Discrete Structures!

CMPSC 102

Plots

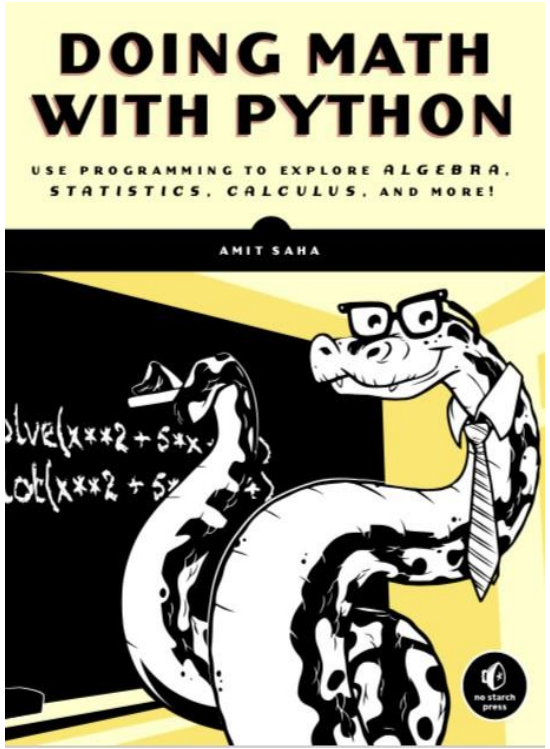
ALLEGHENY COLLEGE

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?

