Discrete Structures!

CMPSC 102

Plots

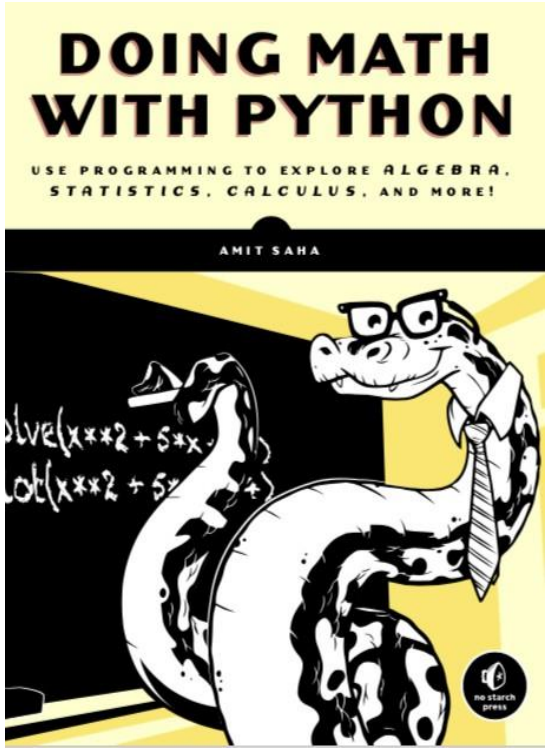
ALLEGHENY COLLEGE

Key Questions and Learning Objectives

- How do I implement data structures to create plots? How do I install such masterful software to do this?!

- To remember and understand some concepts about plots, and the code used to make them from matplotlib.

Where Are We Now?

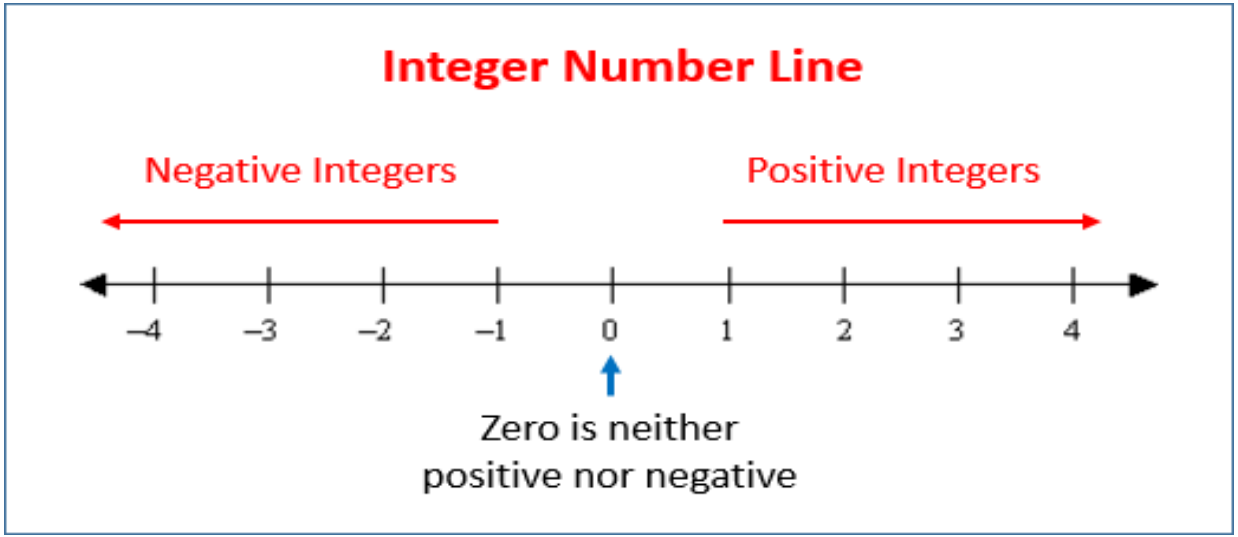


Saha, Chapter 2: Visualizing Data with graphs

- How to present data with graphics
- Plotting basic numbers
- Plotting results from equations
- Plotting all kinds of things!

A Number Line: x

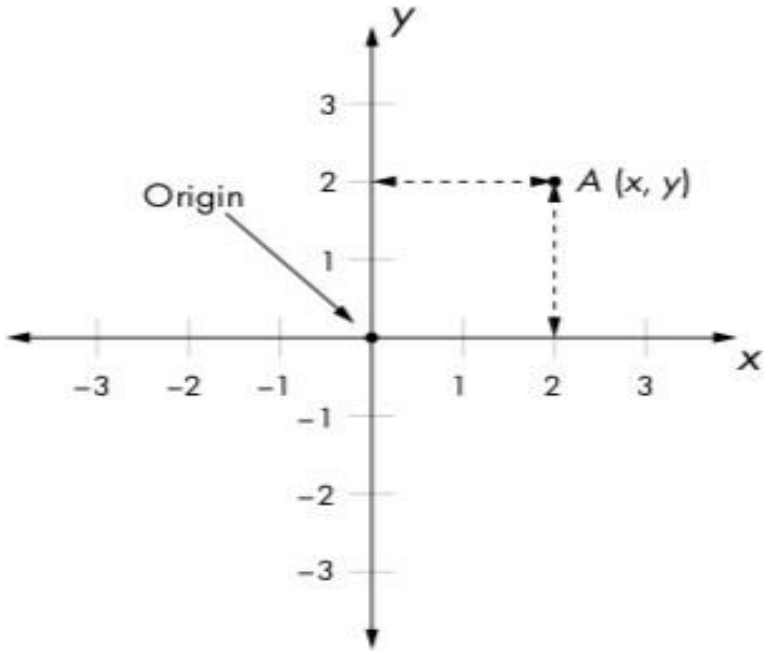
Denoted R



- The x-axis runs horizontally left to right
- The middle of the number line is where $x = 0$
- Left of 0: negative numbers (all kinds of numbers!)
- Right of 0: positive numbers (all kinds of numbers, too!)

Cartesian System, 2-D Coordinates: x and y

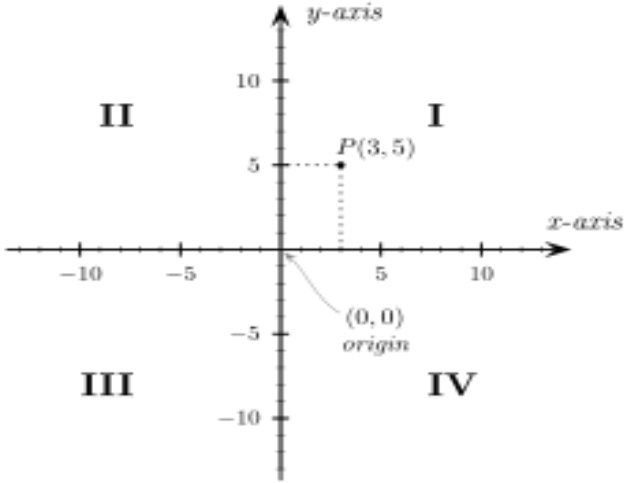
Denoted R^2



- The *x*-axis runs along the bottom (horizontally left to right)
- The *y*-axis runs along the side (vertically bottom to top)
- Typically, the (0, 0) point (the origin) is shown where $x = 0$ and $y = 0$

2-D Coordinates: x and y

Denoted R^2

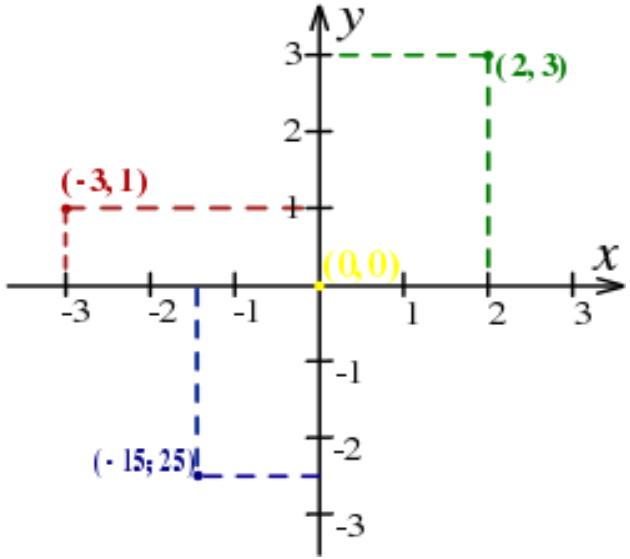


- The intersection of the values of x and y creates the 2-D point (called the ordered pair) on the canvas.
- There are four quadrants defined by:

