Discrete Structures: CMPSC 102

Oliver BONHAM-CARTER

Fall 2019
Week 11

Quiz 2 is Coming Up

Please note

Quiz 2 is coming up Friday 15^{th} November

- Quiz 2 topics to come on Wednesday
- Start going through your slides and notes!

Quiz 2

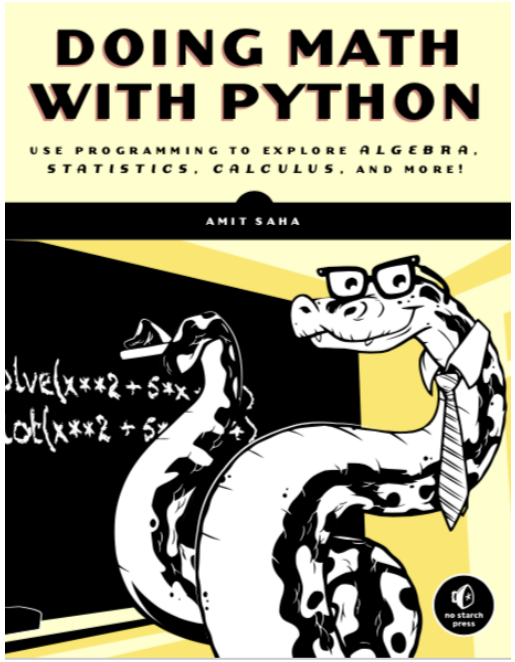
- Given on Friday 15^{th} November during class time (11am)
- Online format
- One hour to complete; although more time is given if necessary
- Around Fifteen questions: Multi-choice, True/False, Matching and Short answer
- Code: Picking out bugs from code or determining output

What to study

- **Slides, notes, with chapters to add detail to class material**
- Main ideas since Exam 1 and associated samples of code
- Graph Theory
  - Explain the initial (*first*) problem of graph theory
  - Terms: *adjacency*, vertex *degrees*, *isolated nodes*, *order*, *size*, *paths*
  - Applications of graphs
- Objects and Classes
  - Functions versus Methods
  - Recognizing correct class syntax in code
  - Classes and their variables, as opposed to root variables
  - *Self* variables versus non-*self* variables
  - The `__init__` method and the concept of *docstrings*
- Conceptual questions concerning the use of classes and building plots

Where Are We Now?

