**Data Focused Python**

**Homework 4**

**Mini 3 2020**

***Due at 9:00 pm on Tuesday, Feb. 11***

***You will lose 1 point every 5 minutes after that time***

1. **(100 points) Pandas Series and DataFrame and Matplotlib**
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.

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', '2 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**:

**['01/02/19', 2.40, 2.40, 2.42, 2.51, 2.60, 2.50,**

**2.47, 2.49, 2.56, 2.66, 2.83, 2.97]**

**...**

**['09/13/19', 1.99, 1.98, 1.96, 1.92, 1.88, 1.79,**

**1.76, 1.75, 1.83, 1.90, 2.17, 2.37]**

**...**

Create a file named **daily\_yield\_curves.txt** containing a neatly formatted table of this information for the year 2019.

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 01/02/19 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 01/02/19’**, **‘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, 2, 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, ‘2 mo’ : 2, …, ’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 ],**

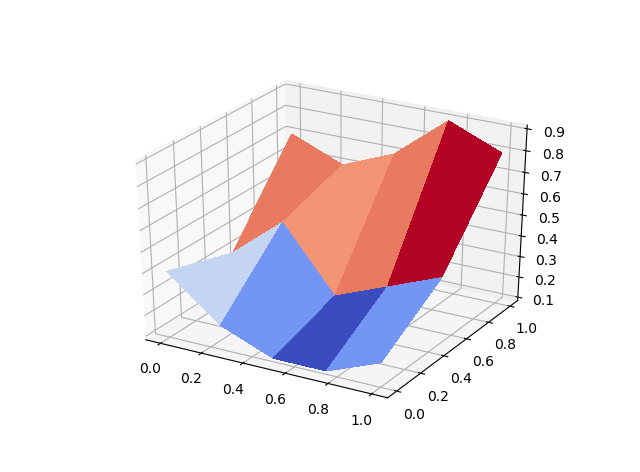
**[ .7, .6, .7, .9, .8 ] ])**

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

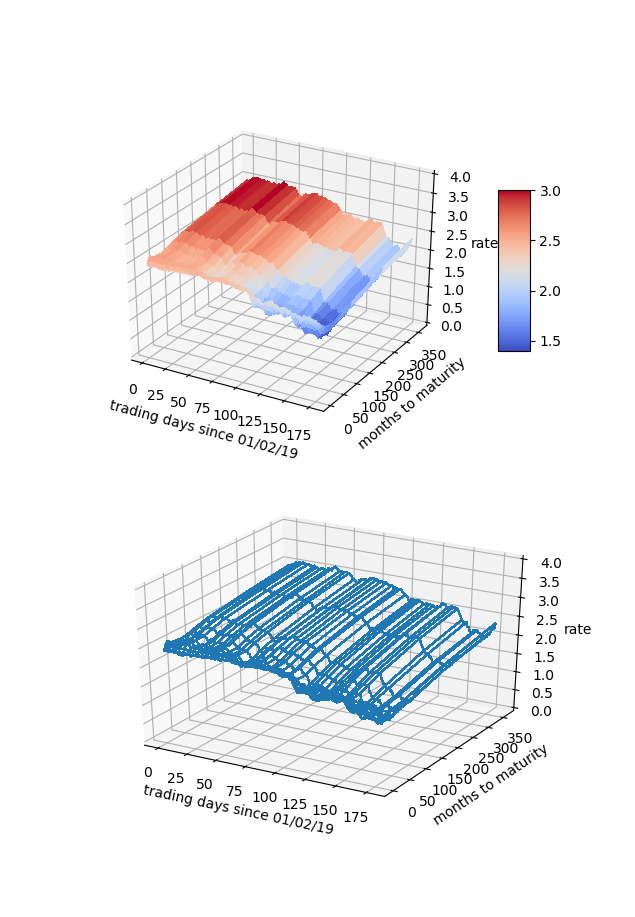
**plt.show()**

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 very similar to the screen shot on the next page:



Surface and Wireframe Plots of the yield curve data should look similar to these (but with a few more months of data: these plots only go through September 13, 2019):



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 (**‘01/02/2019’**, …, **‘09/13/2019’**, …), 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, 2 months, 3 months, …, 30 years. You will see that during 2019, interest rates for all maturities have fallen. Generally, it is considered more risky to lend for longer periods of time, so a “normal” yield curve slopes up: interest rates are lowest at 1 month, higher at 1 year, higher still at 10 years, and highest at 30 years. This is about what we see on January 8, 2019:

**Date 1 mo 2 mo 3 mo 6 mo 1 yr 2 yr 3 yr 5 yr 7 yr 10 yr 20 yr 30 yr**

**01/08/19 2.40 2.42 2.46 2.54 2.60 2.58 2.57 2.58 2.63 2.73 2.88 3.00**

But in late August, 2019, the yield curve *inverted*, and long-term interest rates were actually lower than short term rates:

**Date 1 mo 2 mo 3 mo 6 mo 1 yr 2 yr 3 yr 5 yr 7 yr 10 yr 20 yr 30 yr**

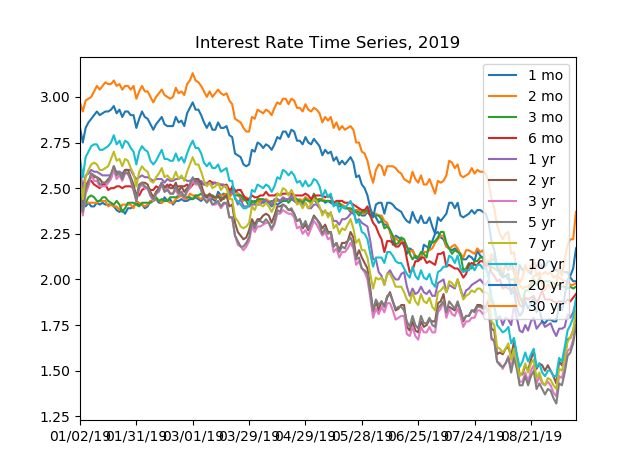
**08/29/19 2.10 2.03 1.99 1.89 1.75 1.53 1.44 1.40 1.46 1.50 1.78 1.97**

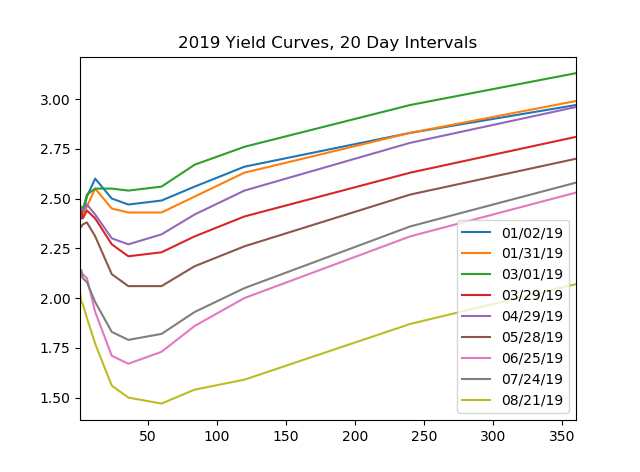
This may—or may not!—be an indicator of impending recession.

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 240. The column labels should be **‘01/02/19’**, **‘01/31/19’**, …, **‘06/25/19’**, … if you do this correctly. You will need to modify the row labels from **‘1 mo’**, **‘2 mo’**, and so forth, to the corresponding integer number of months—1, 2, …, 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 similar to the examples shown on the next page (but with more data):





***When finished, put your b\_soup\_1.py source code file into a zip archive named* Team***N***\_HW4.zip *file, where*** *N* ***is your team number, then upload your .zip archive to Canvas.***