Plotting Data with Python and The Jupyter Notebook.

- *Paper 5*
- *Eng 290*
- *By: Bryce DeAlessio*

Table of Contents

Introduction

There are plenty of software tools available for graphical plotting of data. Microsoft Excel is likely the tool most are familiar with. Some other tools might include Minitab or Google Doc's Spreadsheets. However there are other tools available that are much more powerful than a spreadsheet program but require the knowledge of a programming language to use. Matlab is probably one the most well known programming environments for plotting data. While Matlab is a very useful tool it is also not free nor is it's source-code Open Source. This makes it hard to share code and the insights gained from executing it upon any given dataset. This Tutorial Aims to give the reader just enough information to get started using Python & the Jupyter Notebook as a powerful, free & open source alternative for plotting and processing data. The Example data set used in this tutorial was obtained as part of exercise 4 for a class called ENGR 290 "Engineering Technical Writing" taught at Central Connecticut State University by Pamela Ann-Reardon[1]

Overview

Python is a general purpose programming language who's main features are it's human readable syntax, it's "batteries included"[2] philosophy, quality documentation, and it's ability to inter-operate with other programming languages. Python also has a liberal open source license which makes it free to use and modify both personally and commercially.

The Jupyter Notebook is a web-browser based interactive programming environment that can mix many types of rich web based content along with code execution.[5] This paper is written entirely inside of a Jupyter Notebook document. A Jupyter Notebook Document contains visual input and output cells. Input cells can be set to contain code, plain text, or web based content like Markdown, HTML and even LaTex.

The following major topics will be covered:

1. Installation of Python and the Jupyter Notebook via a packaged distribution called "Anaconda"[4].
2. Starting the Jupyter Notebook Application.
3. Writing and Explanation of the code Necessary to plot the given data-set.

Step1: Installing Python & Jupyter Notebook

As previously discussed Python does not come with the third party packages that make up what is commonly referred to as the "Scientific Stack"[1] which includes the Jupyter Notebook, they must be installed separately. Therefore Installing Python & the Jupyter Notebook is the process of downloading and installing Python from https://www.python.org (https://www.python.org) and then using the Python Package Manager called "PIP" to download and install the rest of the necessary packages. However This can be difficult especially on windows because it requires what's called a compiler for many of the scientific packages that leverage code from faster compiled languages like C & C++. Thankfully there are a number of Python distributions which bundle Python along with pre-compiled versions of all the fancy tools that make working with Python so awesome. This Bundling makes it easy to get started. The Number one distribution for computing is called "Anaconda" and it contains all of the most popular packages for scientific computing. "Anaconda" is produced by a company called "Continuum Analytics".

Equipment: A modern web browser such as Google Chrome, Firefox or Internet Explorer 11 plus, is required. This tutorial focuses on the Microsoft Windows version of Python therefore a Windows computer is recommended.

1. In an internet Browser go to https://www.continuum.io/downloads (https://www.continuum.io/downloads).
2. For this tutorial we will be using the Microsoft Windows version. So look for the Windows symbol like the one shown in figure 1 on Downloads Page and click on it. This will take you to the windows versions.
3. There are two major versions of Anaconda, One version is for the Python 3.x (3.6 as of writing) version, and the other is for Python 2.7 which is considered legacy(recieves bug fixes and security updates only) therefore this tutorial will focus on python 3.
4. Under each version there is a choice for the 64bit or 32bit version. Select the one that fit's the computer's processor architecture. As most modern (built within the last 5 years) computers & laptops have 64bit processors it's a safe bet to choose the 64 bit version.
5. Click the download button and follow the on-screen instructions to install Anaconda.

6. During the beginning of the installation a dialog will be displayed with a choice to install for all-users or for a single-user. Make sure to select the single-user option.

Figure 1

Windows symbol from Continuum Analytics website,[3]

Result: When the installation is complete the user should see a new folder in their Windows Start-Menu called "Anaconda3". Within this menu notice the Jupyter icon.