Saha's Book

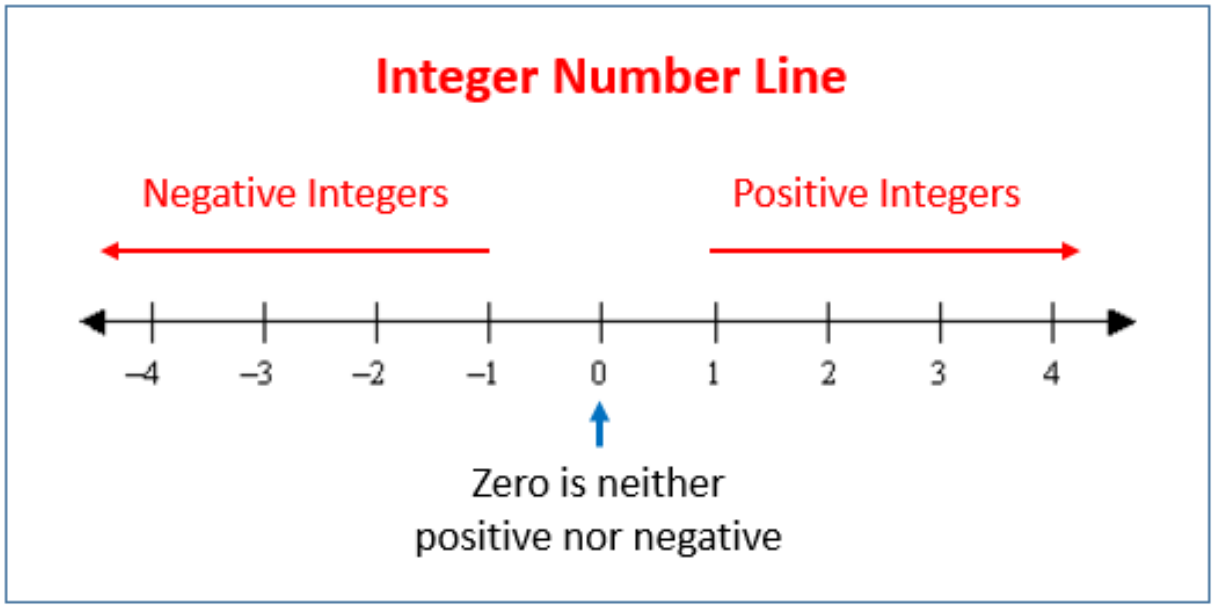


Saha, Chapter 2: Visualizing Data with graphs

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

A Number Line: x

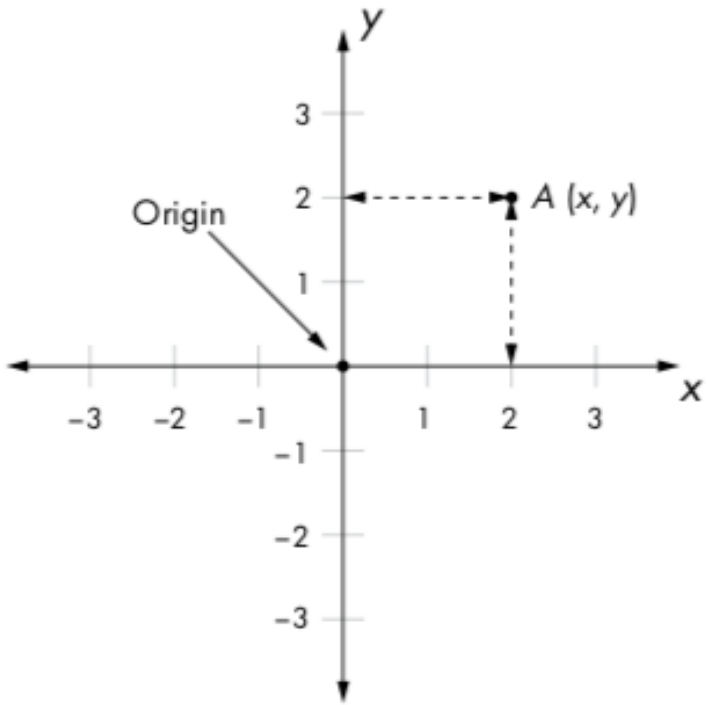
Denoted R



- 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

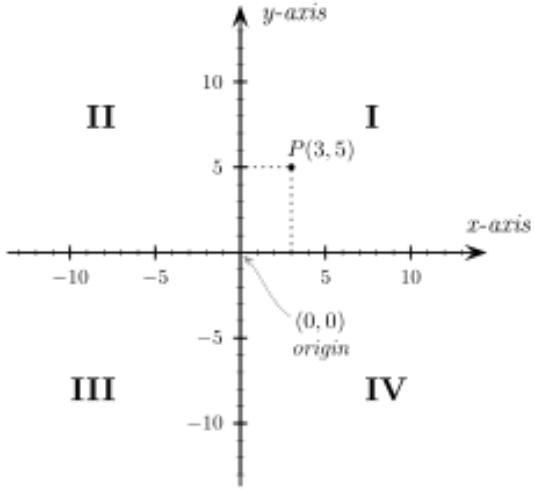
Denoted R^2



- 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

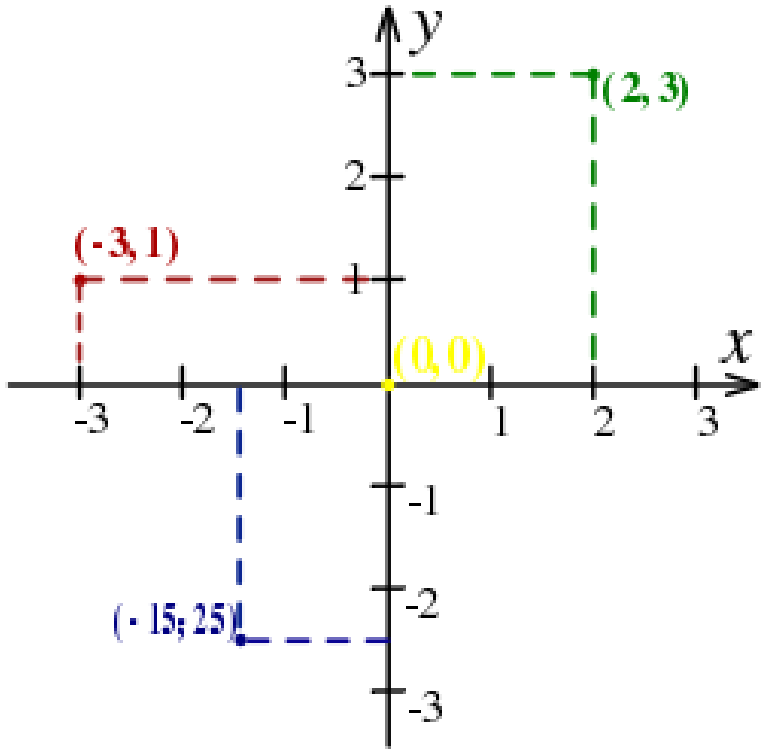
Denoted R^2



- The two number lines are called the x-axis and the y-axis and are called the *coordinate axes*
- 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