1. Quadrant I: (x, y)
2. Quadrant II: $(-x, y)$
3. Quadrant III: $(-x, -y)$
4. Quadrant IV: $(x, -y)$

Example Coordinates: x and y

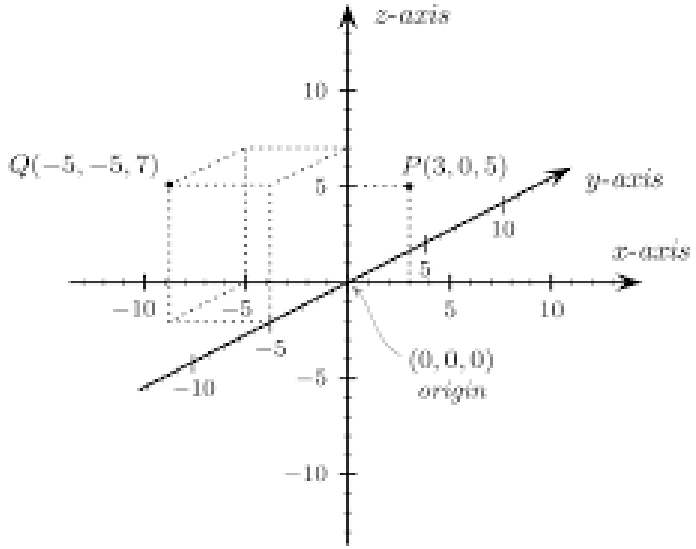
Example plot



- Origin: (0, 0)
- Green: (2, 3)
- Red: (−3, 1)
- Blue: (−1.5, −2.5)

3-D Coordinates: x , y , and z

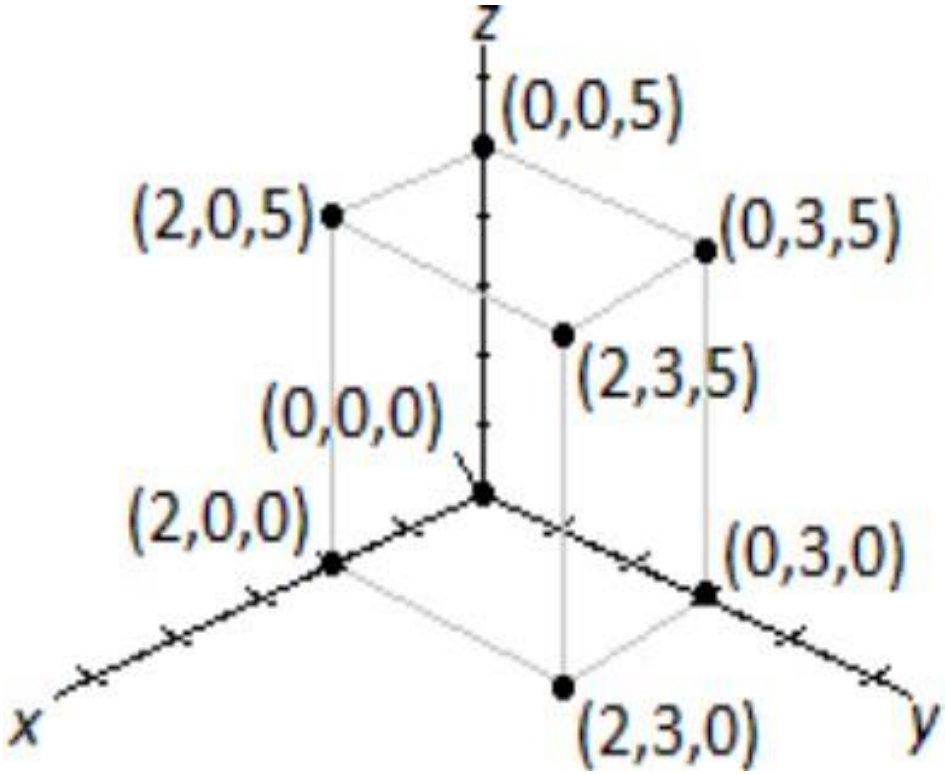
Denoted R^3



- The three number lines are called the *x*-axis, the *y*-axis, and the *z*-axis and are called the *coordinate axes*
- The intersection of the values of *x*, *y* and *z* creates the point defined by the ordered triple on the canvas.

3-D Coordinates: x , y , and z

Example plot



Matplotlib

- Matplotlib is a Python plotting library
- Produces publication-quality figures in Python in a variety of hardcopy formats and interactive environments across platforms.
- Allows you to plot your data without much extra coding

Setting Up Virtual Environment

- Create a project directory

```
mkdir projects
cd projects
```

- Create virtual environment using Python

```
python3 -m venv myenv
# see the file tree
find . -not -path '*/\.*'
```

- Activate myenv the virtual environment

```
source myenv/bin/activate  # macOS/Linux
myenv\Scripts\activate     # Windows
```

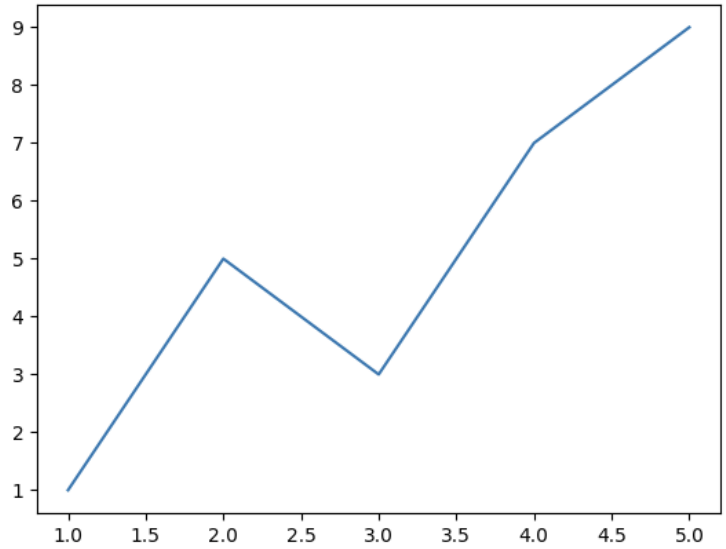
- Install Dependencies

```
pip install matplotlib
pip install numpy
```

Your First Plot

Plot some simple points

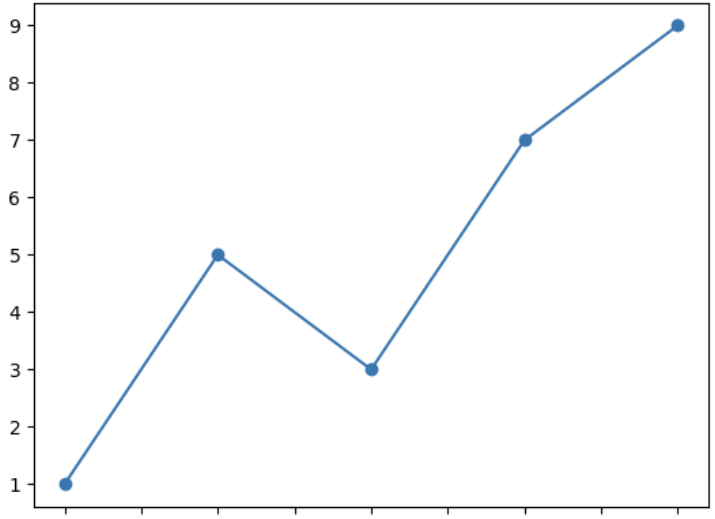
```
import matplotlib.pyplot as plt #get the library
x_num = [1,2,3,4,5] #def of x
y_num = [1,5,3,7,9] # def of y
plt.plot(x_num, y_num) # gives mem addr of obj
plt.show() # draw the plot on canvas
```