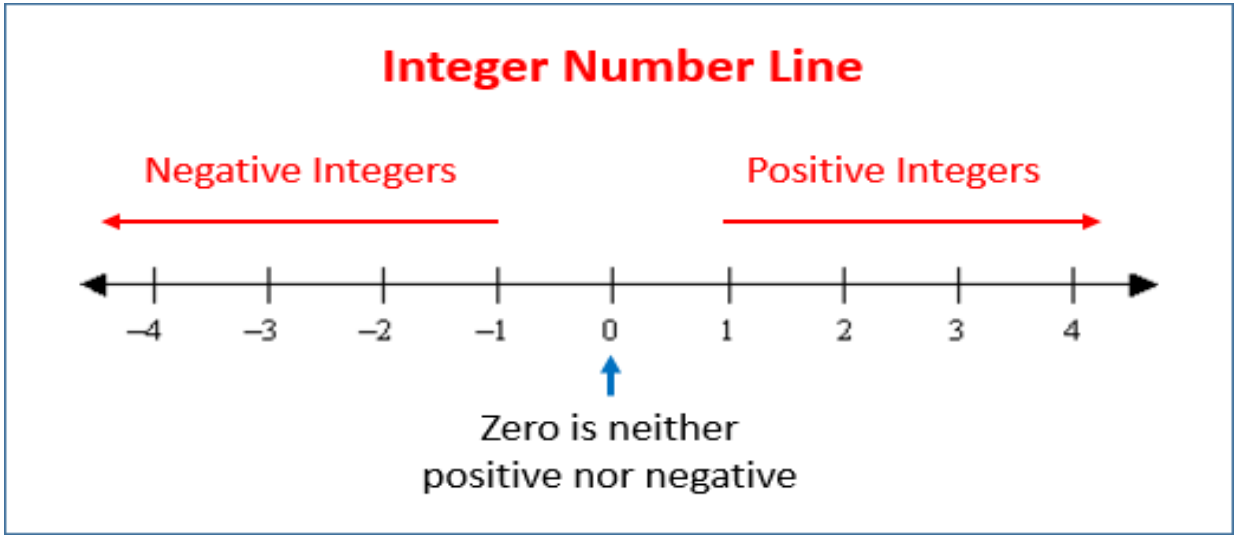


Saha, Chapter 2: Visualizing Data with graphs

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

A Number Line: x

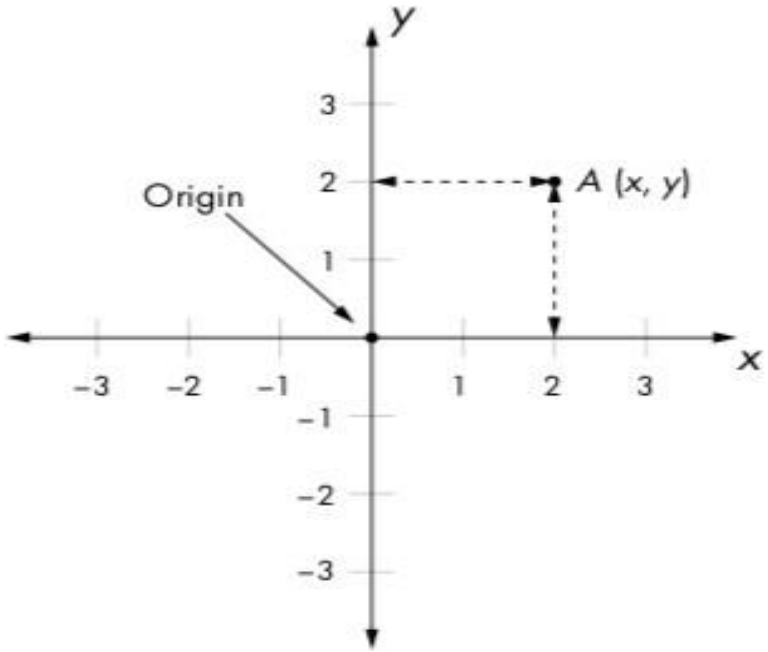
Denoted R



- 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

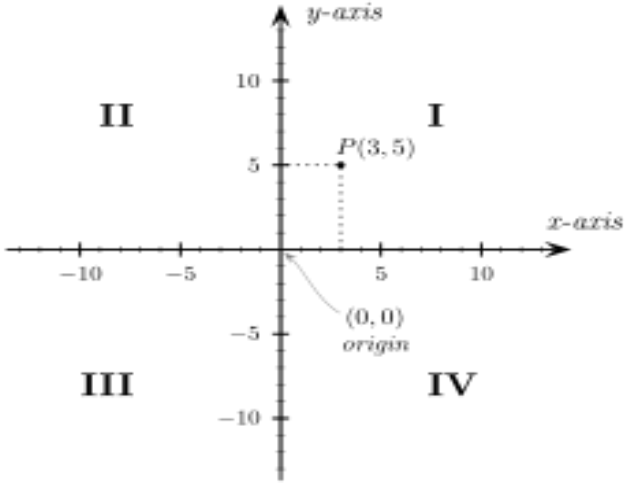
Denoted R^2



- 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

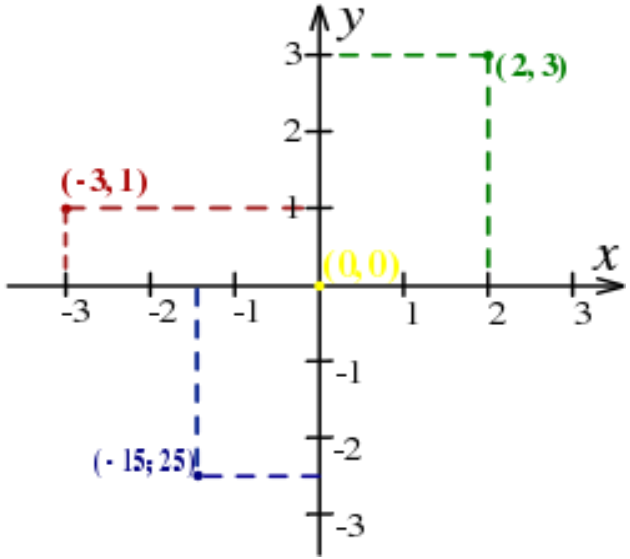


- The intersection of the values of x and y creates the 2-D point (called the ordered pair) on the canvas.
- There are four quadrants defined by:

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

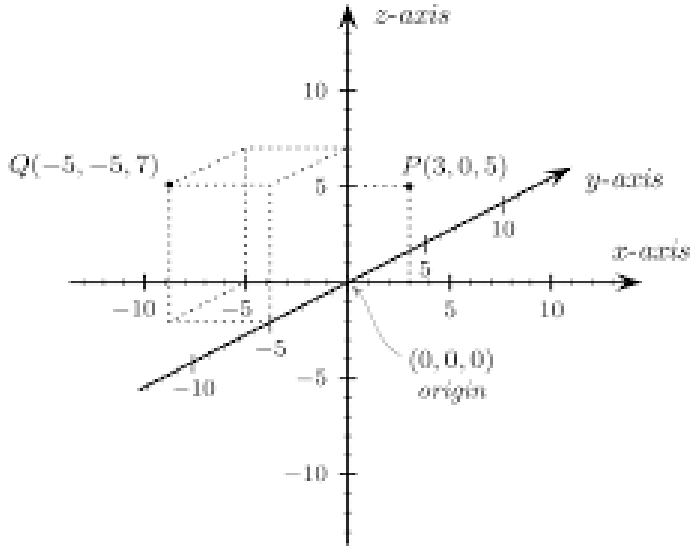
Example plot



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

3-D Coordinates: x , y , and z

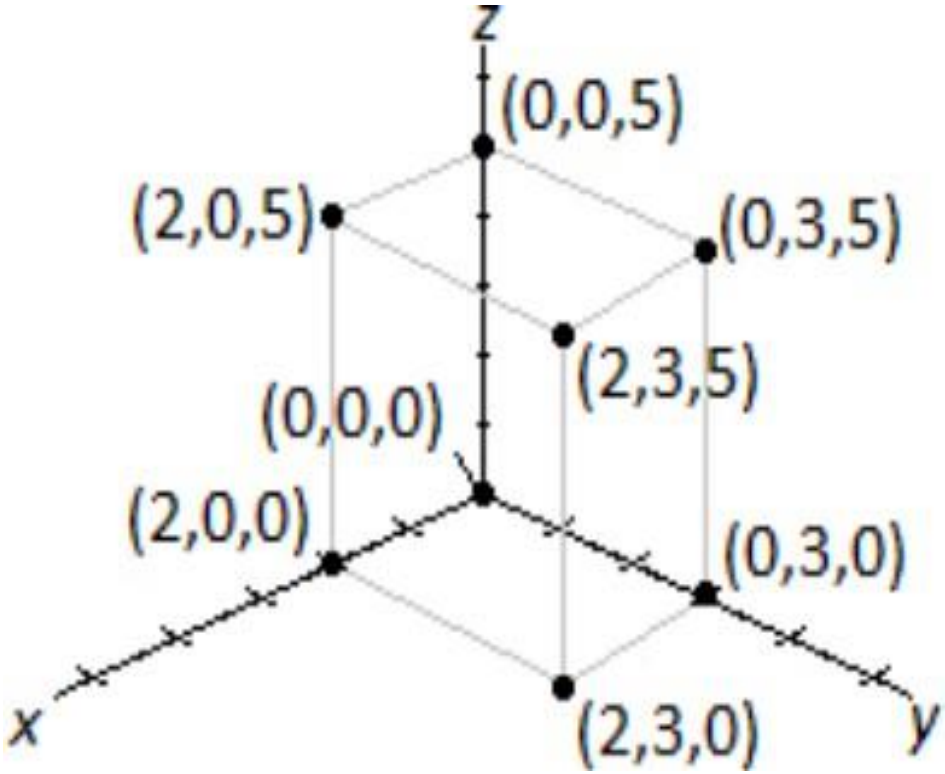
Denoted R^3



- 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
```

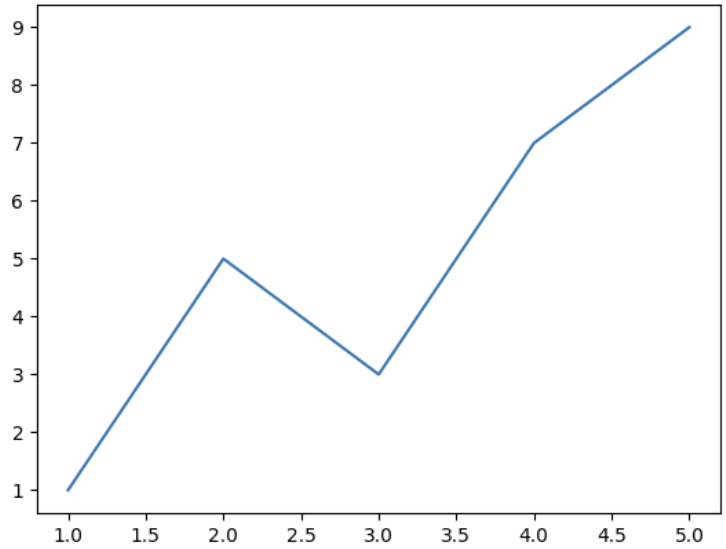
- Install Dependencies

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

Your First Plot

Plot some simple points

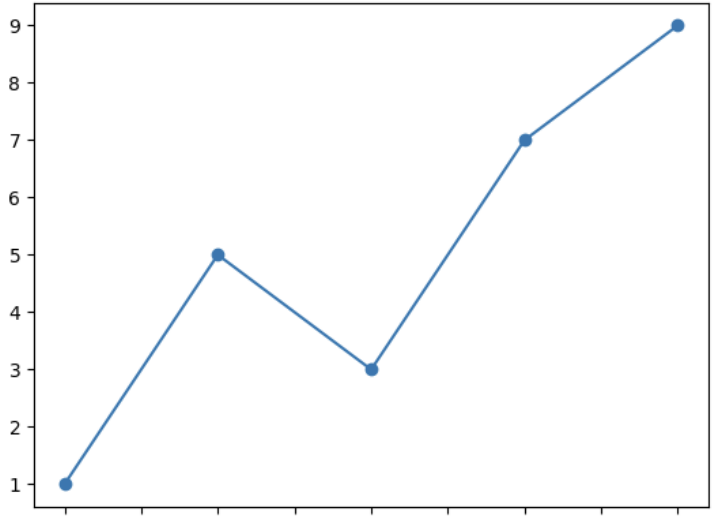
```
import matplotlib.pyplot as plt #get the library
x_num = [1,2,3,4,5] #def of x
y_num = [1,5,3,7,9] # def of y
plt.plot(x_num, y_num) # gives mem addr of obj
plt.show() # draw the plot on canvas
```



Gimme Points, Not Lines

Plot some basic numbers using points

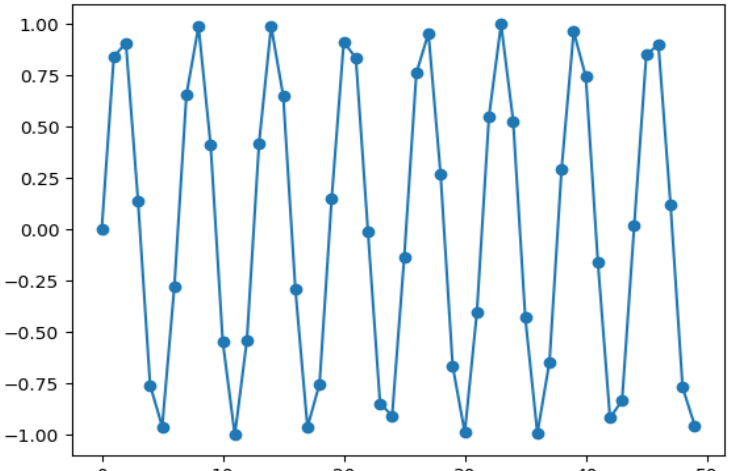
```
import matplotlib.pyplot as plt #get the library
x_num = [1,2,3,4,5] #def of x
y_num = [1,5,3,7,9] # def of y
plt.plot(x_num, y_num, marker='o')
# also including 'o', '*', 'x', and '+' as points
plt.show() # draw the plot on canvas
```



Another Amazing Example!

Plot the sin wave

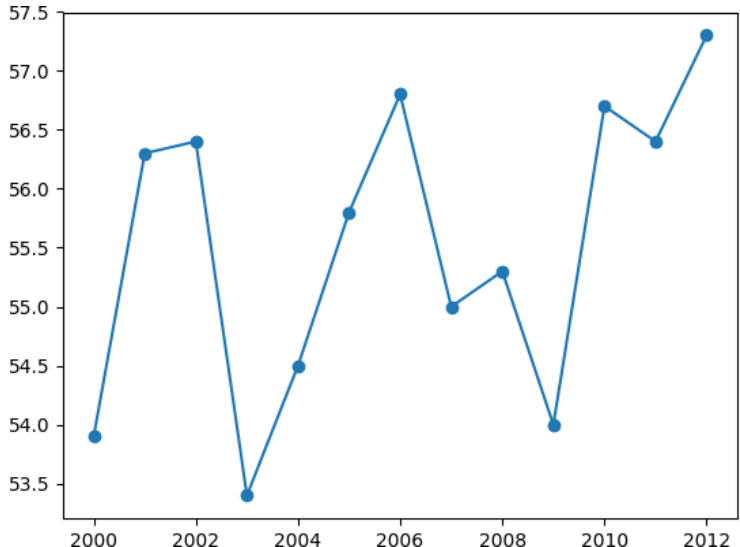
```
import matplotlib.pyplot as plt #get the library
import math
x_num = [i for i in range(50)]
y_num = [math.sin(i) for i in x_num]
plt.plot(x_num, y_num, marker='o')
# also including 'o', '*', 'x', and '+' as points
plt.show() # draw the plot on canvas
```



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
```



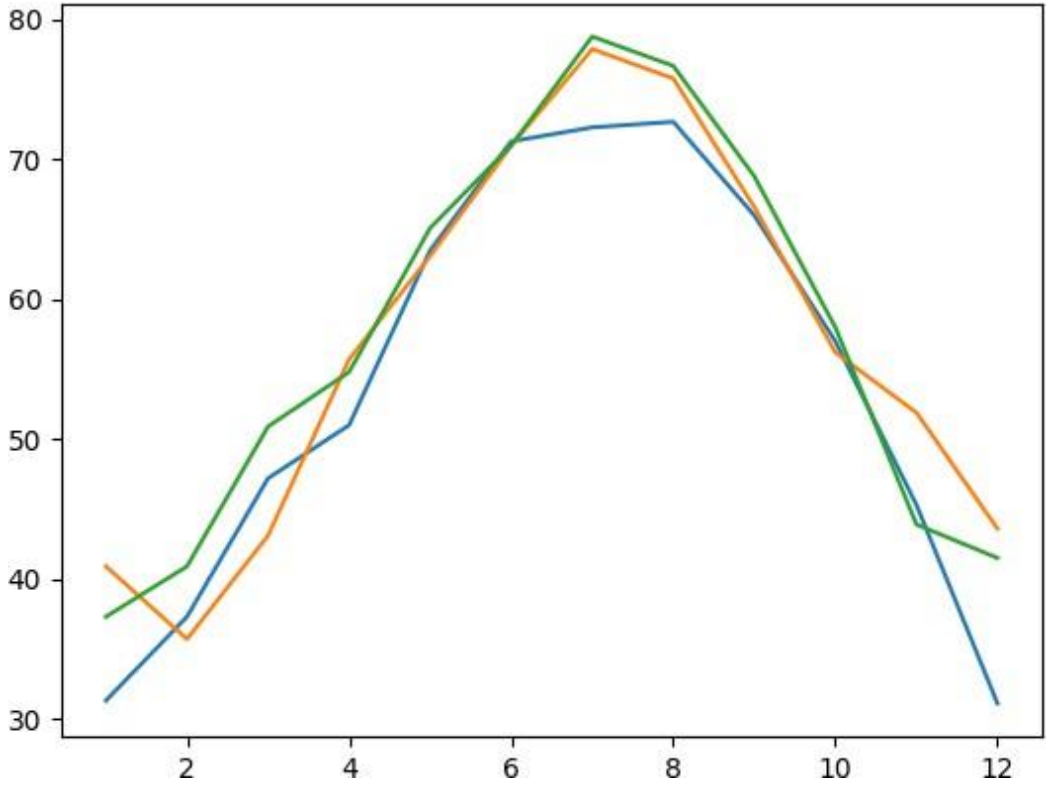
Three Plots Together! Amazing!

