**Data Focused Python, 2018 K1**

**Homework 4**

***Due At 11:59 pm Monday, Sept. 24***

1. **(100 points) Pandas Series and DataFrame**
2. The **b\_soup\_1.py** file contains the code from the Week 2 lecture notes, showing how to start with the HTML for a web site and process that HTML into a **list** of table data value strings (**str**) using the **BeautifulSoup** module.

First, modify **b\_soup\_1.py** so that the program’s only output is the final sequence of table cell value **list**s: no **yc\_temp.txt** file, no intermediate results being displayed, etc.

Then, modify the code at the end of the program so that the table cell values are accumulated into a **list** of **list**s, representing the table of rows, something like this:

**daily\_yield\_curves = [**

**[ …** *header list* **… ],**

**[ …** *first data list* **… ],**

**…**

**[ …** *final data list* **… ]**

**]**

The first “inner” **list** should represent the header row:

**['Date', '1 mo', '3 mo', '6 mo', '1 yr', '2 yr', '3 yr',**

**'5 yr', '7 yr', '10 yr', '20 yr', '30 yr']**

Following that should be a **list** for each data row. Be sure to convert each interest rate value from a string to a **float**:

**['01/02/18', 1.29, 1.44, 1.61, 1.83, 1.92, 2.01,**

**2.25, 2.38, 2.46, 2.64, 2.81]**

**...**

**['09/14/18', 2.02, 2.16, 2.33, 2.56, 2.78, 2.85,**

**2.90, 2.96, 2.99, 3.07, 3.13]**

Create a file named **daily\_yield\_curves.txt** containing a neatly formatted table of this information.

1. Investigate **matplotlib**’s 3D Surface Plot and Wireframe Plot (**https://matplotlib.org/Matplotlib.pdf**). Produce a 3D Surface Plot of the daily yield curves, with days since 01/02/18 on the X axis, months to maturity on the Y axis (from 1 month to 360 months), and rate on the Z axis. Orient the plot in such a way that this yield curve evolution surface is reasonable to look at. Set axis labels like **‘trading days since 01/02/18’**, **‘months to maturity’**, and **‘rate’** so that the user can tell which axis represents which dimension in the plot. After you have produced a Surface Plot, produce a Wireframe Plot of the same information. (You do *not* need to save screenshots of your plots.)

The Y axis should show *months to maturity*. You will have to “convert” the column labels into the appropriate integer number of months. You can be unclever about this and use a **list** like **[1, 3, 6, 12, 24, 36, 60, 84, 120, 240, 360]**, or you can be more clever and set up a **dict** mapping from column name to number of months, like **cn\_to\_nm = { ‘1 mo’ : 1, ‘3 mo’ : 3, …, ’30 yr’ : 360 }**. It is okay to be unclever.

***Hint:*** You will need to create an **ndarray** of the interest rate values from the **daily\_yield\_curves** list of lists in order to produce plots.

**matplotlib** facilities for creating 3D Surface Plots and Wireframe Plots make use of **numpy** **ndarrays**. Recall that you can convert a **list** of **list**s to a 2-dimensional **ndarray** using **np.array()**. As an example, try:

**X = np.array([ [ 0, .25, .5, .75, 1 ],**

**[ 0, .25, .5, .75, 1 ],**

**[ 0, .25, .5, .75, 1 ] ])**

**Y = np.array([ [ 0, 0, 0, 0, 0 ],**

**[ .5, .5, .5, .5, .5 ],**

**[ 1, 1, 1, 1, 1 ] ])**

**Z = np.array([ [ .4, .2, .1, .1, .2 ],**

**[ .3, .5, .2, .3, .4 ],**

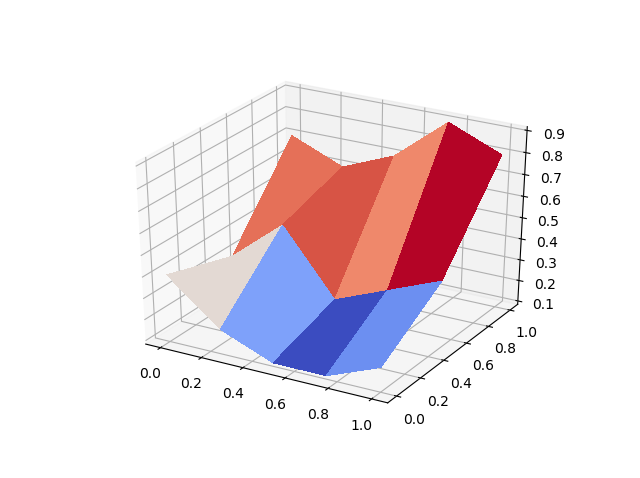
**[ .7, .6, .7, .9, .8 ] ])**

As the last step in creating a plot, you must use the statement

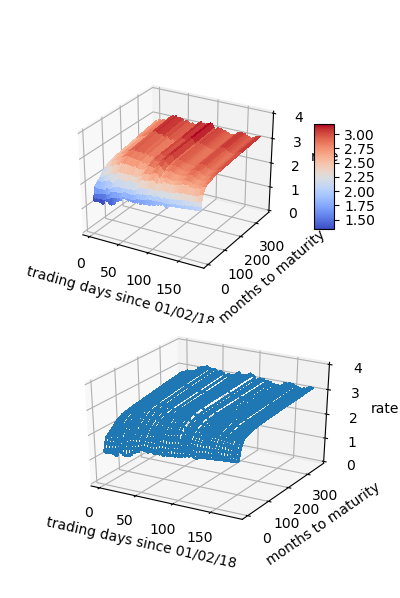
**plt.show()**

to make the plot be drawn on your screen. After the plot has been drawn, click the close button, **X**, in the upper right corner so that your program can continue.