Step2: Starting a Jupyter Notebook

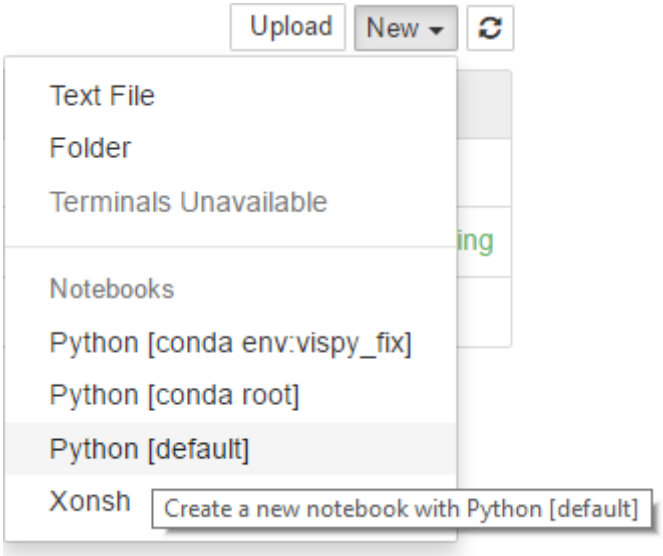
The Jupyter Notebook is web-based Interactive code execution environment. In Python, code is written to text files that are passed to the python interpreter program to run all at once. With the Jupyter Notebook, code can be executed in separate blocks of code called "code-cells" any output from the code executed in a code-cell will be shown in an output-cell directly below the code-cell. These output cells will also display the results of any graphical plots.

Caution: Running the Notebook from the start menu will open the notebook file browser in your computer's current user-profile directory. The file browser cannot browse above this directory for security purposes.

Equipment: A modern web browser such as Google Chrome, Firefox or Internet Explorer 11 plus, is required.

1. Start the Jupyter Notebook by going to the Windows Start menu and selecting "Anaconda3". Then select the icon for the "Jupyter Notebook". In a few seconds the computer's default browser should start with the Jupyter file browser tab active.
2. In the top right of the file tab select the dropdown button that says "New"
3. Select the Python[root] or Python[default] option within the dropdown box. See Figure 2.

Figure 2



Result: A new browser tab should open with the notebook active. With the Jupyter Notebook Environment setup a brief overview of some basics of the Python Language will be reviewed.

Step3: Writing the code

Writing the code is the actual process of writing the various statements and expressions that the computer can translate into special instructions that when executed makes the computer do something. Within the Jupyter Notebook the code in a cell can be executed independently from other code cells at will by clicking the play button in the Notebook toolbar or pressing `ctrl+Enter`

on the keyboard when a cell is selected.

What follows is a brief primer on the basic Python variables, data types, and data structures used in this tutorial and a step by step explanation of the code necessary to plot our snowfall totals. A typical python program begins with the imports, then actual code and if it is a script to run then a "main" statement.

Variables: A variable in Python is a named reference to something else like a number or a string of characters. To assign something to a variable in Python the equal symbol is used, with the variable on the left and value on the right. A variable name must not start with a number and cannot contain spaces. see Figure 3 for an example.

Figure 3

```
a = "Hello World"
b = 10
c = 9.98
```

Data Types: In the Example above, the string "Hello World" has been assigned to variable "a", "b" is assigned an integer and "c" is assigned a float. A string is entered as a sequence of characters wrapped with quotation marks. An integer is entered as a number with no decimal point and a float(short for floating point decimal) is entered as a number with a decimal point. Similar to the way the data type of information in a Microsoft Excel table needs to be set in order for calculations to be performed properly, Python requires some attention to the data type when entering information.