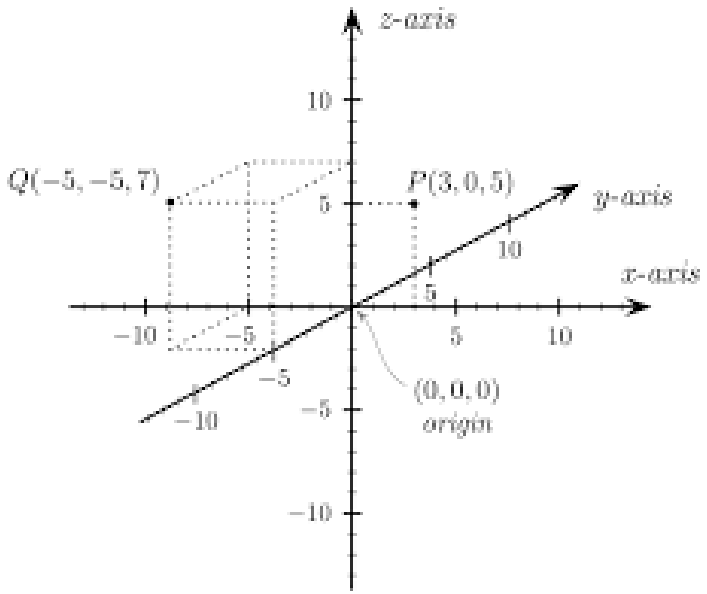
Example plot



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

3-D Coordinates: x, y, and z

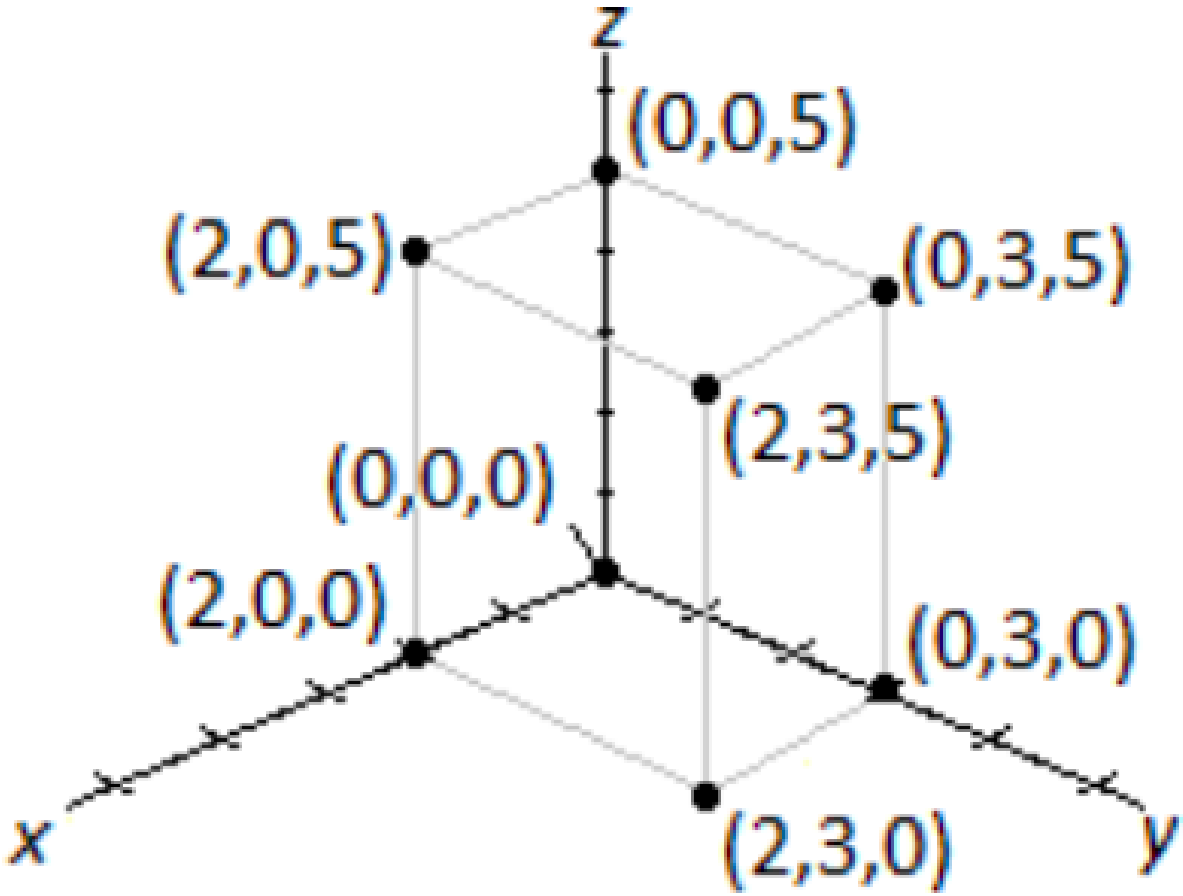
Denoted R^3



- 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.
- The z-axis:

3-D Coordinates: x, y, and z

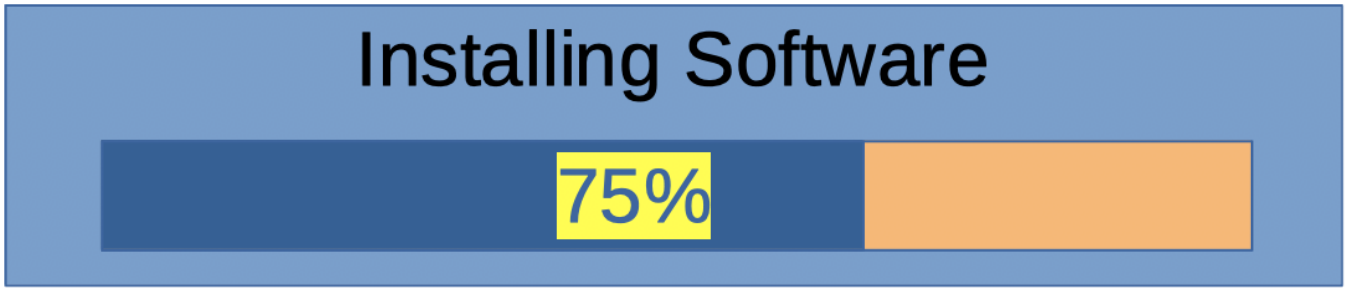
Example plot



Matplotlib

- Matplotlib is a Python 2D 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

Installing Matplotlib



Website

```
https://matplotlib.org/3.1.1/users/installing.html
```

Installation Commands from Bash or Command Prompt

```
python -m pip install -U pip # install PIP
python -m pip install -U matplotlib #install Matplotlib core
```

Checking the Version

```
import matplotlib
matplotlib.__version__ # '3.1.1'
```

Creating Plots with Matplotlib

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

python3

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

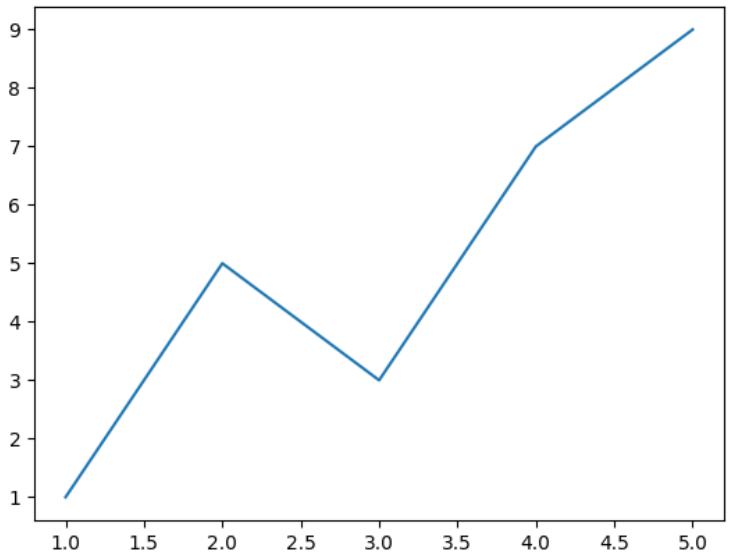
- https://matplotlib.org/index.html
- https://matplotlib.org/3.0.0/users/installing.html