Gimme Points, Not Lines

Plot some basic numbers using points

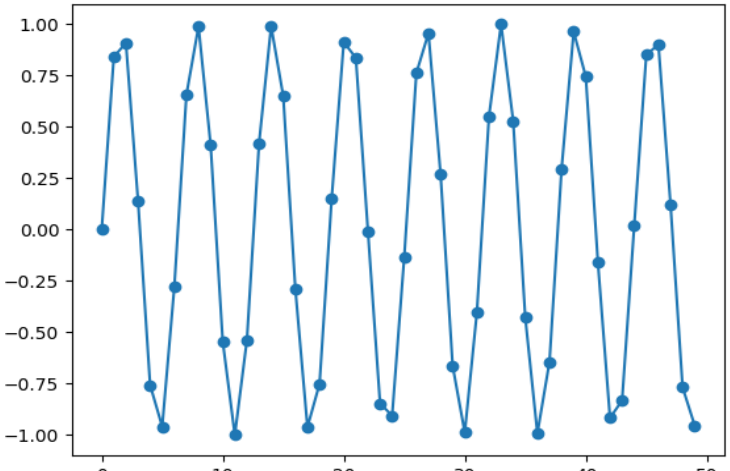
```
import matplotlib.pyplot as plt #get the library
x_num = [1,2,3,4,5] #def of x
y_num = [1,5,3,7,9] # def of y
plt.plot(x_num, y_num, marker='o')
# also including 'o', '*', 'x', and '+' as points
plt.show() # draw the plot on canvas
```



Another Amazing Example!

Plot the sin wave

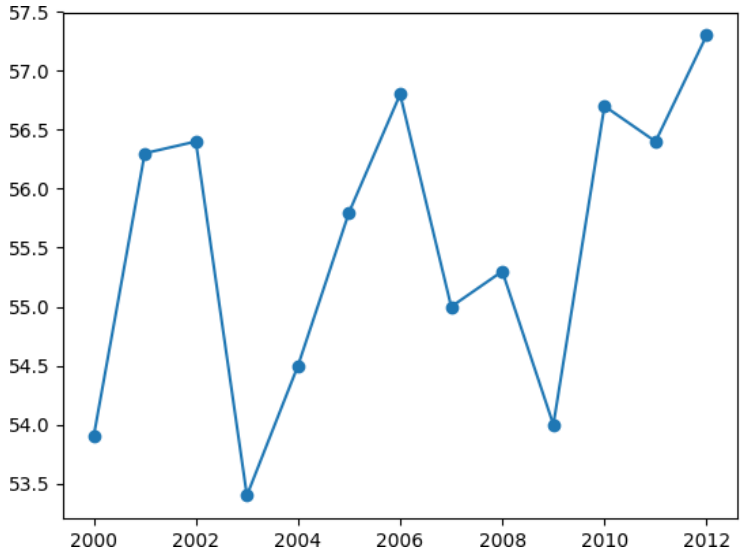
```
import matplotlib.pyplot as plt #get the library
import math
x_num = [i for i in range(50)]
y_num = [math.sin(i) for i in x_num]
plt.plot(x_num, y_num, marker='o')
# also including 'o', '*', 'x', and '+' as points
plt.show() # draw the plot on canvas
```



Yet, Another Amazing Example!

Plot the temperature in NYC and save the file too!

```
import matplotlib.pyplot as plt
nyc_temp = [53.9, 56.3, 56.4, 53.4, 54.5, 55.8, 56.8, 55.0, 55.3, 54.0, 56.7, 56.4, 57.3]
years = range(2000, 2013)
plt.plot(years, nyc_temp, marker='o')
# also including 'o', '*', 'x', and '+' as points
plt.savefig('mygraph.png') #save in root directory
plt.show() # draw the plot on canvas
```



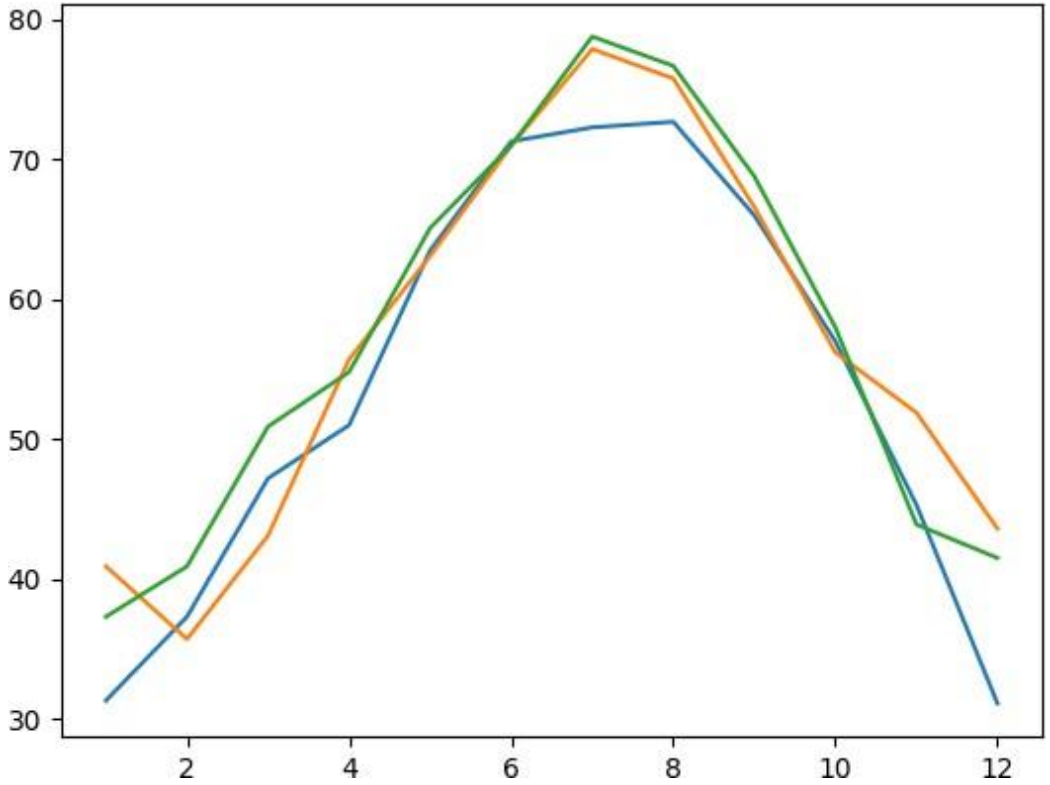
Three Plots Together! Amazing!

Plot the temperature in NYC aggregated by time

```
import matplotlib.pyplot as plt
months = range(1, 13)

nyc_temp_2000 = [31.3, 37.3, 47.2, 51.0, 63.5, 71.3,
72.3, 72.7, 66.0, 57.0, 45.3, 31.1]

nyc_temp_2006 = [40.9, 35.7, 43.1, 55.7, 63.1, 71.0,
77.9, 75.8, 66.6, 56.2, 51.9, 43.6]

nyc_temp_2012 = [37.3, 40.9, 50.9, 54.8, 65.1, 71.0,
78.8, 76.7, 68.8, 58.0, 43.9, 41.5]

plt.plot(months, nyc_temp_2000, months, nyc_temp_2006, months, nyc_temp_2012)
plt.savefig('mygraph.png') #save in root directory
plt.show() # draw the plot on canvas
```

Three Plots Together! Amazing!

