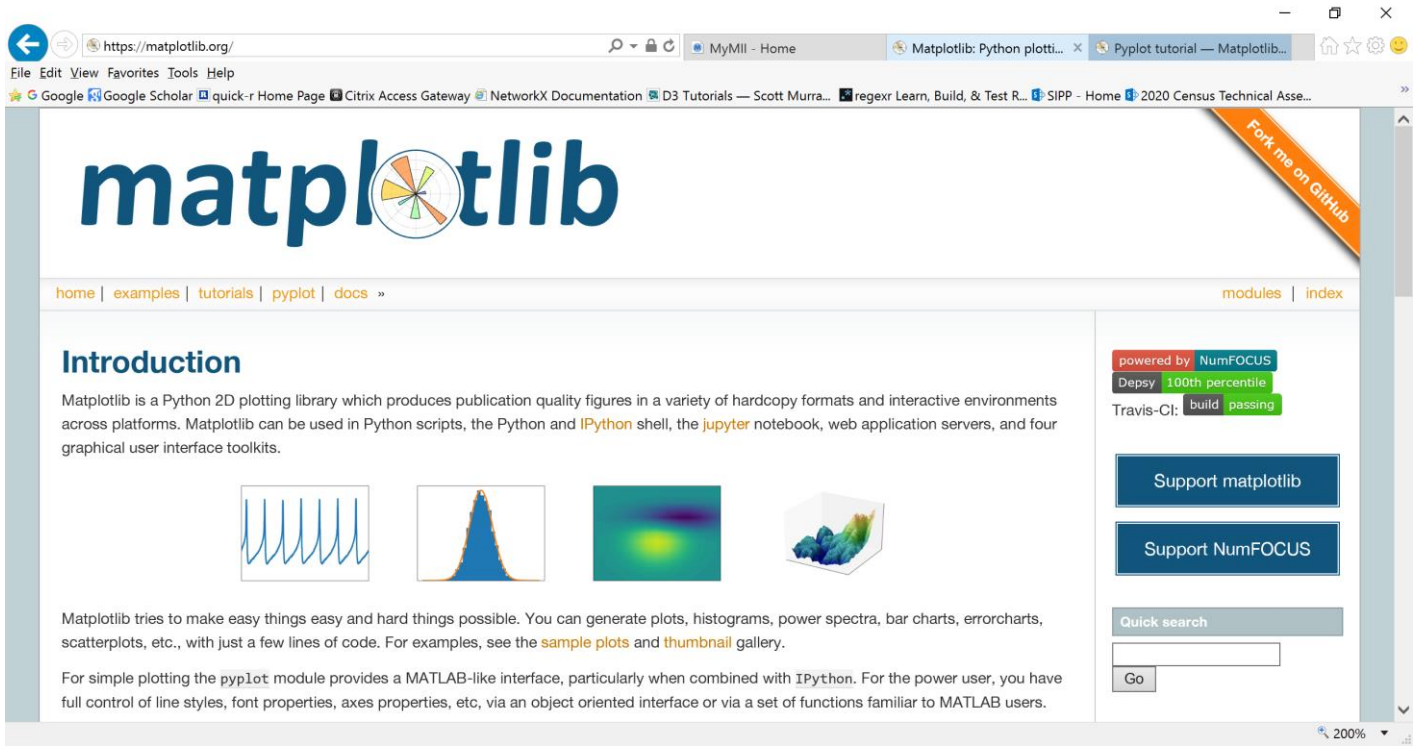
04-1 Graphics and Plots Introduction

CSI 500

Matplotlib and Pyplot

- Matplotlib is a Python-based open source data visualization project
  - Graphics gallery of examples with full source code available
  - Outstanding online documentation
- Pyplot is a Python-specific set of APIs for plots and graphs
  - We'll be working thru the tutorial examples