Your First Plot

Plot some simple points

Place in python3 or in a python3 program file

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

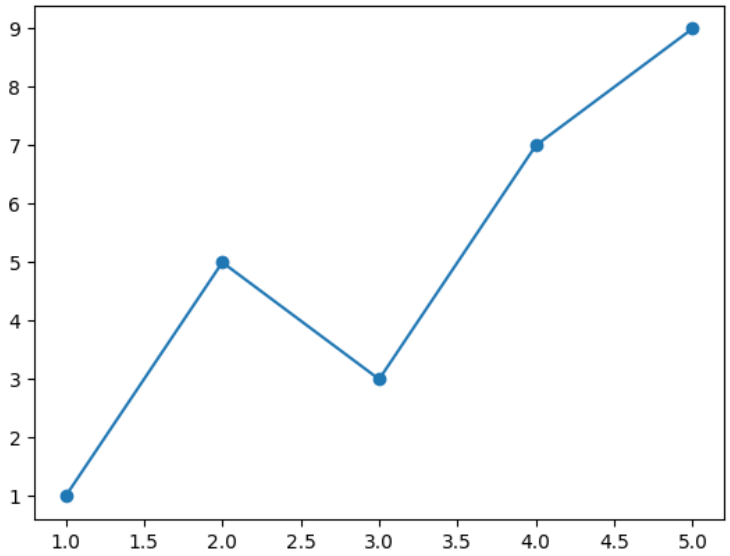


Gimme Points, Not Lines

Plot some basic numbers using points

Place in python3 or in a python3 program file

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

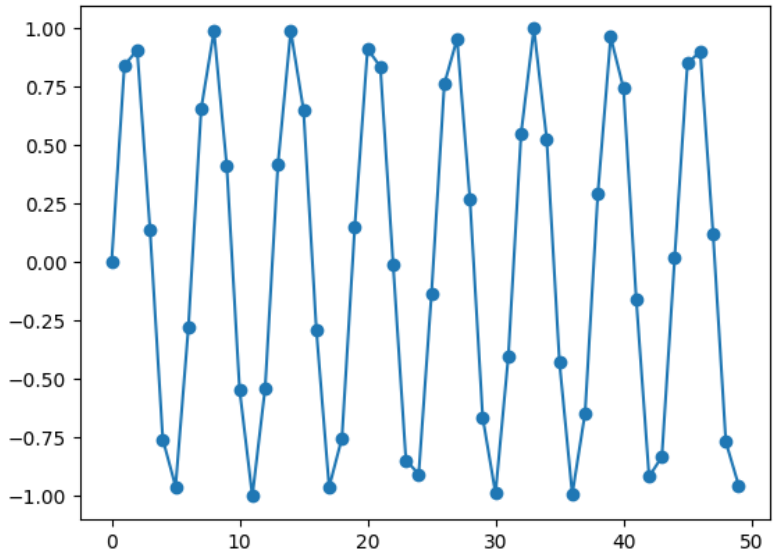


Another Amazing Example!

Plot the sin wave

Place in python3 or in a python3 program file

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

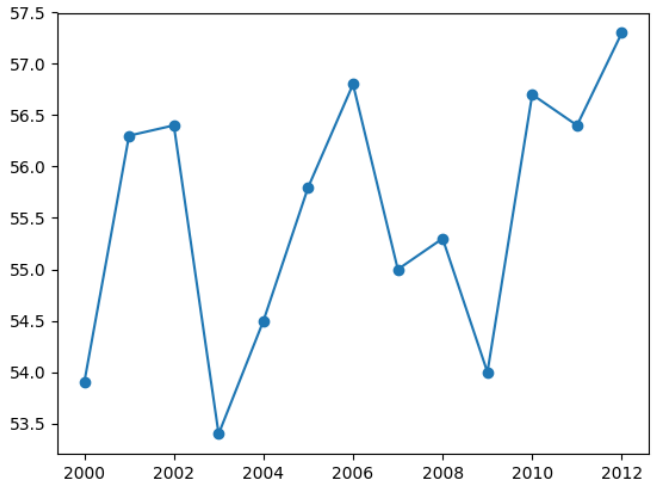


Yet, **Another** Amazing Example!

Plot the temperature in NYC and save the file too!

Place in python3 or in a python3 program file

```
from pylab import plot, show, savefig #note savefig
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)
plot(years, nyc_temp, marker='o')
# also including 'o', '*', 'x', and '+' as points
savefig('mygraph.png') #save in root directory
show() # draw the plot on canvas
```



Three Plots Together! Amazing!

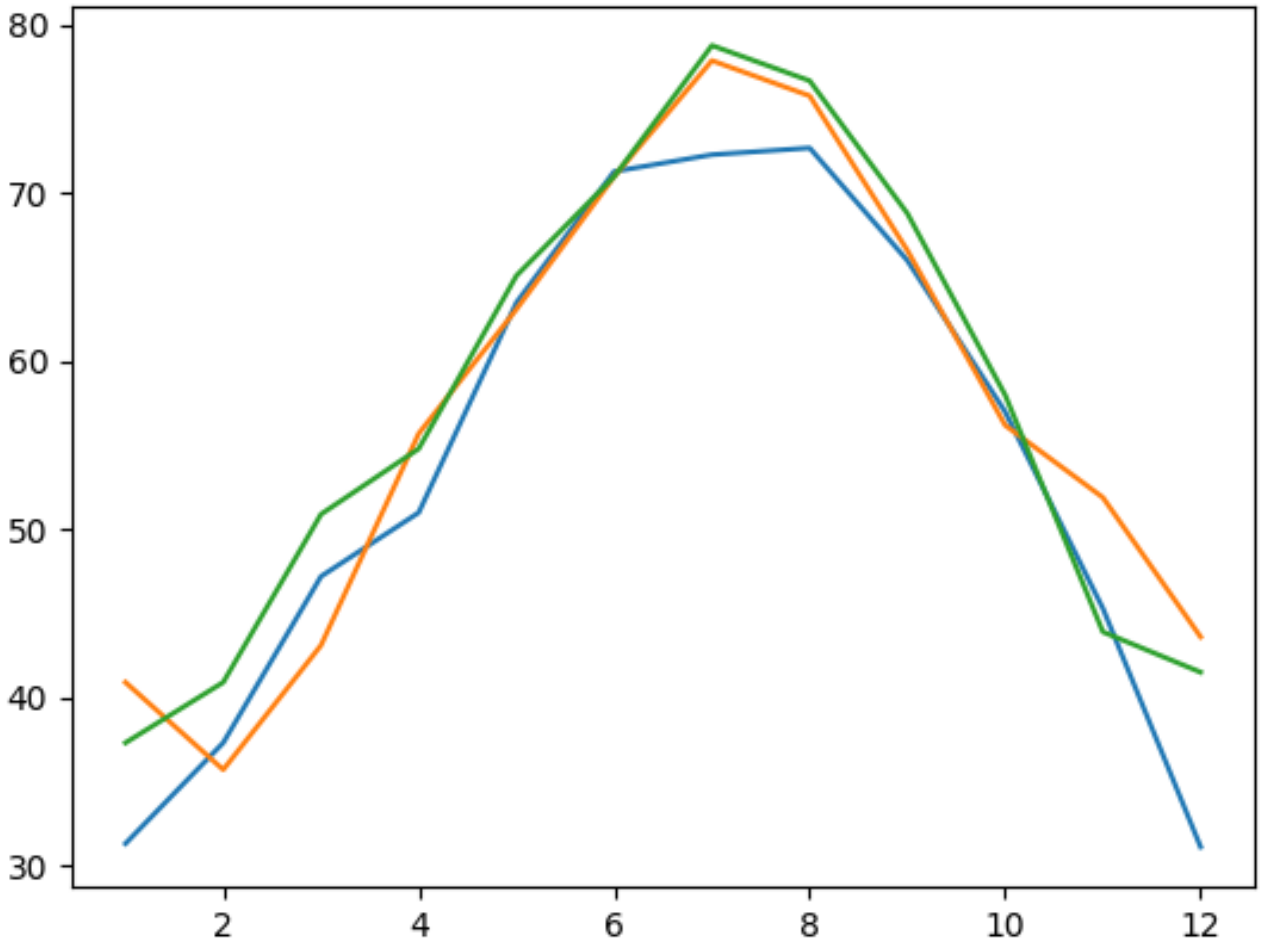
Plot the temperature in NYC aggregated by time