Plot the temperature in NYC aggregated by time



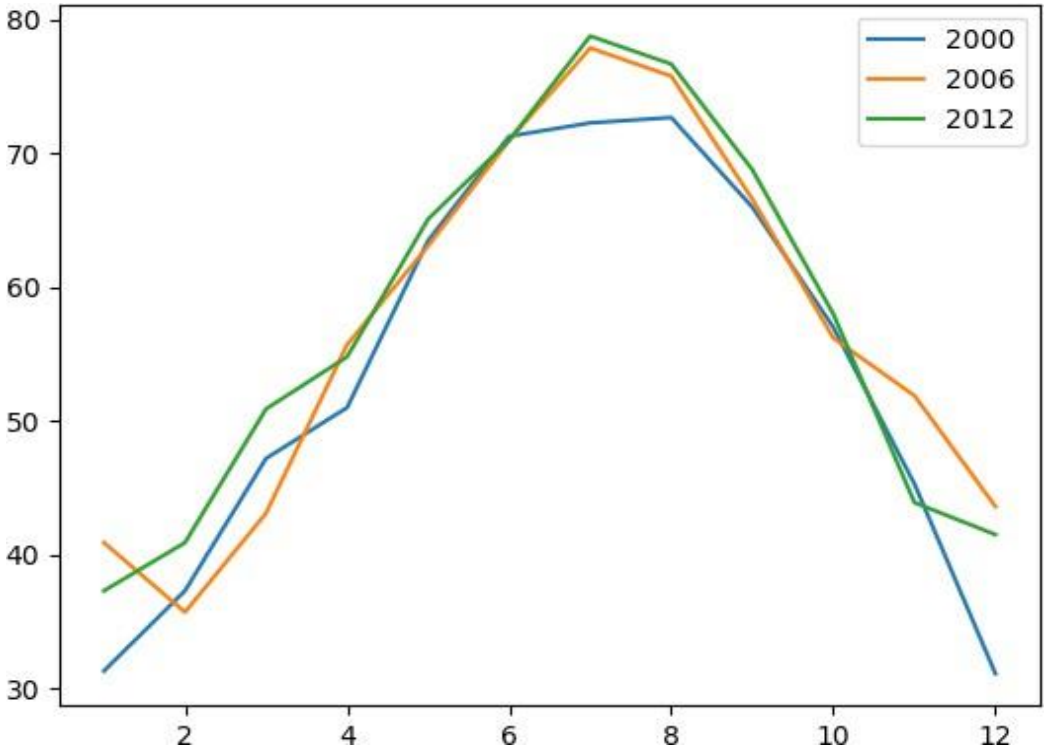
Three Plots Together! And a LEGEND Too!

Plot the temperature in NYC aggregated by time

```
import matplotlib.pyplot as plt
months = range(1, 13)

nyc_temp_2000 = [31.3, 37.3, 47.2, 51.0, 63.5, 71.3,
72.3, 72.7, 66.0, 57.0, 45.3, 31.1]

nyc_temp_2006 = [40.9, 35.7, 43.1, 55.7, 63.1, 71.0,
77.9, 75.8, 66.6, 56.2, 51.9, 43.6]

nyc_temp_2012 = [37.3, 40.9, 50.9, 54.8, 65.1, 71.0,
78.8, 76.7, 68.8, 58.0, 43.9, 41.5]

plt.plot(months, nyc_temp_2000, months, nyc_temp_2006, months, nyc_temp_2012)
plt.legend([2000, 2006, 2012]) # make the legend
plt.savefig('mygraph.png') #save in root directory
plt.show() # draw the plot on canvas
```

Three Plots Together! And a LEGEND Too!

Plot the temperature in NYC aggregated by time



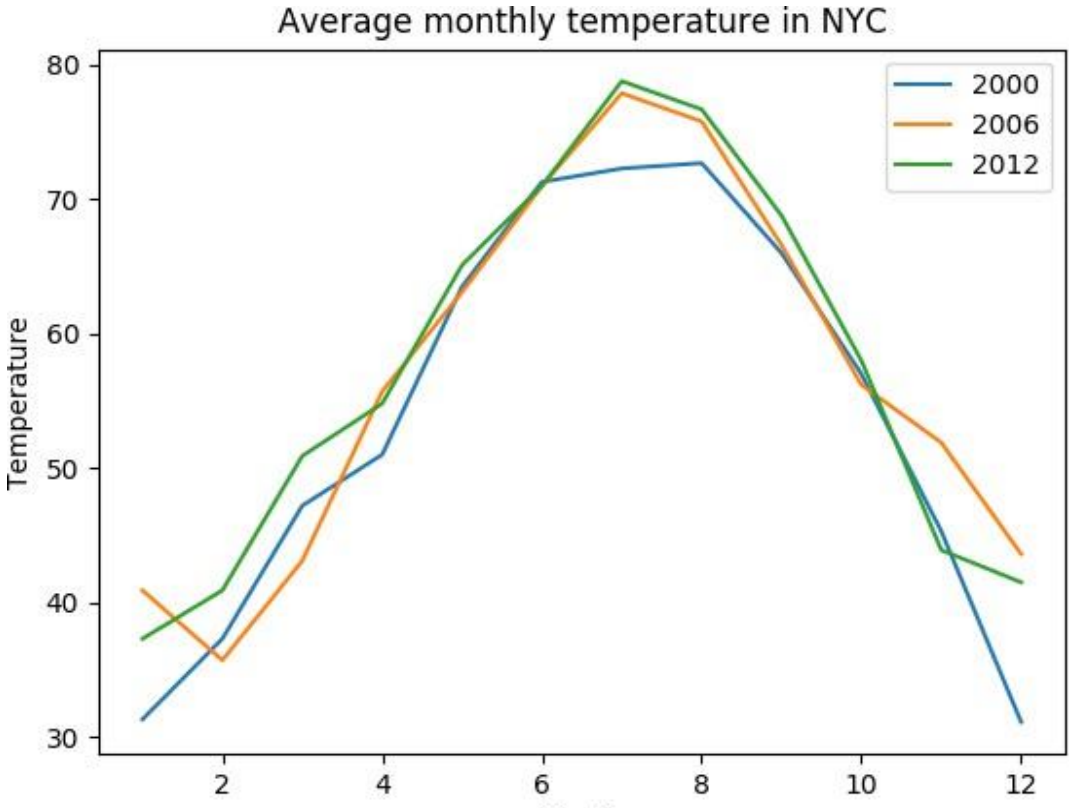
Add Title and Axes Descriptions!

Plot the temperature in NYC aggregated by time

```
import matplotlib.pyplot as plt
months = range(1, 13)

nyc_temp_2000 = [31.3, 37.3, 47.2, 51.0, 63.5, 71.3,
72.3, 72.7, 66.0, 57.0, 45.3, 31.1]

nyc_temp_2006 = [40.9, 35.7, 43.1, 55.7, 63.1, 71.0,
77.9, 75.8, 66.6, 56.2, 51.9, 43.6]

nyc_temp_2012 = [37.3, 40.9, 50.9, 54.8, 65.1, 71.0,
78.8, 76.7, 68.8, 58.0, 43.9, 41.5]

plt.plot(months, nyc_temp_2000, months, nyc_temp_2006, months, nyc_temp_2012)
plt.title('Average monthly temperature in NYC')
plt.xlabel('Month') #x-axis label
plt.ylabel('Temperature') #y-axis label
plt.legend([2000, 2006, 2012]) #legend

plt.savefig('mygraph.png') #save in root directory
plt.show() # draw the plot on canvas
```

Add Title and Axes Descriptions!