https://matplotlib.org/

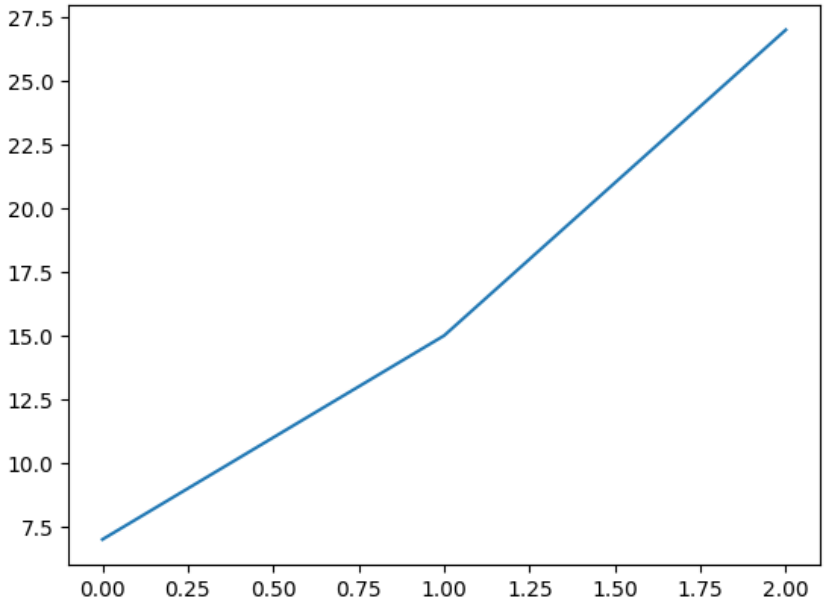


https://matplotlib.org/tutorials/introductory/pyplot.html#sphx-glr-tutorials-introductory-pyplot-py

Our first example

- make a 2D plot of some data
  - first argument is assumed to be a List containing Y values
- Result is a basic 2D graph
  - note: X axis starts at 0

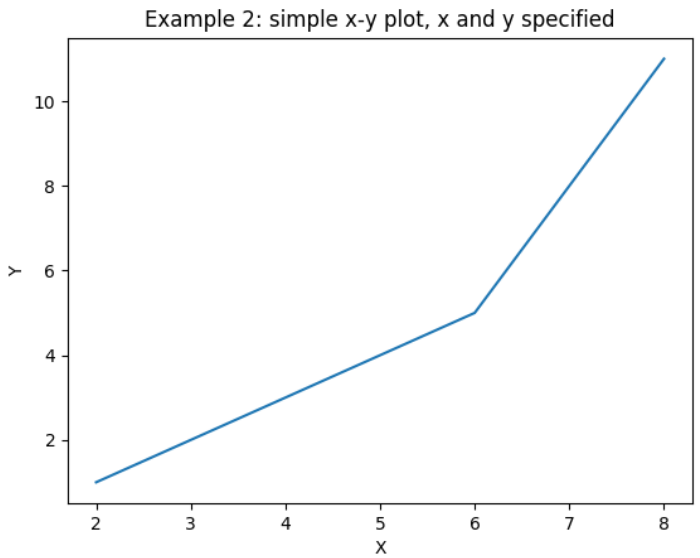
```
[nn]  import matplotlib.pyplot as plt

[nn]  plt.plot( [7, 15, 27] )
[<matplotlib.lines.Line2D object at 0x025B05DF2E80>]
[nn]  plt.show()
```



Another 2D example

- Here we specify the X and Y values
- Add some embellishments
  - Y label
  - X label
  - Title
- Things to notice
  - X axis begins at 2, as we requested
  - plot assumes coordinate pair is x[i], y[i]

```
import matplotlib.pyplot as plt

yvals = [ 1, 3, 5, 11 ]
xvals = [ 2, 4, 6, 8 ]
plt.plot( xvals, yvals )

plt.ylabel('Y')
plt.xlabel('X')
plt.title('Example 2: simple x-y plot, x and y specified')

plt.show()
```



Plots display features

- Iconography
  - Matplotlib supports a wide variety of simple icons in addition to the default smooth line
- Colors
  - Matplotlib supports a small set of popular colors with single character abbreviations
  - Much larger color set is supported via RGB, HSB, and hex code color specification