Place in python3 or in a python3 program file

```
from pylab import plot, show, savefig #note savefig
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]

plot(months, nyc_temp_2000, months, nyc_temp_2006,
months, nyc_temp_2012)
savefig('mygraph.png') #save in root directory
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!

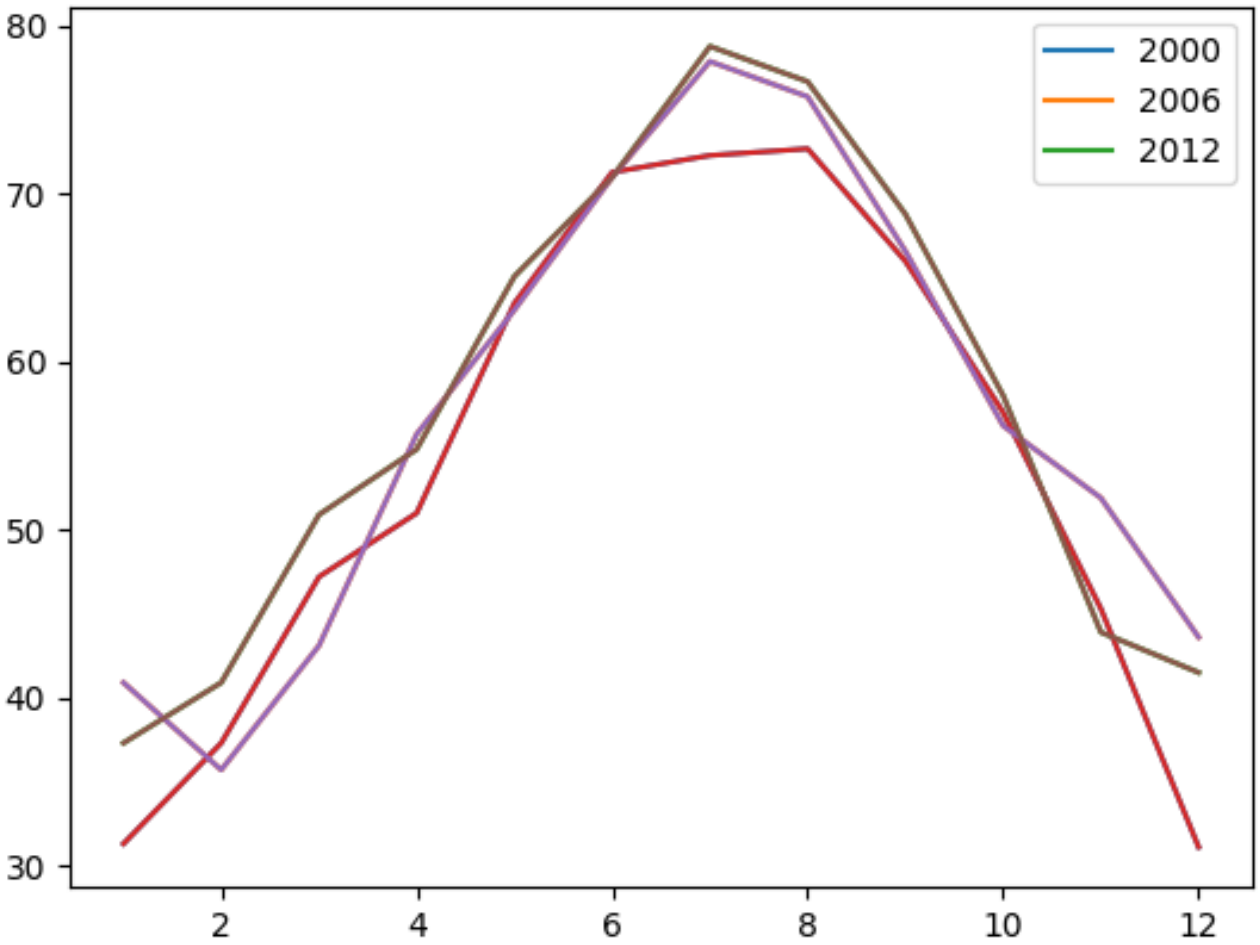
Plot the temperature in NYC aggregated by time

Place in python3 or in a python3 program file

```
from pylab import plot, show, savefig, legend
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]