A surface plot of these test **ndarrays**, **X**, **Y**, and **Z**, should look similar to the screen shot on the next page:



Surface and Wireframe Plots of the yield curve data should look similar to this:



1. Our interest rate table is a natural Pandas **DataFrame**, with trading dates as rows and bond maturities as columns. From the **daily\_yield\_curves** **list** of **list**s, create a **DataFrame** named **yield\_curve\_df** with the date strings as the row labels (**‘01/02/2018’**, …, **‘07/20/2018’**), the bond maturities as the column labels (**‘1 mo’**, …, **’30 yr’**), and the corresponding interest rate values as the row/column item values. Use appropriate slices/loops/comprehensions involving **daily\_yield\_curves** to create **yield\_curve\_df**.

**DataFrame** has a **plot()** member function that uses **matplotlib**. You can use **yield\_curve\_df.plot()** to create a plot with rows on the horizontal axis, values on the vertical axis, and with each column represented as a different line. You will still need to use

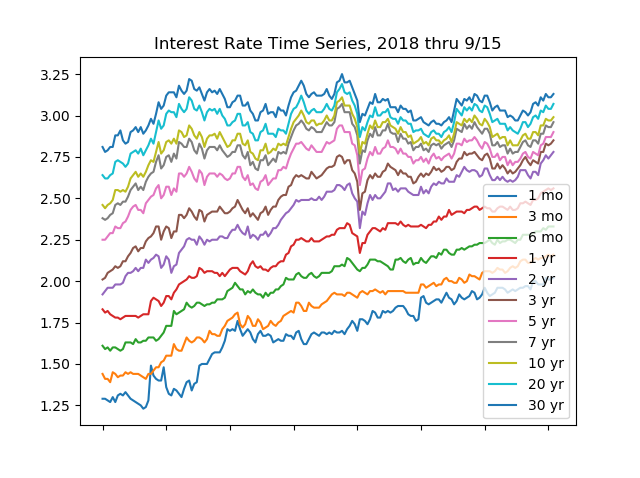
**plt.show()**

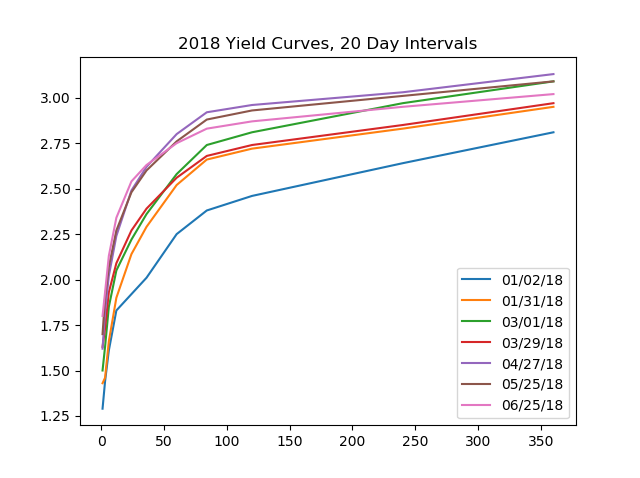
to make the plot be drawn on your screen. Since the rows are trading days, this plot will be of the *time series* of interest rates for each maturity: 1 month, 3 months, …, 30 years. You will see that so far during 2018, short-term interest rates are rising relatively rapidly, whereas long-term rates have fluctuated but not risen very much. This so-called *flattening of the yield curve* may signal a slowing economy.

If we *transpose* **yield\_curve\_df**, so that trading dates become the columns and maturities become the rows, then a **plot()** will show us the daily yield curve for every trading day so far this year. This will be an unreadable mess with over 100 lines.

From **yield\_curve\_df** create a **DataFrame** object named **by\_day\_yield\_curve\_df**, containing the transpose of **yield\_curve\_df** *but* only including a column for every 20th trading day, that is, day 0, day 20, day 40, …, day 120. The column labels should be **‘01/02/18’**, **‘01/31/18’**, …, **‘06/25/28’** if you do this correctly. You will need to modify the row labels from **‘1 mo’**, **‘3 mo’**, and so forth, to the corresponding integer number of months—1, 3, …, 360—in order for the plot’s horizontal axis to make sense.

The by-maturity time series plot and the by-trading-day yield curve plots should look about like the examples shown on the next page:





***REMEMBER*** to put your **b\_soup\_1.py** code file into a **.zip** archive and upload to the course web site.