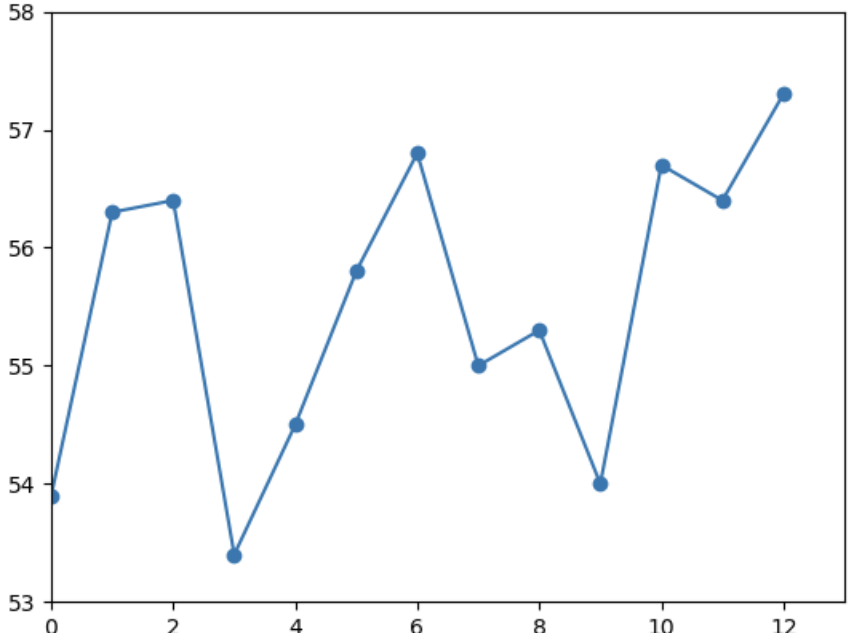
Plot the temperature in NYC aggregated by time



Changing the Field of View

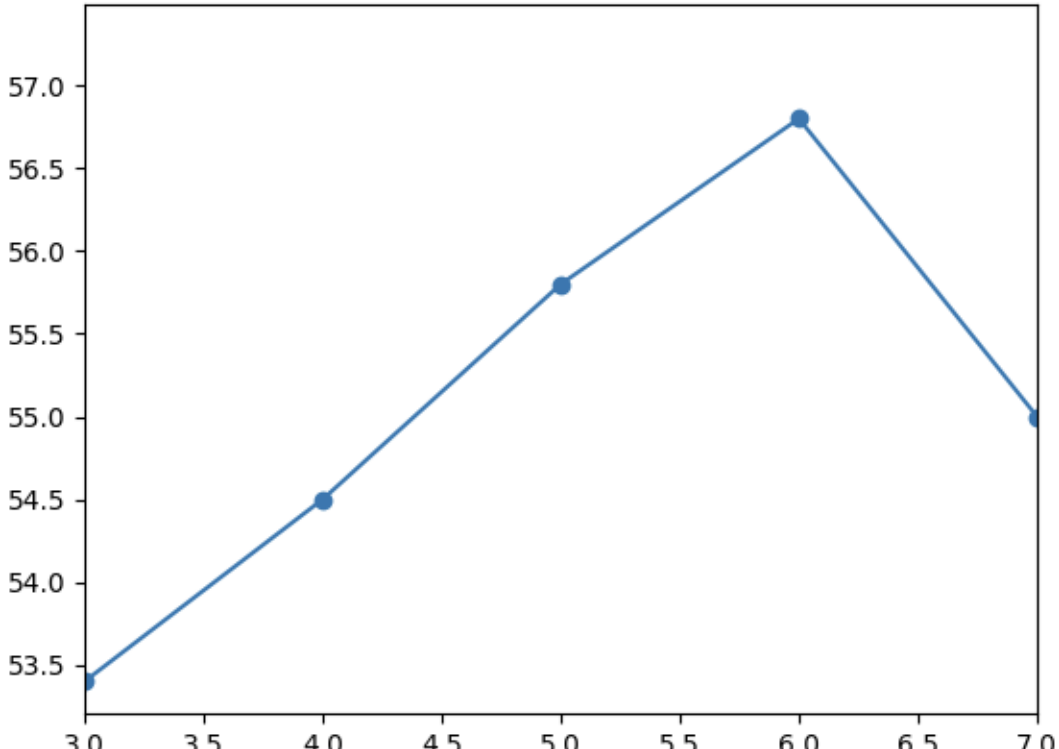
Change the axes of the plot

```
nyc_temp = [53.9, 56.3, 56.4, 53.4, 54.5, 55.8, 56.8, 55.0, 55.3, 54.0, 56.7, 56.4, 57.3]
plt.plot(nyc_temp, marker='o')
plt.axis(xmin = 0, xmax = 13, ymin = 53, ymax = 58)
plt.show()
```



COOL!!! Change the axes again to change focus!

```
plt.plot(nyc_temp, marker='o')
plt.axis(xmin = 3, xmax = 7)
plt.show()
```

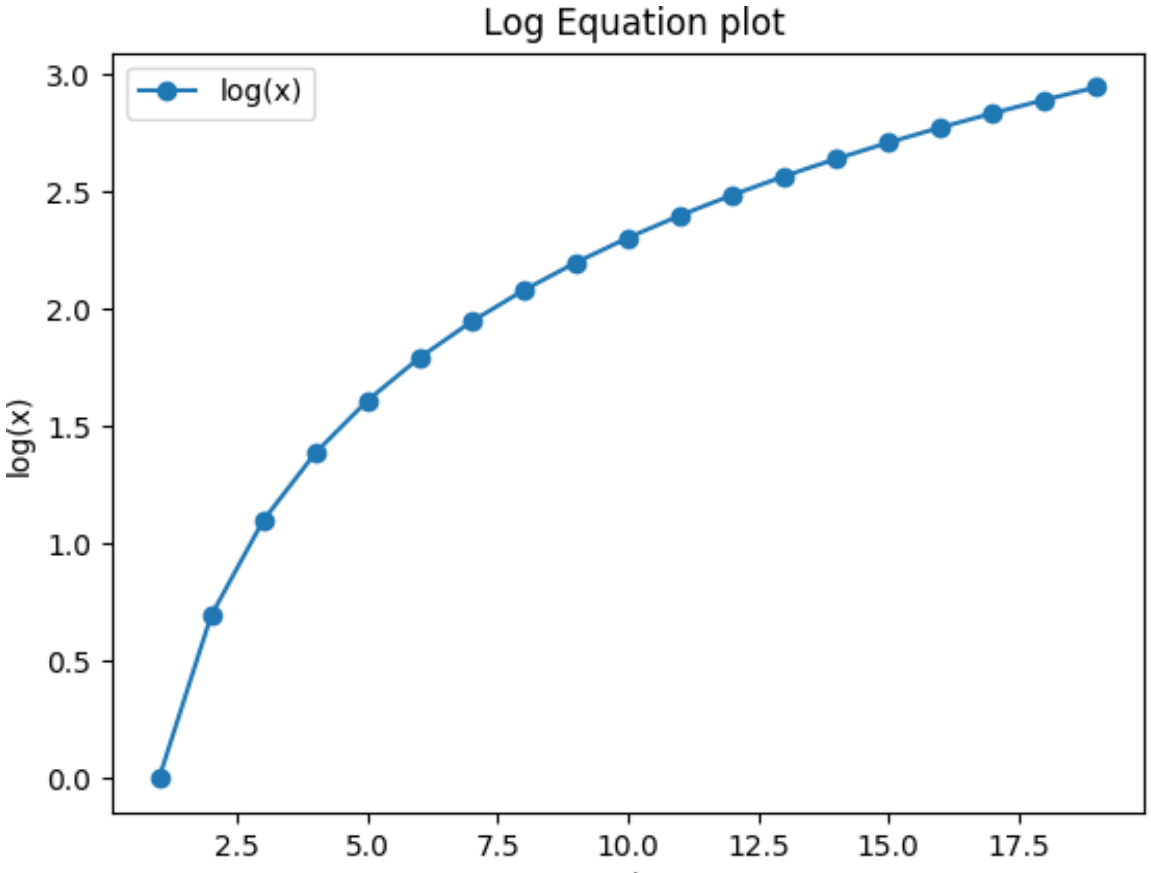


Plotting the Log Equation

```
import matplotlib.pyplot as plt
import math

x = [i for i in range(1,20)] #list x values
y = [math.log(i) for i in x] #list y values

plt.plot(x,y, marker = 'o')

plt.title(' Log Equation plot')
plt.xlabel('x Values') #x-axis label
plt.ylabel('log(x)') #y-axis label
plt.legend(['log(x)']) #legend

plt.savefig('myLogPlot.png') #save in root directory
plt.show() # draw the plot on canvas
```