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

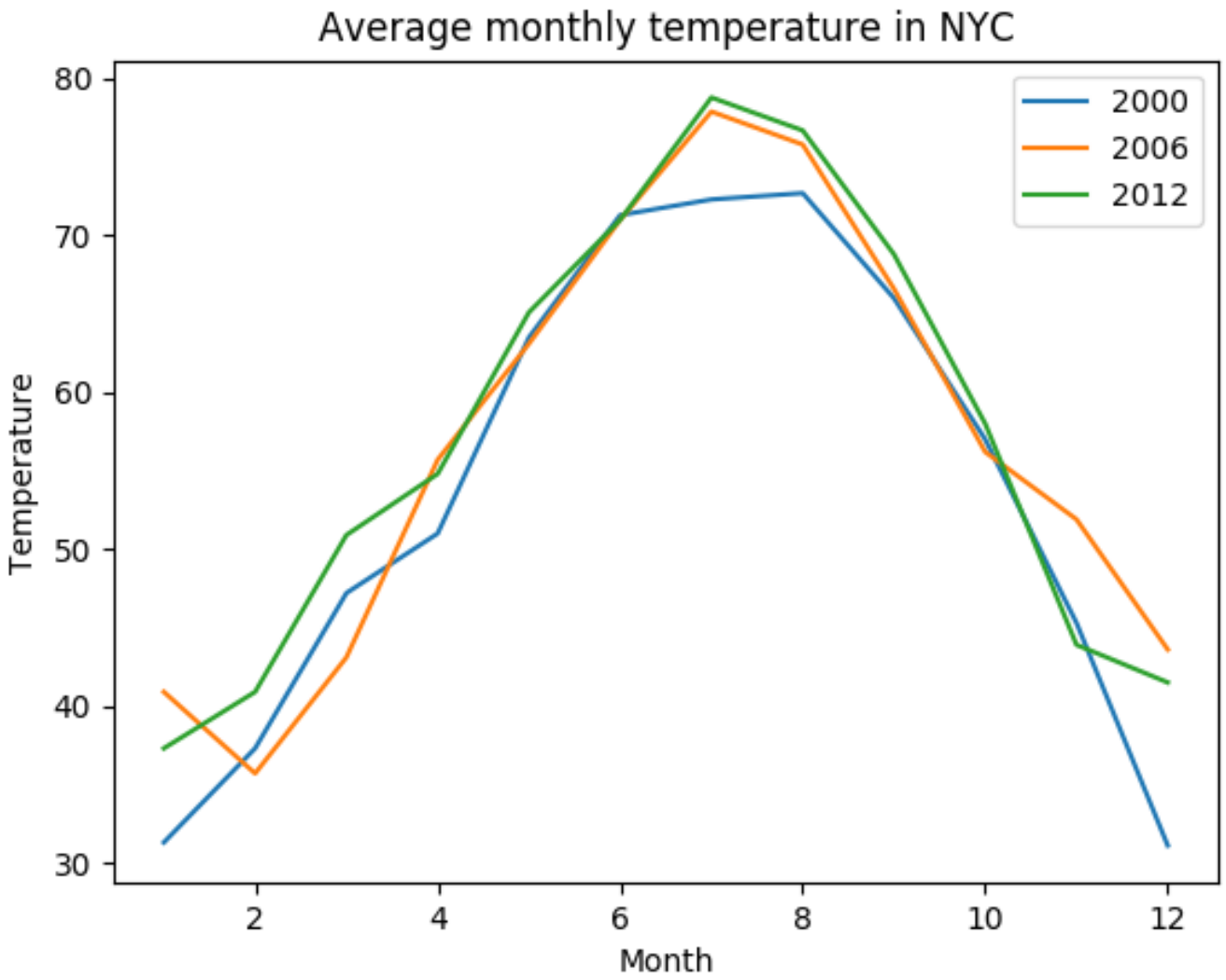
Place in python3 or in a python3 program file

```
from pylab import plot, show, title, savefig, xlabel, ylabel, legend, axis
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]

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

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

Sorry about the fine print. :-(

Add Title and Axes Descriptions!

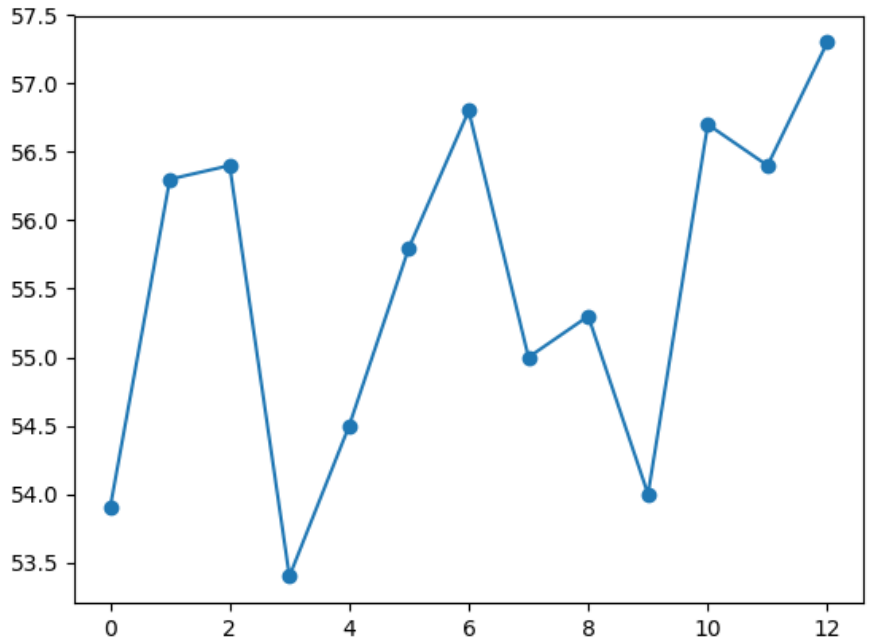
Plot the temperature in NYC aggregated by time



Changing the Field of View

Start with this plot, then we will change axis focus

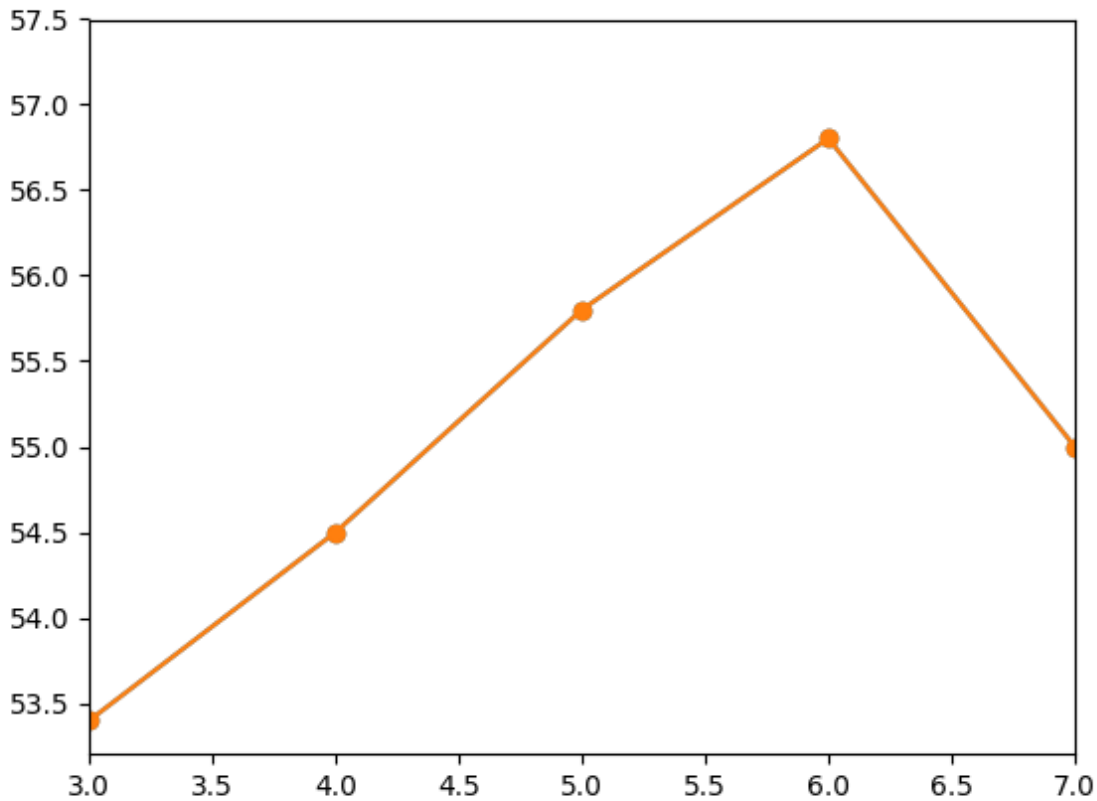