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



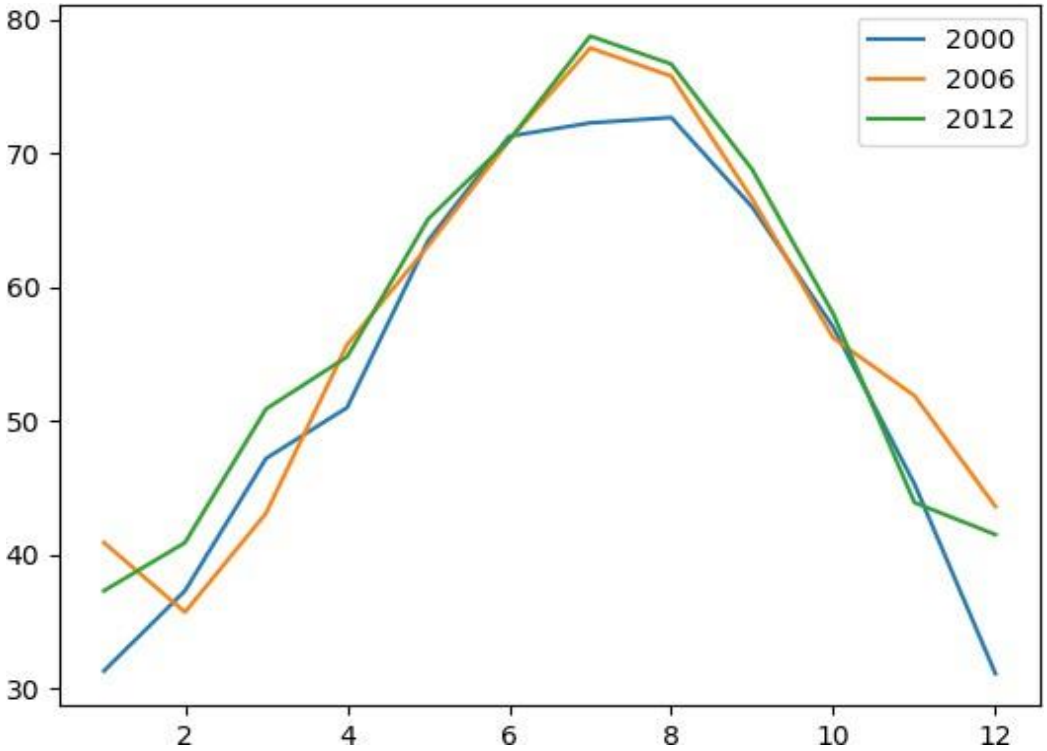
Three Plots Together! And a LEGEND Too!

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



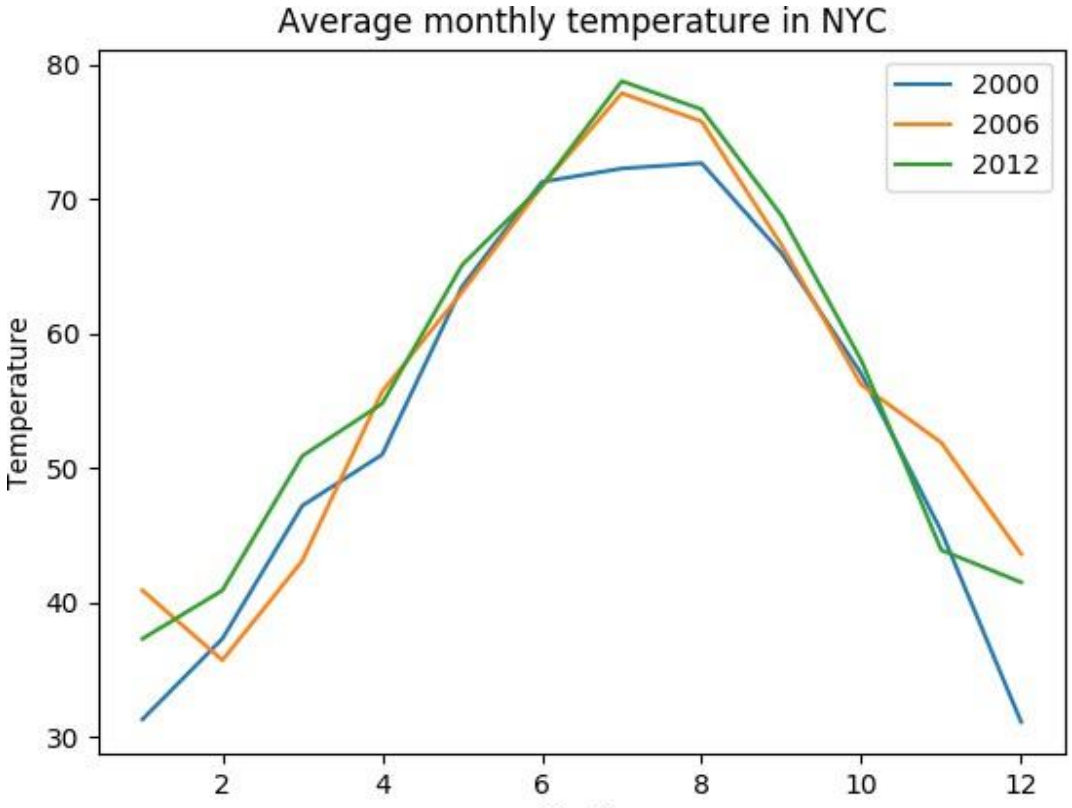
Add Title and Axes Descriptions!

