Table of contents

Certification

> I certify that this project is my own work and is not the work of others. I agree not to share my solution with others. - Danielle Warden.

Getting Started

Welcome to another adventure in Jupyter Notebooks!

Explanation

What are data frames used for? A data frame is one of the most commonly used pandas objects. You can think of it as approximating a spreadsheet or a SQL table.

While you could use Excel for all phases of your project, Jupyter gives you a bit more lifting power. There are other reasons to use Jupyter, such as an improved ability to test and debug, as well as scalability and some other performance issues. We could go down a rabbit hole in a serious discussion regarding the merits of Excel and the merits of Jupyter, but let's not. Let us code!

Import libraries

The magic won't work without your libraries, so let's get those added.

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
```

Crafting the Visual

Once you get your libraries imported, you then need to get your data imported. First you must get the file path. In the following example, I've hard coded the file path. This works well when you are testing on your own system. It's preferable when you release your notebook to the wilds, to use a relative file path. Why? Your user's file path will be different. Using a relative file path will account for this.

Unfortunately, relative pathing seems to have some issues at present.

https://365datascience.com/question/importing-an-excel-document-onto-python/

I've also included some instructions that I'd like to use 'Date' as the index column. I've left the relative path to be illustrative of the difference. Update the hardcoded path when you test this on your system.

Hard coded path:

```
In [2]: TestData = pd.read_csv(r"C:\Users\danie\Desktop\Intermediate Python\Asg10\WardenAsg10\Stock01.c
        sv", index_col='Date')
```

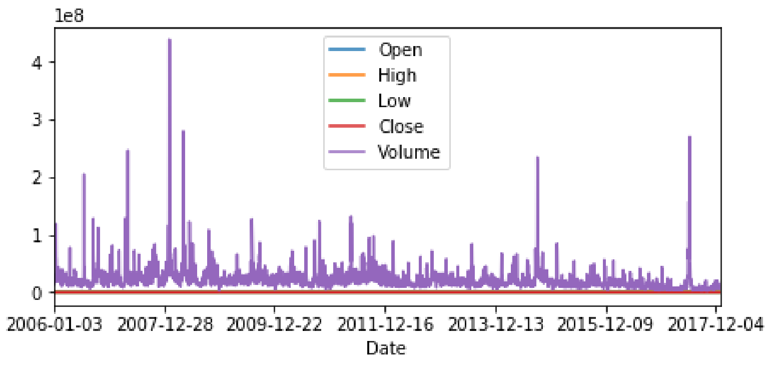
Relative path:

```
In [ ]: TestData = pd.read_csv("../Stock01.csv", index_col='Date')
```

Once the data is loaded, you need to massage the data a bit for readability. Why? Run the next cell to see why...

```
In [3]: TestData.plot(figsize = (7,3))

Out[3]: <matplotlib.axes._subplots.AxesSubplot at 0x2364bb19348>
```



This is a mess! The data is valid, but not accessible for your user, it's data overwhelm.

In the following code, the data is restricted to only the columns High and Low, and a date range is specified.

```
In [4]: TestData = TestData.loc[:,'High':'Low']
        TestData01 = TestData.loc['2006-01-06':'2008-05-28']
```

In [5]:
```
TestData01.head(3)
```

Out[5]:

	High	Low
Date		
2006-01-06	43.57	42.80
2006-01-09	43.66	42.82
2006-01-10	43.34	42.34

Let's increase the values.

In [6]:
```
TestData02 = TestData01*2
TestData02.head(3)
```

Out[6]:

	High	Low
Date		
2006-01-06	87.14	85.60
2006-01-09	87.32	85.64
2006-01-10	86.68	84.68

Let's spread the values out a bit.

In [7]:
```
TestData03 = TestData02.loc[:]
TestData03['High']+=10
TestData03.head(3)
```

Out[7]:

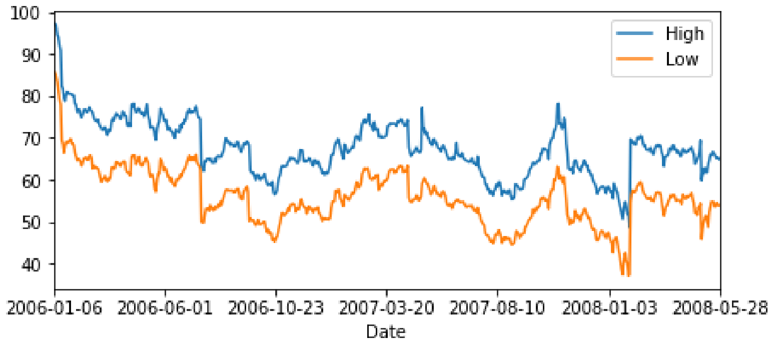
	High	Low
Date		
2006-01-06	97.14	85.60
2006-01-09	97.32	85.64
2006-01-10	96.68	84.68

Results

Here's the result of your coding efforts!:

```
In [8]: TestData03.plot(figsize = (7,3))
        plt.yticks(np.arange(40, 105, step=10))
```

```
Out[8]: ([<matplotlib.axis.YTick at 0x2364bd7c3c8>,
          <matplotlib.axis.YTick at 0x2364bd61a48>,
          <matplotlib.axis.YTick at 0x2364bd02d08>,
          <matplotlib.axis.YTick at 0x2364bdbfb88>,
          <matplotlib.axis.YTick at 0x2364bdbfe08>,
          <matplotlib.axis.YTick at 0x2364bdbba48>,
          <matplotlib.axis.YTick at 0x2364bdb84c8>],
         <a list of 7 Text yticklabel objects>)
```



##The End##