Icons	
'-' solid line style	'3' tri_left marker
'--' dashed line style	'4' tri_right marker
'-.' dash-dot line style	's' square marker
':' dotted line style	'p' pentagon marker
'.' point marker	'*' star marker
',' pixel marker	'h' hexagon1 marker
'o' circle marker	'H' hexagon2 marker
'v' triangle_down marker	'+' plus marker
'^' triangle_up marker	'x' x marker
'<' triangle_left marker	'D' diamond marker
'>' triangle_right marker	'd' thin_diamond marker
'1' tri_down marker	'|' vline marker
'2' tri_up marker	'_' hline marker

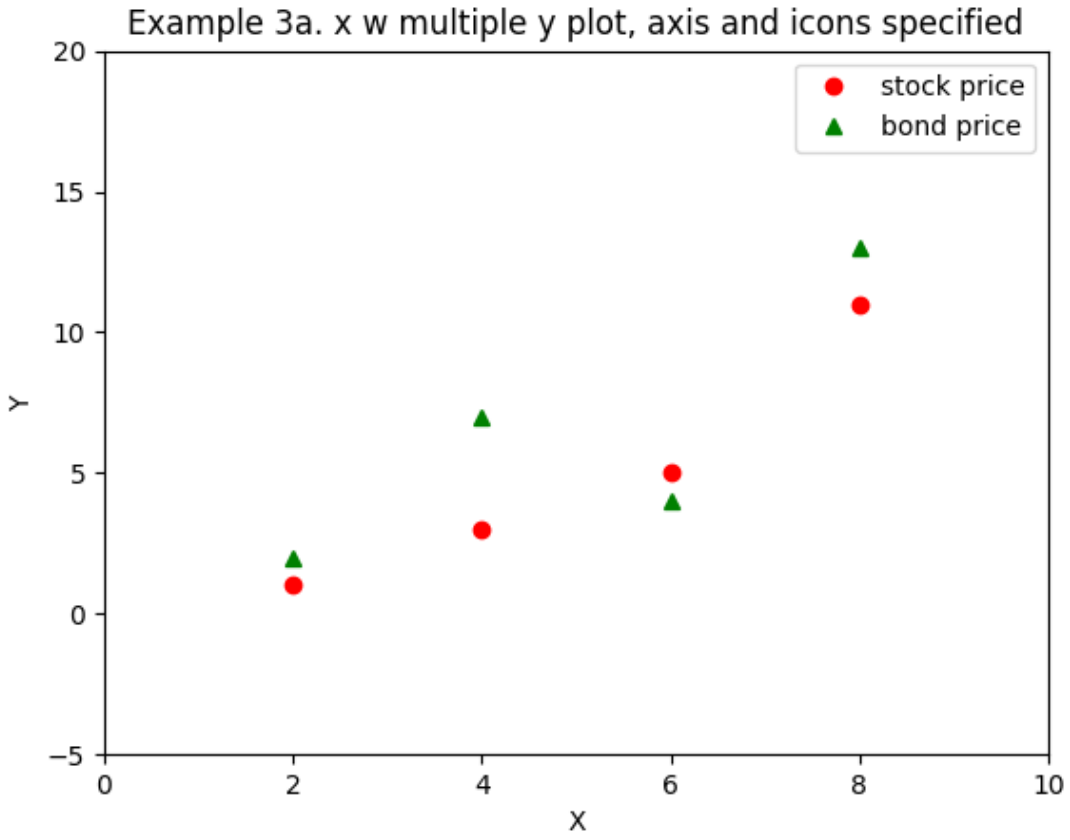
Colors	
‘b’ blue	‘m’ magenta
‘g’ green	‘y’ yellow
‘r’ red	‘k’ black
‘c’ cyan	‘w’ white

Another 2D example

- Here we specify the X and two sets of Y values
- One set of Y values uses red circles
  - indicated by 'ro'
- One set of Y values uses green triangles
  - indicated by 'g^'
- Axis limits for x and y are specified
  - note use of 4 element List
- We've added a legend as well

```
import matplotlib.pyplot as plt

y1 = [ 1, 3, 5, 11 ]
y2 = [ 2, 7, 4, 13 ]
xvals = [ 2, 4, 6, 8 ]

plt.plot( xvals, y1, 'ro', label = 'stock price' )
plt.plot( xvals, y2, 'g^', label = 'bond price' )

xmin = 0
xmax = 10
ymin = -5
ymax = 20
plt.axis( [ xmin, xmax, ymin, ymax ] )

plt.ylabel('Y')
plt.xlabel('X')
plt.title('Ex 3a. x w multiple y plots, axis & icons')
plt.legend()
plt.show()
```