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!

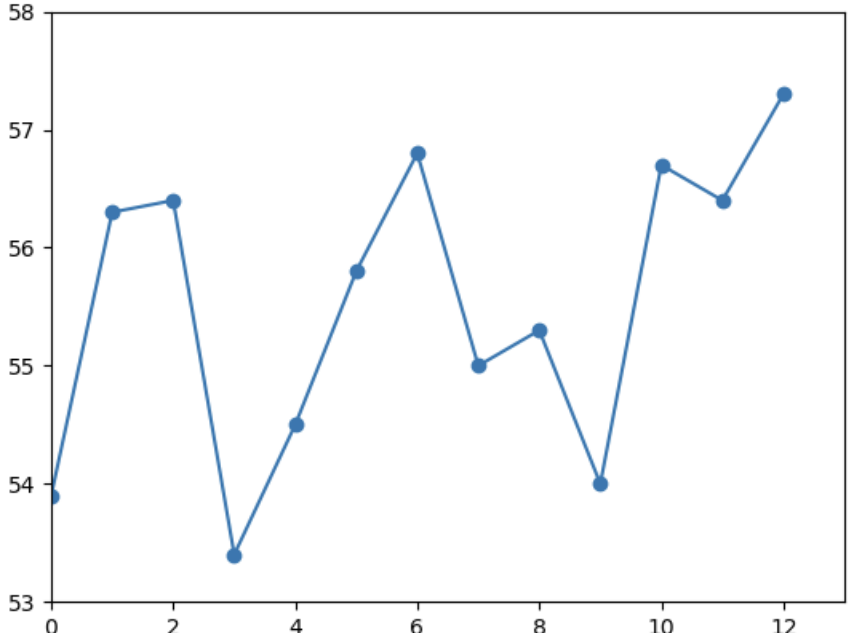
Plot the temperature in NYC aggregated by time



Changing the Field of View

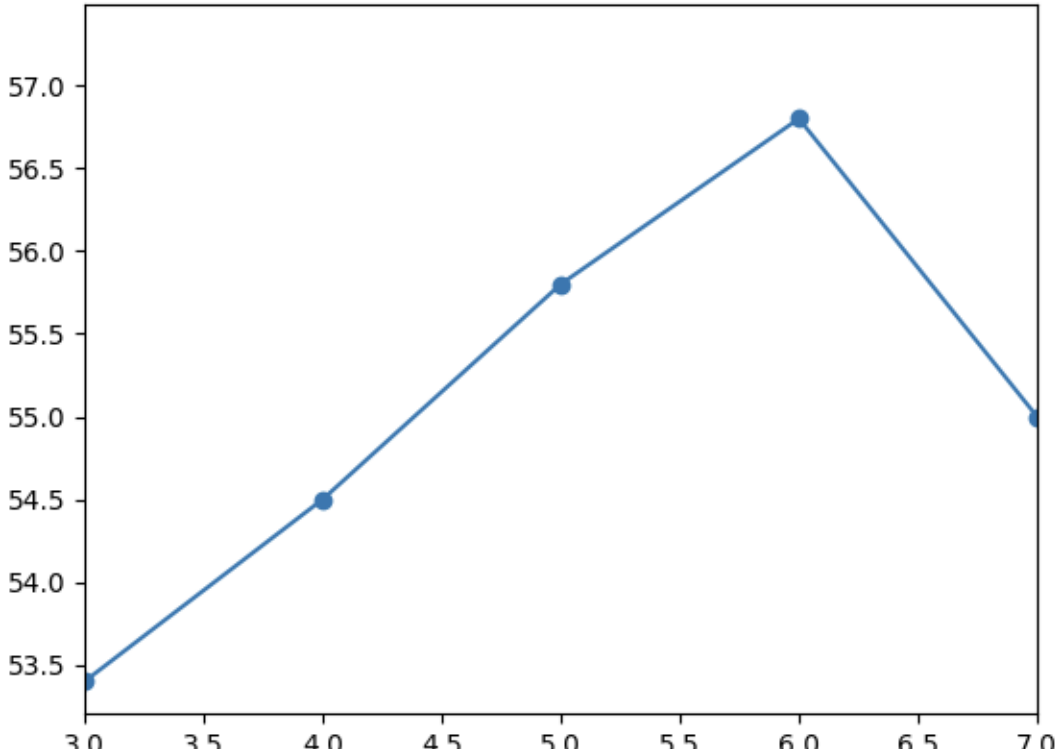
Change the axes of the plot

```
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()
```