Data Structures: A data structure is simply a way of storing many pieces of information together in an easy to manage way. One of the primary data structures in python is the List. A list is a sequential data structure that can be used to store any type of data. A list can be created by typing out a comma separated sequence of values bounded by an opening and closing square bracket. In Figure 4 below a list of integers and strings is created and assigned to the variable "my_list".

Figure 4

```
In [1]: my_list = [8, "six", 7, "five", 3, "Oh", 9]
```

Individual items in a list can be accessed via an integer based index for the desired item. For example to retrieve the string `"six"` from `my_list` you would access it using the integer 1. The syntax for accessing an item from a list is to type a opening bracket directly adjacent to the list variable, an integer for the index location of the item and then a closing bracket.

Figure 5

```
In [2]: my_list[1]
Out[2]: 'six'
```

Note: The integer 1 is used because indexing in python and almost every other computer programing language (except matlab) uses zero based indexing.

Step 3.a: Writing the Code - imports

The import statement in Python is what allows you to use the code in other python modules(files) within your own program. Typically one would write functions, which are re-usable blocks of code that are given a name and then use those functions in many places in other modules. Any function or object from an imported module can be accessed with a period "`.`" directly adjacent to the module name. For this tutorial there will not be much discussion on what an object is except to say that everything in Python is actually an object and an object is a piece of data with functions attached to it that work on the data within that object. These functions are accessed with the "`.`" as well.

What follows is a step by step explanation of the code necessary to plot our snowfall totals. A typical python program begins with the imports, then actual code and if it is a script to run then a "main" statement.

1. Line #1 imports the pandas package which provides us with something called a "DataFrame" which is a table-like data structure similar to a Microsoft Excel sheet.
2. On line #2 we import the plotting package called matplotlib. Matplotlib has many sub-packages and we access the plotting commands sub-package called "pyplot" with the '.' (dot operator). Adding the "as" statement with a new variable creates an easier to type shorcut to the matplotlib.pyplot module.
3. Line #3 is not python specific code but a shortcut command for the notebook which tells it to display graphical plots in the notebook page beneath the code-cell rather than a separate window.
4. With the code cell still selected, execute the cell by clicking the play button in the jupyter menu-bar or typing ctrl+Enter.

Figure 6

In [3]:
```
import pandas
import matplotlib.pyplot as plt
%matplotlib inline
```

All the extra code from these modules is now available for use and the necessary data for plotting can be input.

Step3.b: Writing the Code - Data Input

Data input in the case of this tutorial comprises using the built-in "range" function to generate a monotonically increasing series of numbers and manual input into a python list. There are many other ways to input data for processing such as reading csv files or even downloading weather data directly from a government web-site. For this Tutorial it will be kept simple.

1. Line #1 creates a list of the years from our Snowfall data-set and assigns it to a variable called "years". While the range of years can be manually written out python's "range" function is used instead. The "range" function can take a single integer(whole number); two integers, a starting number and an ending number; or optionally, a third number defining a step between each number generated. The numbers generated always end up one less than the number given. This is common for most programming languages as computers always count from zero. Note: the range function does not actually produce a list of values right away it creates an iterator that yields one value at a time. Using the "list" constructor function immediately pulls each value from an iterator like range and stores it in the list.
2. Line #2 a feature of a python list called sort is used. "sort" is what is called a method. A method is a function or action that is attached to the list and can be used on the data inside. The list is sorted to make sure the Years are in descending order just like the example data-set.
3. In Line #3 The snow fall totals must be input directly to create the list due to their random nature. This list is constructed by writing out a comma separated sequence of numbers bounded with an opening and closing square bracket. The list is assigned to the variable `snow_totals`
4. Again Execute the code as previously performed.

Now that the data is in two separate lists they can be glued together into Something called a DataFrame. A DataFrame comes from the Pandas Library that was imported in the previous set of steps. It is a two dimensional Tabular data structure similar to a Microsoft Excel Sheet.