The Plotted Log(x)

Plot the temperature in NYC aggregated by time



Setting Up Virtual Environment

- Create a project directory

```
mkdir projects
cd projects
```

- Create virtual environment using Python

```
python3 -m venv myenv
# see the file tree
find . -not -path '*/\.*'
```

- Activate myenv the virtual environment

```
source myenv/bin/activate  # macOS/Linux
myenv\Scripts\activate     # Windows
```

- Install Dependencies

```
pip install matplotlib
pip install numpy
```

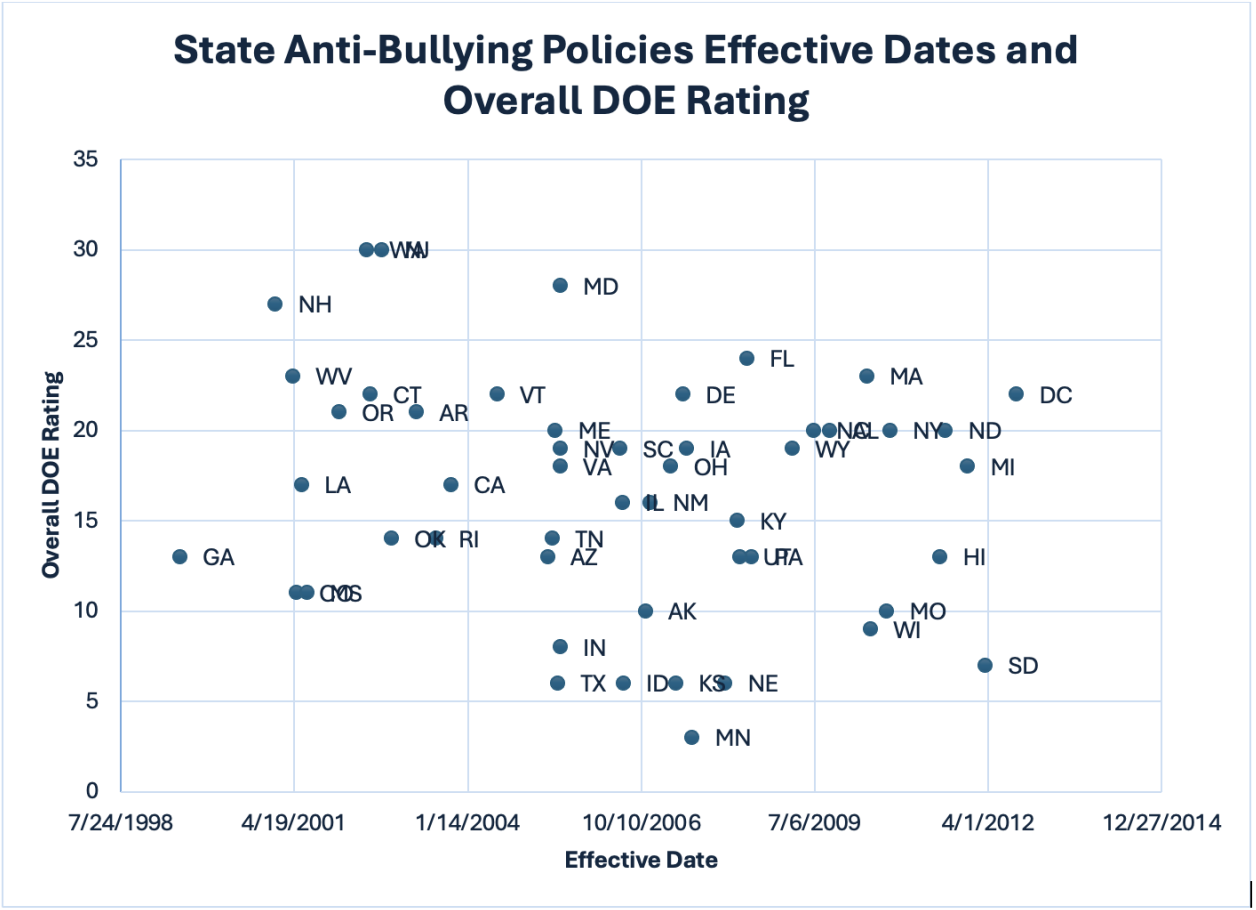
State Anti-Bullying Policies Effective Dates and Overall DOE Rating

Table 1 State Anti-Bullying Policies Effective Dates and Overall DOE Rating

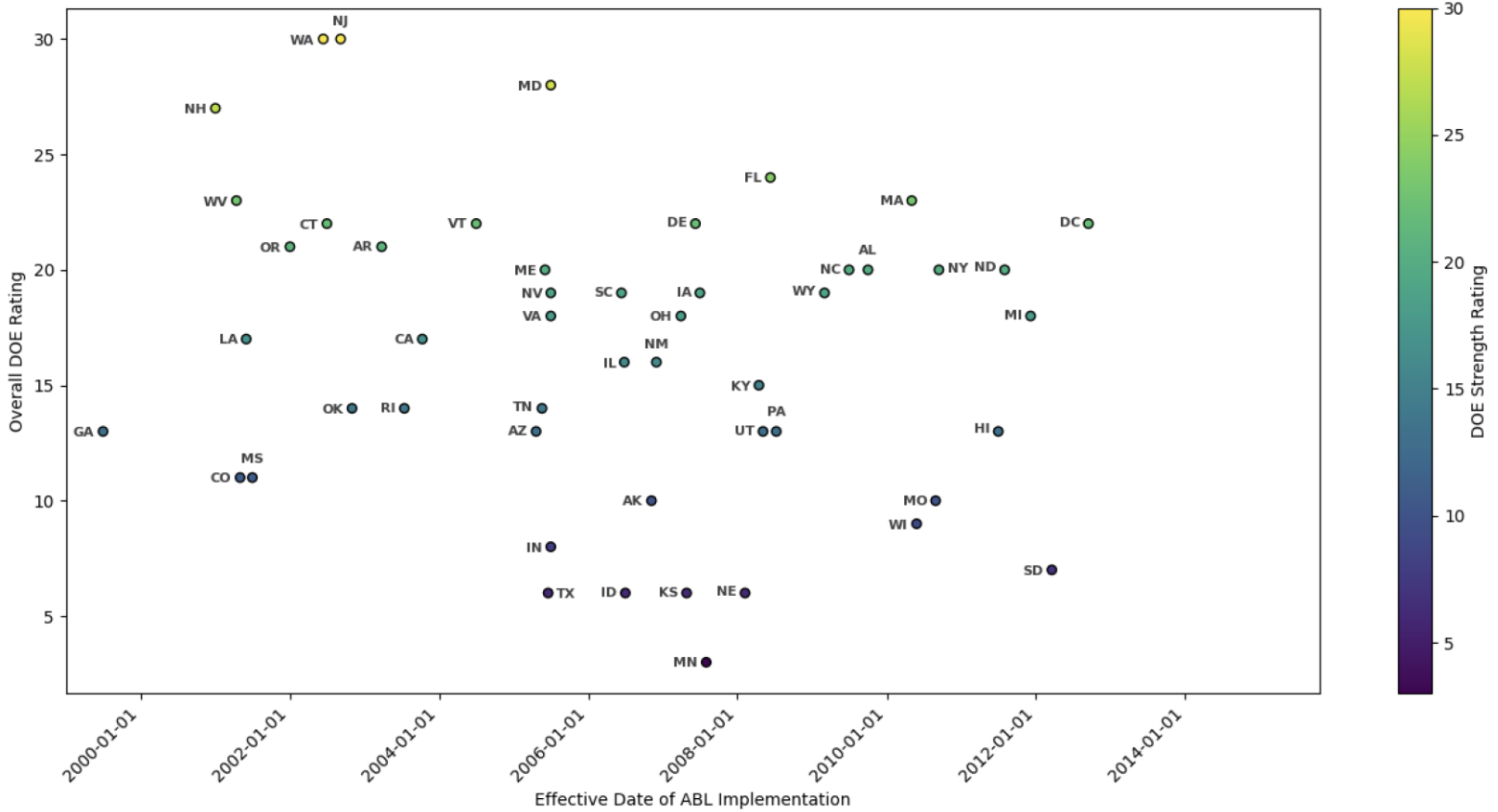
State	Effective Date	Overall DOE Rating	State	Effective Date	Overall DOE Rating
Alabama	10/01/2009	20	Montana	04/21/2015	-
Alaska	11/06/2006	10	Nebraska	02/07/2008	6
Arizona	04/20/2005	13	Nevada	07/01/2005	19
Arkansas	03/26/2003	21	New Hampshire	01/01/2001	27
California	10/11/2003	17	New Jersey	09/06/2002	30
Colorado	05/02/2001	11	New Mexico	11/30/2006	16
Connecticut	07/01/2002	22	New York	09/13/2010	20
Delaware	06/09/2007	22	North Carolina	06/30/2009	20
District of Columbia	09/14/2012	22	North Dakota	08/01/2011	20
Florida	06/10/2008	24	Ohio	03/30/2007	18
Georgia	07/01/1999	13	Oklahoma	11/01/2002	14
Hawaii	07/01/2011	13	Oregon	01/01/2002	21
Idaho	07/01/2006	6	Pennsylvania	07/09/2008	13
Illinois	06/26/2006	16	Rhode Island	07/15/2003	14
Indiana	07/01/2005	8	South Carolina	06/12/2006	19
Iowa	07/01/2007	19	South Dakota	03/19/2012	7
Kansas	04/27/2007	6	Tennessee	05/19/2005	14
Kentucky	04/15/2008	15	Texas	06/18/2005	6
Louisiana	06/01/2001	17	Utah	05/05/2008	13
Maine	06/03/2005	20	Vermont	07/01/2004	22
Maryland	07/01/2005	28	Virginia	07/01/2005	18
Massachusetts	05/03/2010	23	Washington	06/13/2002	30
Michigan	12/06/2011	18	West Virginia	04/14/2001	23
Minnesota	08/01/2007	3	Wisconsin	05/27/2010	9
Mississippi	07/01/2001	11	Wyoming	03/02/2009	19
Missouri	08/28/2010	10			