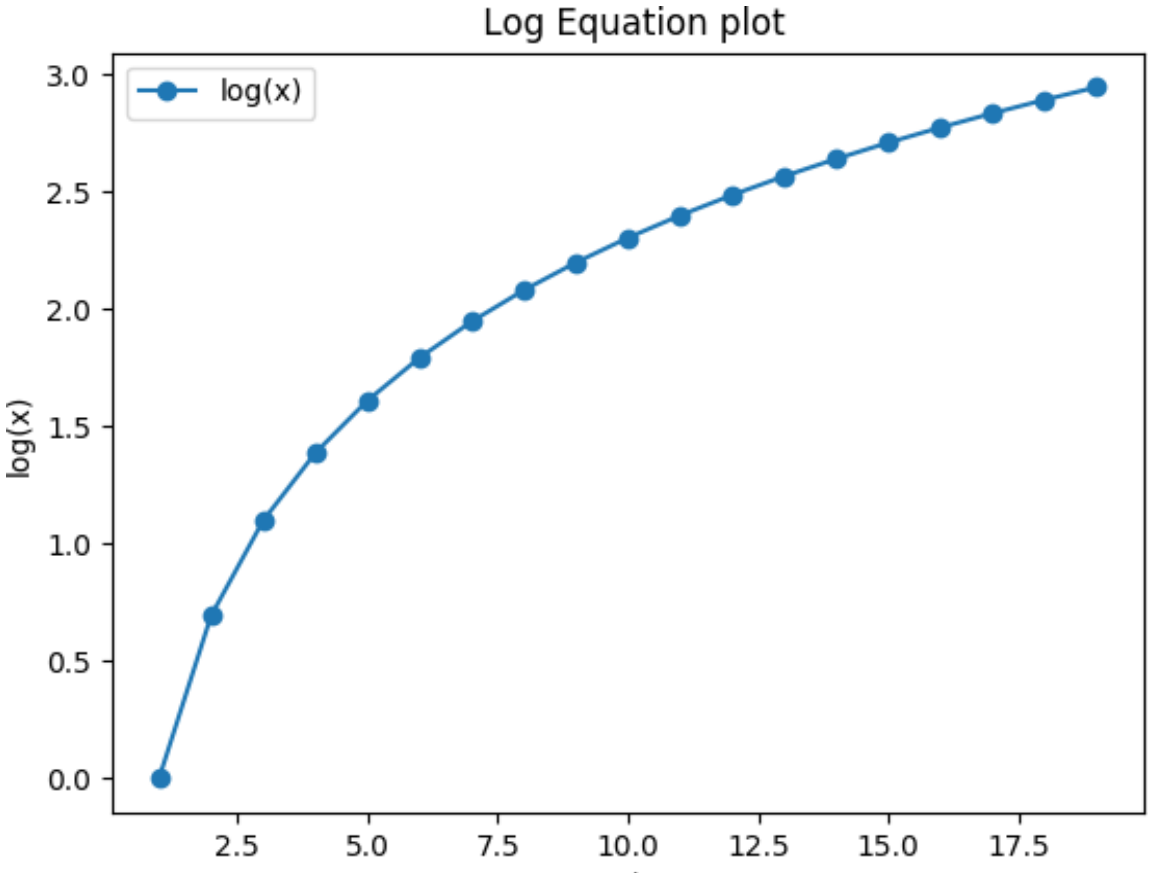


Plotting the Log Equation

```
import matplotlib.pyplot as plt
import math

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

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 '*/\.*'
```

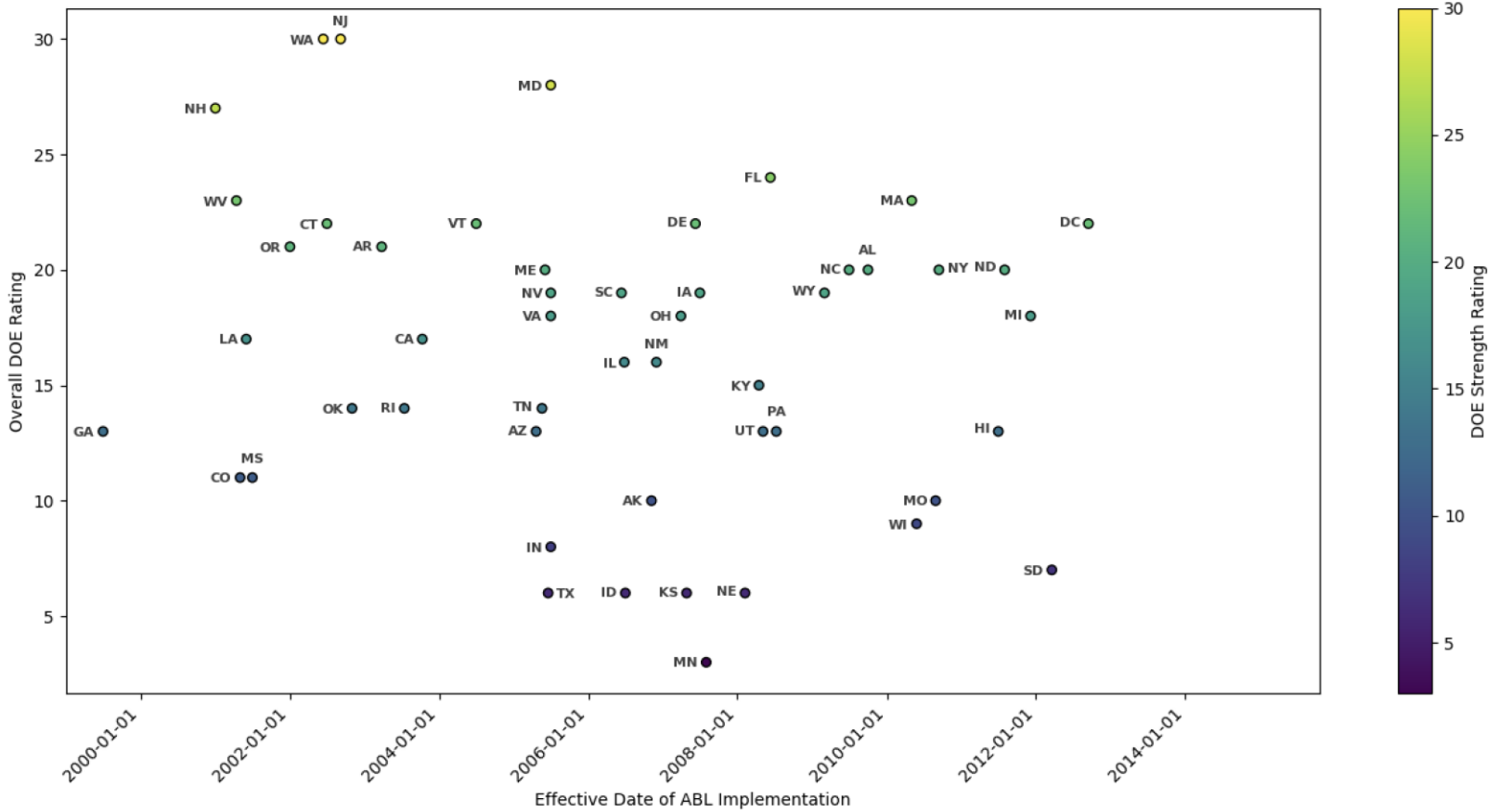
- Activate myenv the virtual environment