```
nyc_temp = [53.9, 56.3, 56.4, 53.4, 54.5, 55.8,
56.8, 55.0, 55.3, 54.0, 56.7, 56.4, 57.3]
plot(nyc_temp, marker='o')
axis()
#(-0.60, 12.6, 53.205, 57.495)
show()
```



COOL!!! The Field of View Has Been Changed!

Set the x-axis, min and max

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



Plotting the Log Equation

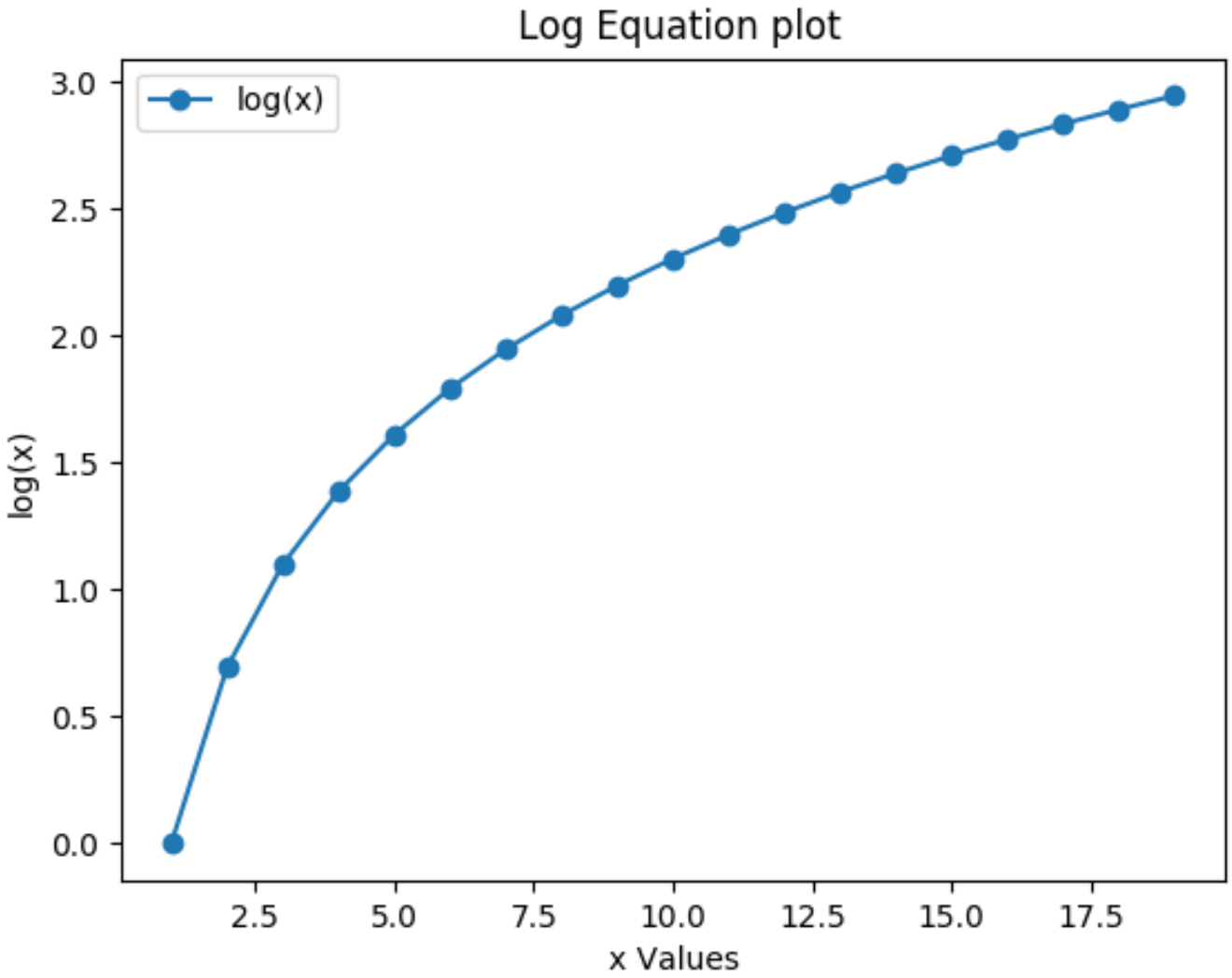
Log Plot

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

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

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

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

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

Sorry about the fine print. :-(

The Plotted Log(x)

Plot the temperature in NYC aggregated by time



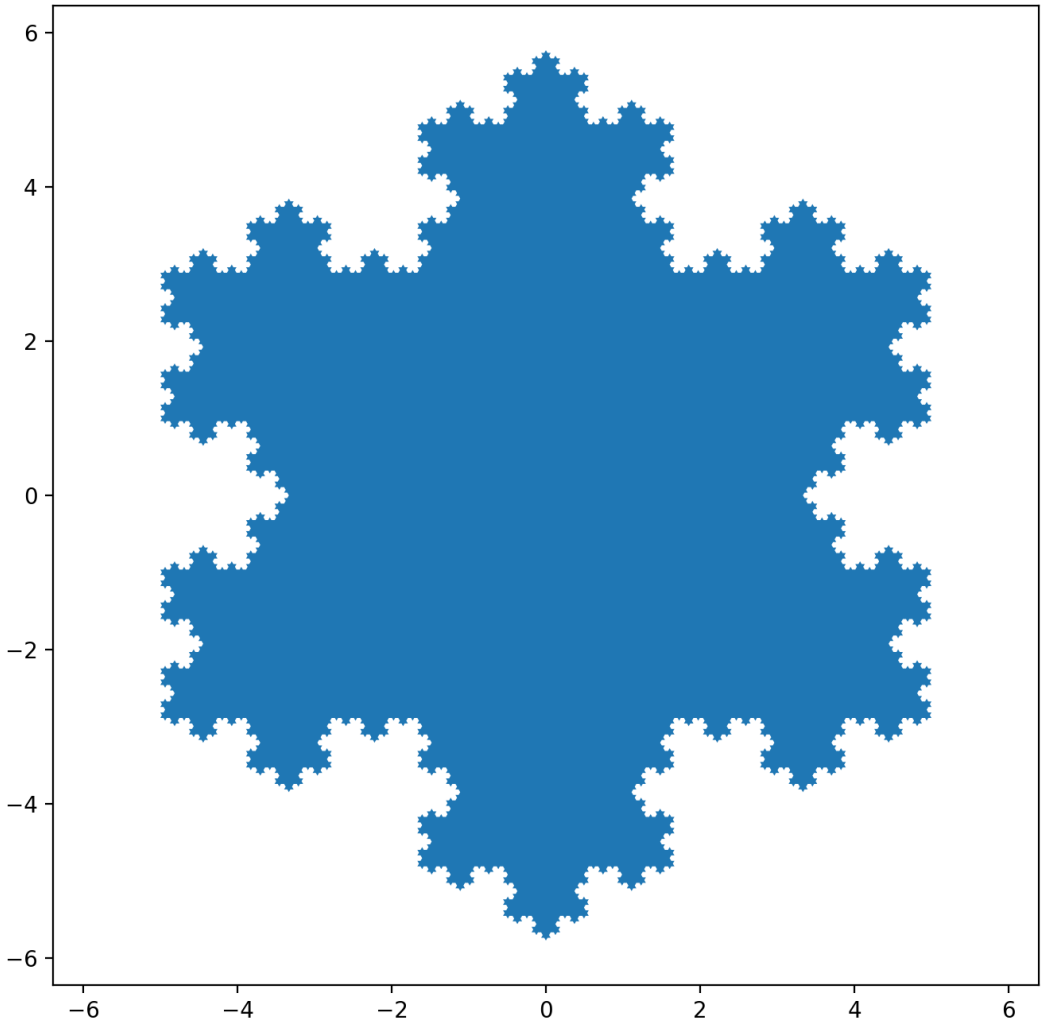
Koch Snowflakes

Source file: kochSnowflake.py


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

def koch_snowflake(order, scale=10):
    """  ref: https://matplotlib.org/3.1.1/gallery/lines_bars_and_markers
         /fill.html#sphx-glr-gallery-lines-bars-and-markers-fill-py"""

    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
    points = _koch_snowflake_complex(order)
    x, y = points.real, points.imag
    return x, y

x, y = koch_snowflake(order = 5) # thhe order is recursion dept
plt.figure(figsize=(8, 8))
plt.axis('equal')
plt.fill(x, y)
plt.show()
```