What it looks like



Quick aside: List Comprehensions

- The "list comprehension" is a shortcut used in Python to generate a list
- Very similar to what we've used all along with for-loops

Example with a For-Loop

```
for k in range(0,5):
    print(k)

0
1
2
3
4
```

Example with a List Comprehension

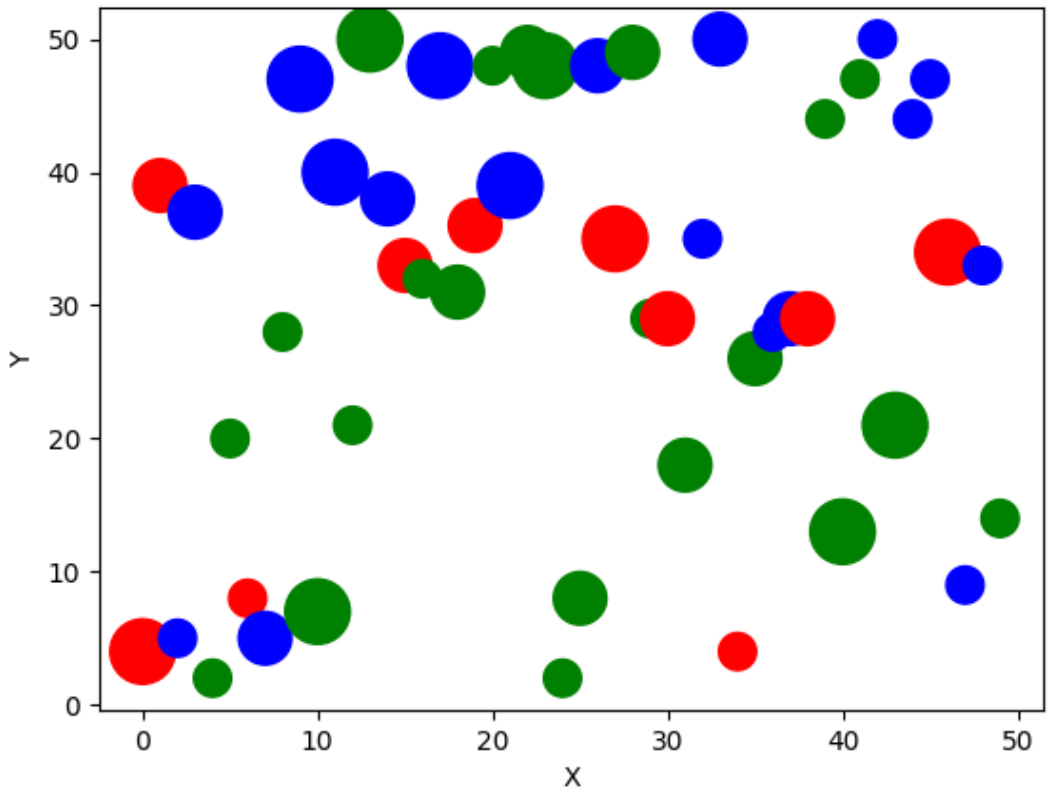
```
foo = [ k for k in range(0,5) ]

print(foo)
[ 0, 1, 2, 3, 4 ]
```

Scatterplot example

- specify some size constants
- define size_list for icon sizes
- define color_list for icon colors
- define a mydata dictionary
  - x holds List of x values
  - y holds List of y values
  - sizes holds List of icon sizes
  - colors holds List of icon colors
- plt.scatter builds the scatter plot
  - parameters are the dictionary key values

```
import numpy as np
import matplotlib.pyplot as plt

numvals = 50
maxval = 50
size_list = [ 200, 400, 600 ]
color_list = [ 'red', 'green', 'blue' ]

mydata = { 'x' : [ n for n in range(0, numvals) ],
        'y' : np.random.random_integers(0, maxval, numvals ),
        'sizes' : [ np.random.choice( size_list )
                for n in range(0, numvals) ],
        'colors' : [ np.random.choice( color_list )
                 for n in range(0, numvals) ]
        }

plt.scatter( 'x', 'y', s='sizes', c='colors', data=mydata)

plt.xlabel('X')
plt.ylabel('Y')

plt.show()
```