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

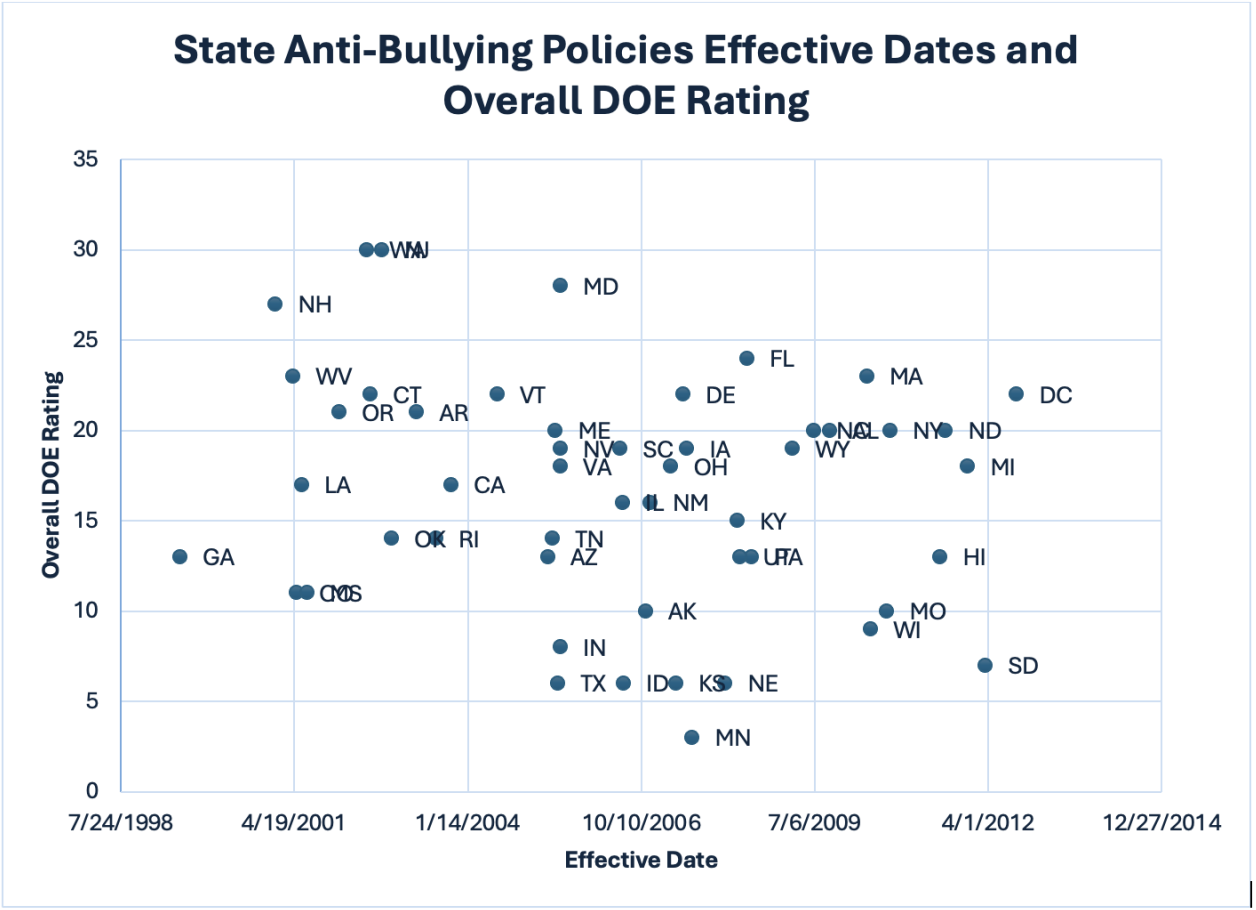
- Install Dependencies

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

State Anti-Bullying Policies Effective Dates and Overall DOE Rating



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.

```
python3 from pylab import plot, show
```

https://matplotlib.org/

Koch Snowflakes

Source file: kochSnowflake.py

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

def koch_snowflake(order, scale=10):
        """ class to drive the program """
        def koch_snowflake_complex(order):
                if order == 0:
                        # initial triangle
                        angles = np.array([0, 120, 240]) + 90
                        return scale / np.sqrt(3) * np.exp(np.deg2rad(angles) * 1j)
                else:
                        ZR = 0.5 - 0.5j * np.sqrt(3) / 3
                        p1 = koch_snowflake_complex(order - 1) # start points
                        p2 = np.roll(p1, shift=-1) # end points
                        dp = p2 - p1 # connection vectors
                        new_points = np.empty(len(p1) * 4, dtype=np.complex128)
                        new_points[::4] = p1
                        new_points[1::4] = p1 + dp / 3
                        new_points[2::4] = p1 + dp * ZR
                        new_points[3::4] = p1 + dp / 3 * 2
                        return new_points

        # end of koch_snowflake_complex()
        points = koch_snowflake_complex(order)
        x, y = points.real, points.imag
        return x, y

# end of koch_snowflake() class
```

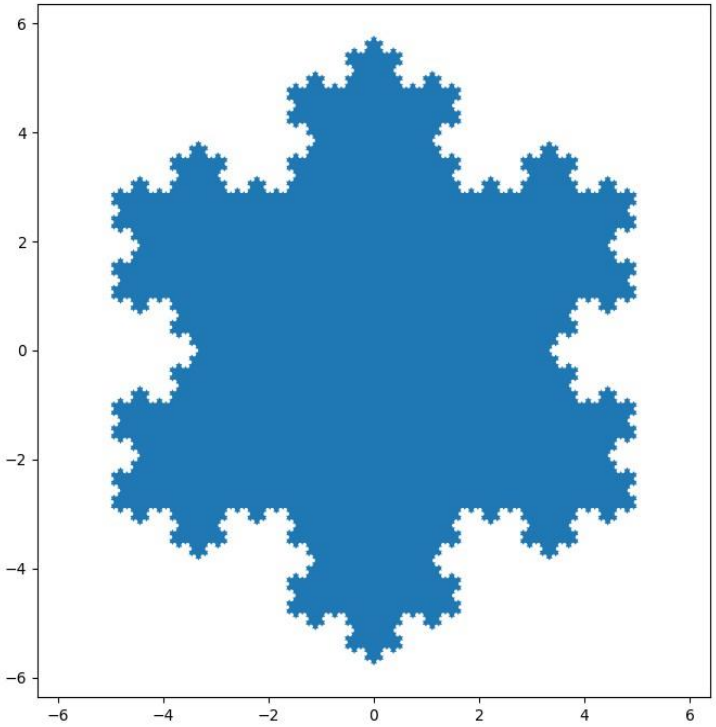
Koch Snowflakes

Source file: kochSnowflake.py

```
def oneStar() -> None:
                """" generate one star """"
                x, y = koch_snowflake(order = 5) # thhe order is recursion dept
                plt.figure(figsize=(8, 8))
                plt.axis('equal')
                plt.fill(x, y)
                plt.savefig('koch_oneStar.png')
                #plt.show()
# end of oneStar()

def threeStars() -> None:
                """" generate one star """"
                x, y = koch_snowflake(order=2)
                fig, (ax1, ax2, ax3) = plt.subplots(1, 3, figsize=(9, 3),
                                                                subplot_kw={'aspect': 'equal'})
                ax1.fill(x, y)
                ax2.fill(x, y, facecolor='lightsalmon', edgecolor='orangered', linewidth=3)
                ax3.fill(x, y, facecolor='none', edgecolor='purple', linewidth=3)

                plt.savefig('koch_threeStars.png')
                #plt.show()