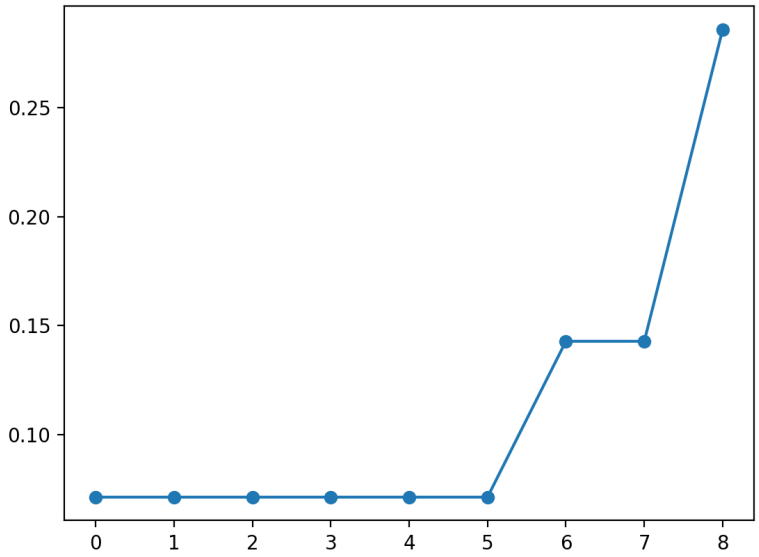
Other Types of Plots: The Koch Snowflake

Source file: kochSnowflake.py



Let's Code

Application: A Frequency Calculator



The string is, *hello there*

Gallery Website

`https://matplotlib.org/3.1.1/gallery/index.html`