Notes: DOE: Department of Education

State Anti-Bullying Policies Effective Dates and Overall DOE Rating



State Anti-Bullying Policies Effective Dates and Overall DOE Rating



Creating Plots as files with Matplotlib

- We first need to know that the library is installed on your machine.

pip list

or

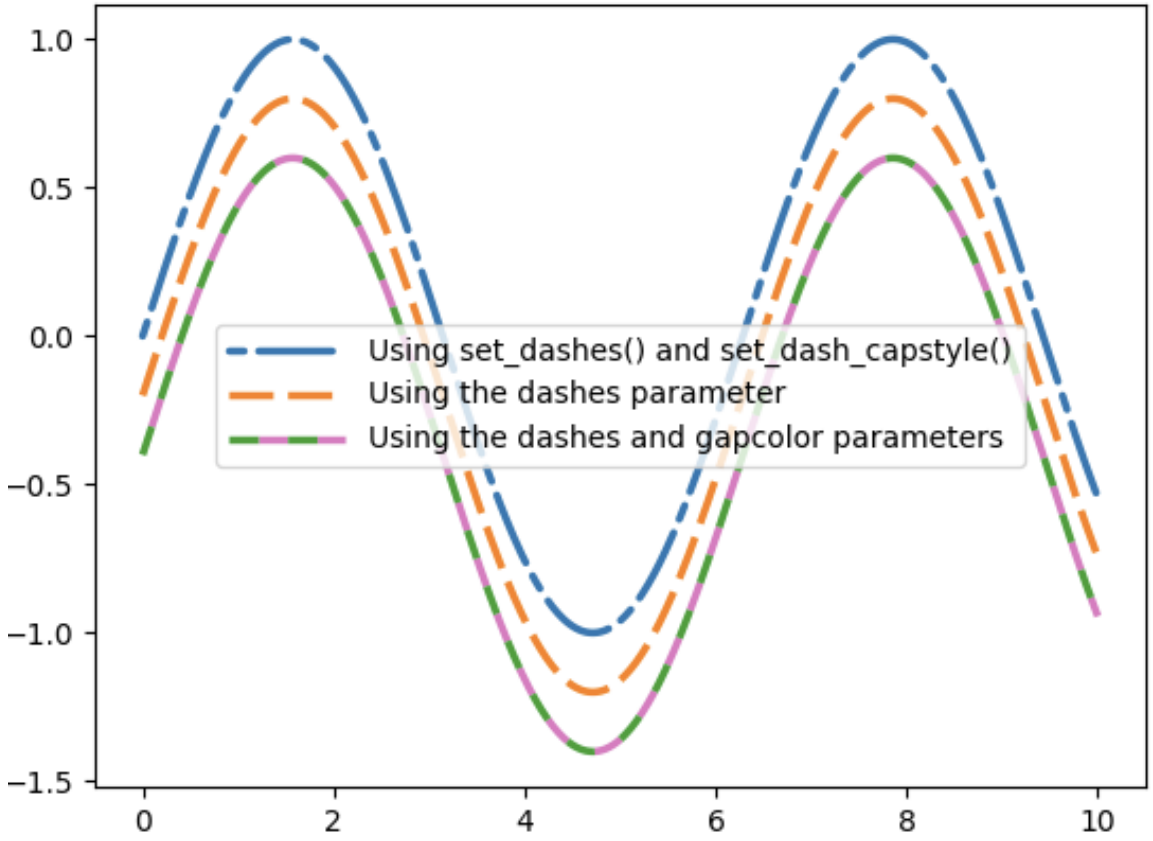
pip show matplotlib

https://matplotlib.org/

Dashed line style configuration

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

x = np.linspace(0, 10, 500)
y = np.sin(x)

plt.rc('lines', linewidth=2.5)
fig, ax = plt.subplots()

# Using set_dashes() and set_capstyle() to modify dashing of an existing line.
line1, = ax.plot(x, y, label='Using set_dashes() and set_dash_capstyle()')
line1.set_dashes([2, 2, 10, 2]) # 2pt line, 2pt break, 10pt line, 2pt break.
line1.set_dash_capstyle('round')

# Using plot(..., dashes=...) to set the dashing when creating a line.
line2, = ax.plot(x, y - 0.2, dashes=[6, 2], label='Using the dashes parameter')

# Using plot(..., dashes=..., gapcolor=...) to set the dashing and
# alternating color when creating a line.
line3, = ax.plot(x, y - 0.4, dashes=[4, 4], gapcolor='tab:pink',
label='Using the dashes and gapcolor parameters')