# end of threeStars()

def main() -> None:
                oneStar()
                threeStars()
main()
```

Output: The Koch Snowflake

Source file: kochSnowflake.py



A Number Line: x

Denoted R

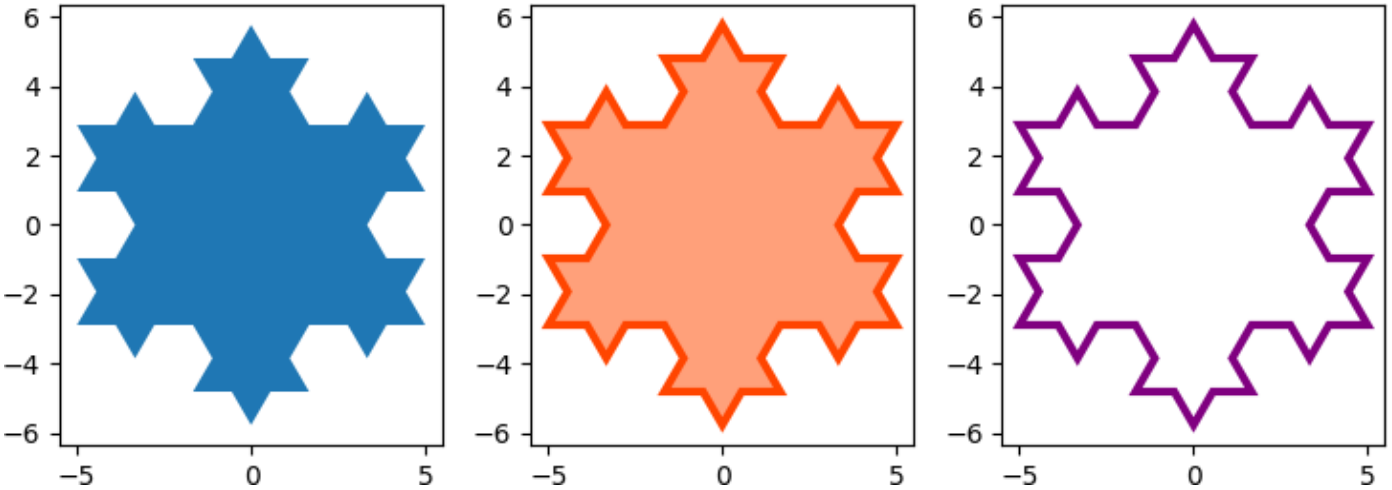


Figure: Three Koch stars as output.

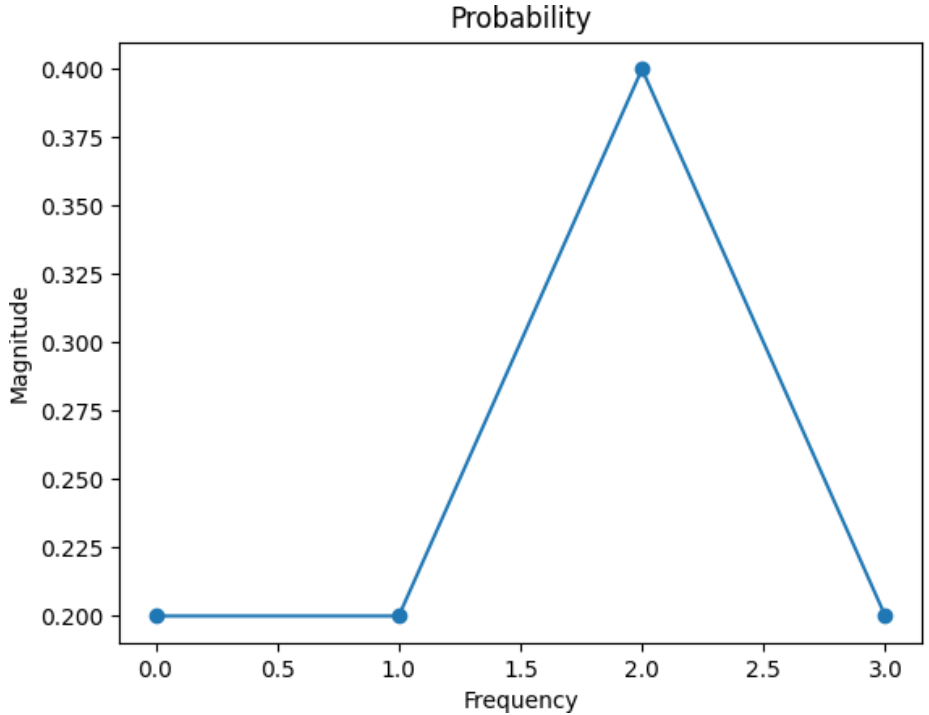
```
# simple plotting tool for frequencies of characters in a string
import matplotlib.pyplot as plt

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*

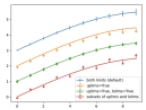
Character Counts: {'h': 1, 'e': 1, 'l': 2, 'o': 1}

Frequencies: [0.2, 0.2, 0.4, 0.2]

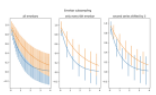
Let's Code

Now, Go Play With a Plot From the Gallery!

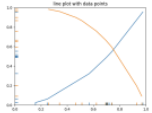
Gallery Website: https://matplotlib.org/stable/gallery/index.html



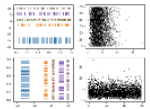
Errorbar limit selection



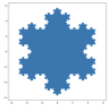
Errorbar subsampling



EventCollection Demo

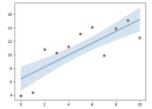


Eventplot demo

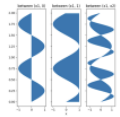


Filled polygon

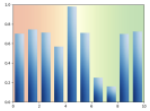
fill_between with transparency



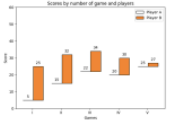
Fill the area between two lines



Fill the area between two vertical lines

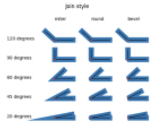


Bar chart with gradients



Hat graph

Discrete distribution as horizontal bar chart



JoinStyle