What it looks like



Barchart example

- useful for basic statistics
- set up the labels and data
- use a "subplots" to get figure and axes

- use the plt.bar chart method
- assign the 3 returned "bar" objects with specified colors

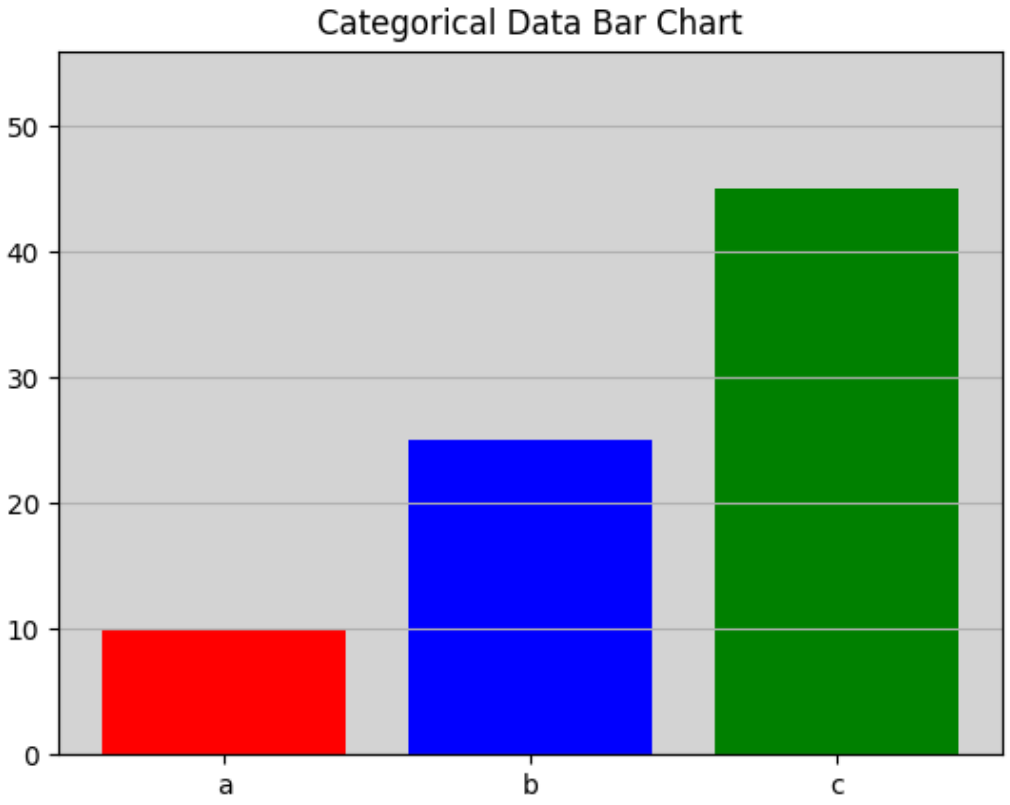
```
import numpy as np
import matplotlib.pyplot as plt

group_labels = ['a', 'b', 'c']
group_ids = [1, 2, 3]
data_values = [10, 25, 45]

fig, ax = plt.subplots()
indexes = np.arange(1,3)

# show figure, don't block
plt.show( block=False)

ba, bb, bc = plt.bar( group_ids, data_values )
ba.set_facecolor('r')
bb.set_facecolor('b')
bc.set_facecolor('g')
```

Barchart example (cont)

- configure the X axis ticks and labels
- add y limits
- add labels and embellishments
- set background color to light gray
- add a y-axis grid
- display

```
ax.set_xticks( group_ids )
ax.set_xticklabels( group_labels )

ax.set_ylim( [0, round( max(data_values) * 1.25) ] )

ax.set_label('Treatment Types')
ax.set_title('Categorical Data Bar Chart')

ax.set_facecolor('lightgray')

ax.grid( axis='y')

plt.show()
```