Figure 7 [1]

```
In [4]: years = list(range(2000, 2011))
        years.sort(reverse=True)
        snow_totals = [60,34,25,27,9,48,44,24,16,33,19]
```

Here a pandas DataFrame is used to unite the two sepparate lists `years` and `snow_totals` into a single table.

1. Line #1 creates an empty DataFrame object and assigns it to the variable `snowfall_table`.
2. Line #2 creates a column in the Dataframe called `"Year"` and assigns the `years` list of values to this column. This uses the item setting syntax where a label is entered within opening and closing square brackets adjacent to the DataFrame variable followed by an equal sign and the list of data to assign to that column.
3. Line #3 does the same as Line #2 but with the `snow_totals` data.
4. Line #4 the variable is typed again so that the table is displayed in the output cell below as part of Figure 8.

Figure 8

In [5]:
```
snowfall_table = pandas.DataFrame()
snowfall_table['Year'] = years
snowfall_table['Snowfall Totals'] = snow_totals
snowfall_table
```

Out[5]:

	Year	Snowfall Totals
0	2010	60
1	2009	34
2	2008	25
3	2007	27
4	2006	9
5	2005	48
6	2004	44
7	2003	24
8	2002	16
9	2001	33
10	2000	19

Step 3.C Writing the Code - Plotting

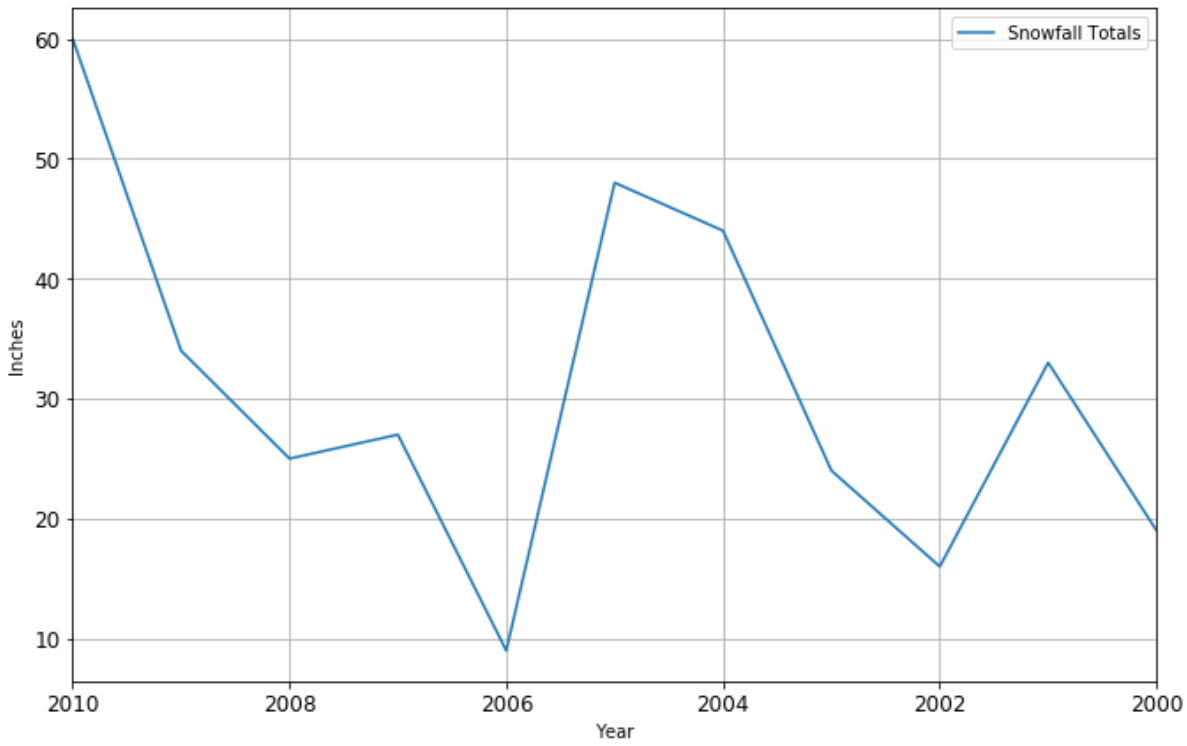
In Figure 7 the plotting functionality of the DataFrame is used to create the necessary line plot. Because of the way this notebook was configured during the import statements step the plot is displayed in an output cell directly beneath the code-cell within this document. The Dataframe utilizes the "matplotlib" plotting module that was imported in step 3.3