ax.legend(handlelength=4)
plt.show()
```

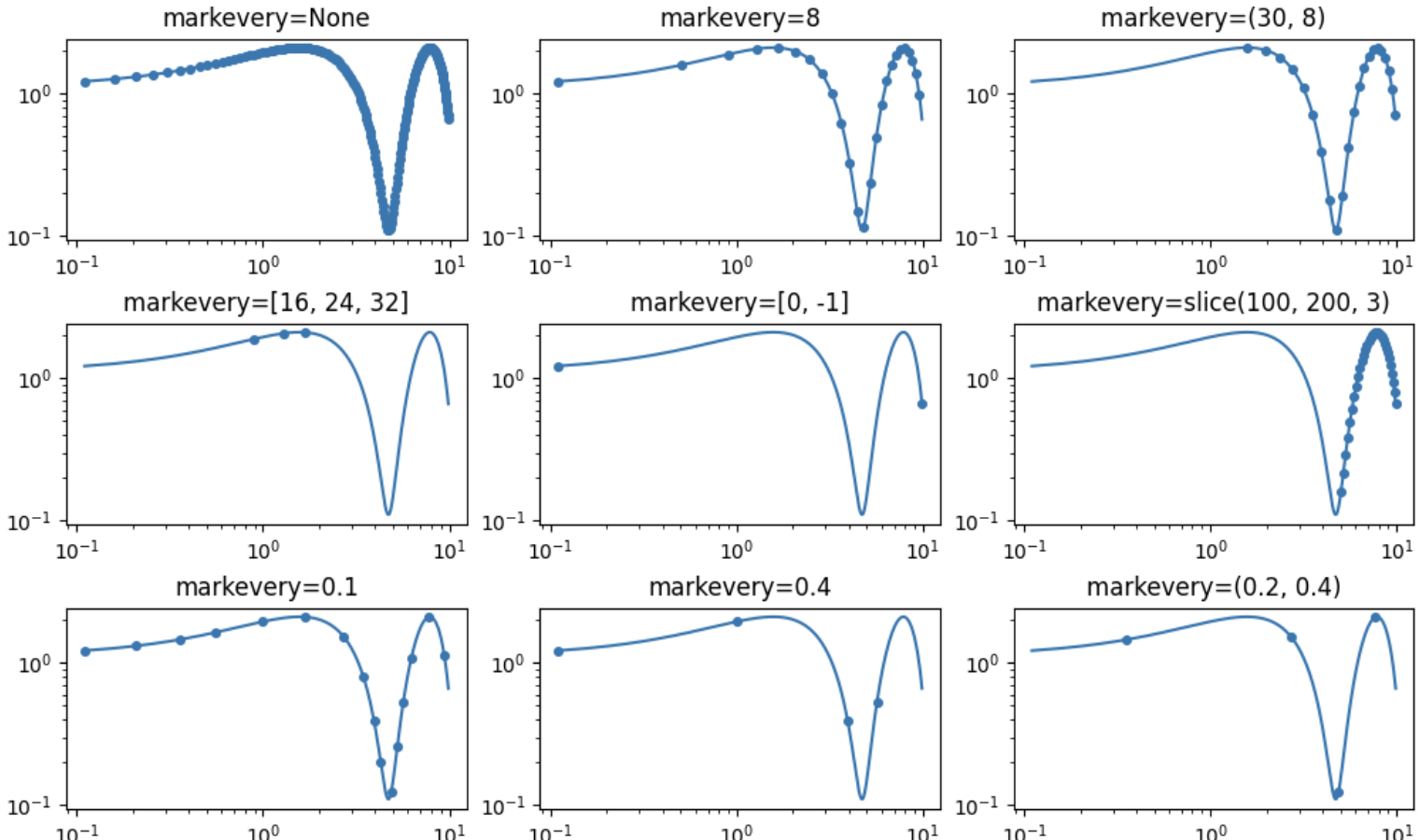
Output: Dashed line



Markevery with log scales

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

# define a list of markevery cases to plot
cases = [
None,
8,
(30, 8),
[16, 24, 32],
[0, -1],
slice(100, 200, 3),
0.1,
0.4,
(0.2, 0.4)
]

# data points
delta = 0.11
x = np.linspace(0, 10 - 2 * delta, 200) + delta
y = np.sin(x) + 1.0 + delta

fig, axs = plt.subplots(3, 3, figsize=(10, 6), layout='constrained')
for ax, markevery in zip(axs.flat, cases):
ax.set_title(f'markevery={markevery}')
ax.set_xscale('log')
ax.set_yscale('log')
ax.plot(x, y, 'o', ls='-', ms=4, markevery=markevery)

plt.show()
```

Markvery with Log Scales



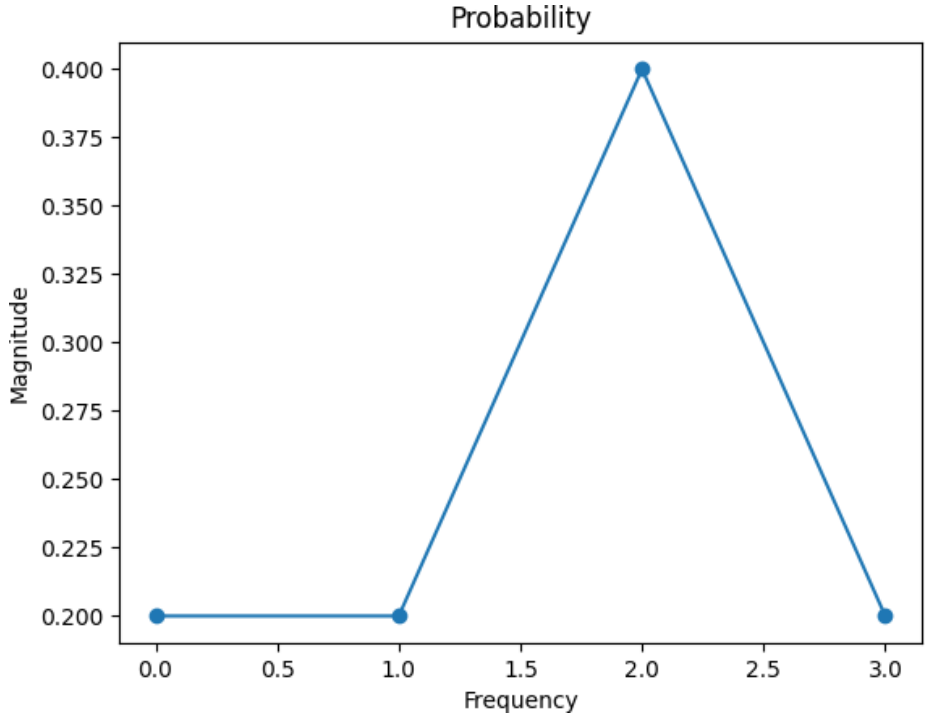
```
# simple plotting tool for frequencies of characters in a string
import matplotlib.pyplot as plt

from pylab import plot, show, title, savefig, xlabel, ylabel, legend

s_str = "hello" # string to study
sCount_dict = {} # save the counts here
# count the letters in the word
for i in s_str:
                    if i not in sCount_dict:
                                        sCount_dict[i] = 1 # add the char to the dictionary with count of one
                    else: # this char is already in the dictionary
                                        sCount_dict[i] = sCount_dict[i] + 1

print(f" Character Counts: {sCount_dict}")

freq_list = [] # list of the frquencies for the chars
for i in sCount_dict:
                    freq_list.append(sCount_dict[i]/len(s_str))
print(f" Frequencies: {freq_list}")

y = freq_list
x = [i for i in range(len(freq_list))]
plot(x,y, marker = 'o')
plt.title("Probability")
plt.ylabel('Magnitude')
plt.xlabel('Frequency')
plt.savefig('frequencyPlot.png')
# show()
```

Application: A Frequency Plotter

Source file: charPlot.py

Let's Code

Output: A Frequency Plot



String: *hello there*

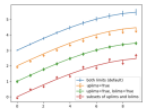
Character Counts: {'h': 1, 'e': 1, 'l': 2, 'o': 1}

Frequencies: [0.2, 0.2, 0.4, 0.2]

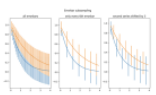
Let's Code

Now, Go Play With a Plot From the Gallery!

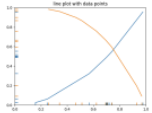
Gallery Website: https://matplotlib.org/stable/gallery/index.html



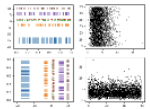
Errorbar limit selection



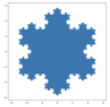
Errorbar subsampling



EventCollection Demo

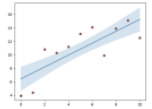


Eventplot demo

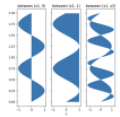


Filled polygon

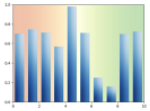
fill_between with transparency



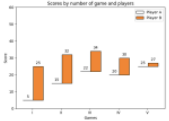
Fill the area between two lines



Fill the area between two vertical lines

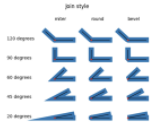


Bar chart with gradients



Hat graph

Discrete distribution as horizontal bar chart



JoinStyle