What it looks like

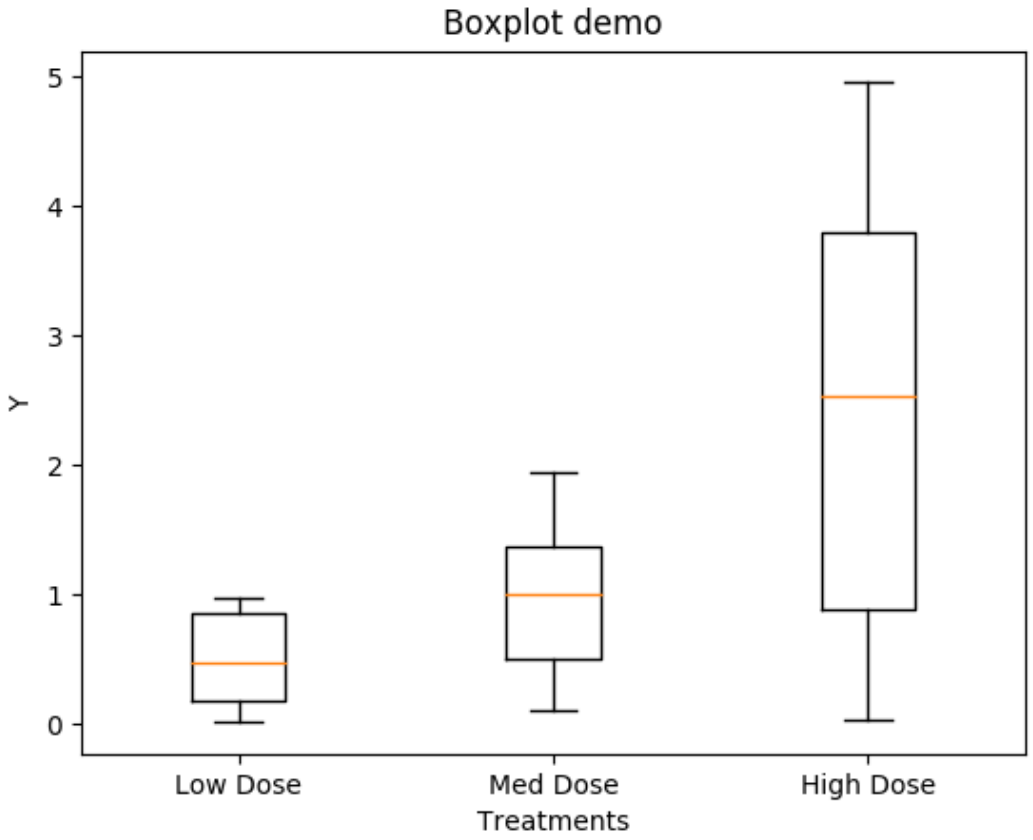


Boxplot example

- useful for basic statistics
- data stored in List object
- labels stored as Strings in List object
- specify labels when boxplot invoked
- Many other options available - this is a basic boxplot

```
import numpy as np
import matplotlib.pyplot as plt

# make some data
d1 = 1.0 * np.random.random(50)
d2 = 2.0 * np.random.random(50)
d3 = 5.0 * np.random.random(50)
label_list = [ 'Low Dose', 'Med Dose', 'High Dose']

# set data as list of lists
data = [d1, d2, d3]
plt.boxplot( data, labels = label_list )

# set up labels and such
plt.title('Boxplot demo')
plt.ylabel('Y')
plt.xlabel( 'Treatments' )

# display
plt.show()
```

What it looks like



Summary

- Matplotlib Pyplot provides extensive plotting and graphics support
  - lots of options available for colors, fonts, annotations, etc
  - Consult online and printed documentation
  - Experiment with small data sets - have fun!
- Use plot() for basic 2-D X-Y plots
- Use scatter() for 2-D scatter plots
- Use bar() for bar charts
- Use boxplot() for box plots