1. Line #1 the plot method of the `snow_plot` DataFrame is used. Within the parenthesis, which signal the plot function to be called parameters are given to designate what to plot and how to configure the plot. The first parameter `x` is given a string `"Year"` to designate the `"Year"` column of the DataFrame be used for the X-Axis and the `y` parameter sets the Y-axis. The `fontsize` parameter sets the font size on the plot axes, `figsize` sets the dimensions of the plot in inches and `grid=True` turns on the major grid lines.
2. Line #2 accesses the yaxis of the plot and sets it's label to "Inches".

Figure 9

In [6]:
```
snow_plot = snowfall_table.plot(x='Year', y='Snowfall Totals', fontsize=12, fi
gsize=(11,7), grid=True)
snow_plot.yaxis.set_label_text('Inches')
```

Out[6]: <matplotlib.text.Text at 0x8dfadd8>



In Order for our plot to be useful outside of the notebook context it can be saved as an image

1. line #1 the `get_figure` function retrieves the whole figure object that frames the plot object and assigns it to the `snow_fig` variable.
2. Line #2 the `savfig` method of the figure object is used to save the plot to an image file called "snowfall_totals.png". The default image format that is output is .png . The file should normally be found within the same directory of the currently running notebook.

figure 10

In [7]:
```
snow_fig = snow_plot.get_figure()
snow_fig.savefig('snowfall_totals.png')
```

Conclusion

Python coupled with the Jupyter Notebook Environment and other scientific packages make plotting and data processing a snap. This tutorial presented only a very small portion of the capabilities of the Python scientific ecosystem. For more information on the Anaconda Python Distribution go to https://www.continuum.io/ (https://www.continuum.io/)[4]. And for more information on project Jupyter go to http://jupyter.org/index.html[5]

Bibliography

[1] P. Reardon, "Excercise 4 - Visuals", ENGR 290 Engineering Technical Writing, Central Connecticut State University, 2017

[2] “Brief Tour of the Standard Library,” 10. Brief Tour of the Standard Library — Python 3.6.1 documentation, 26-Mar-2017. [Online]. Available: https://docs.python.org/3/tutorial/stdlib.html#batteries-included (https://docs.python.org/3/tutorial/stdlib.html#batteries-included).

[3] T. Oliphant, E. Jones and P. Peterson, "The SciPy Stack specification — SciPy.org", Scipy.org, 2017. [Online]. Available: https://www.scipy.org/stackspec.html (https://www.scipy.org/stackspec.html). [Accessed: 08- May- 2017].

[4] “Download Anaconda Now!,” Continuum. [Online]. Available: https://www.continuum.io/downloads (https://www.continuum.io/downloads).

[5] “Project Jupyter,” Project Jupyter, 13-Apr-2017. [Online]. Available: http://jupyter.org/index.html (http://jupyter.org/index.html).

In [17]:
```
%%html
<style>
.text_cell_render > p,ol,ul {font-size: 12px}
.text_cell_render > p,ol,ul {line-height: 18px}
.rendered_html h1 {font-size: 20px}
.rendered_html h2 {font-size: 18px}
.rendered_html h3 {font-size: 16px}
.rendered_html h4 {font-size: 14px}
#paper_5_title {font-size: 24px;
                font-weight: bold}
</style>
```