Assignment 3

Goals

The goal of this assignment is to work with lists and dictionaries in Python.

Instructions

You will be doing your work in a Jupyter notebook for this assignment. You may choose to work on this assignment on a hosted environment (e.g. [tiger](https://tiger.cs.niu.edu/jupyter/)) or on your own local installation of Jupyter and Python. You should use Python 3.8 or higher for your work. To use tiger, use the credentials you received. If you work remotely, make sure to download the .ipynb file to turn in. If you choose to work locally, [Anaconda](https://www.anaconda.com/download) is the easiest way to install and manage Python. If you work locally, you may launch Jupyter Lab either from the Navigator application or via the command-line as jupyter-lab.

In this assignment, we will be working with data about US Senators’ stock trading practices. In this case, we will be using the [Senate Stock Watcher](https://senatestockwatcher.com/) which was built by Timothy Carambat. Rather than using this dataset directly, I have created a subset of this data, which can be read as a list of dictionaries. That data is located [here](http://faculty.cs.niu.edu/~dakoop/cs503-2021fa/a3/senate-stock-trades.json), and I have created a template notebook, [a3.ipynb](http://faculty.cs.niu.edu/~dakoop/cs503-2021fa/a3/a3.ipynb), that contains a cell that will download and read that data. You can right-click and save-as the a3.ipynb file, and, if working on tiger, upload that file. Once loaded, the data is a list of dictionaries where each dictionary has ten key-value pairs. Those keys and a brief description are:

* transaction\_date: the date of the transaction as a **string** in mm/dd/yyyy format
* owner: the owner of the stock (the senator or a family member)
* ticker: the stock ticker symbol (e.g. AAPL)
* asset\_type: whether the asset is a stock, bond, cryptocurrency, etc.
* type: the type of transaction (purchase or sale)
* amount\_range: the amount of the transaction (a range specified by a tuple (min\_amount, max\_amount)). max\_amount can be None to indicate no upper bound
* senator: the name of the senator involved in the transaction

You will be answering queries and writing functions to help analyze this data. You may not use external libraries including statistics, collections, datetime, or pandas for this assignment.

Due Date

The assignment is due at 11:59pm on **Thursday, September 23**.

Submission

You should submit the completed notebook file required for this assignment on [Blackboard](https://webcourses.niu.edu/). The filename of the notebook should be a3.ipynb.

Details

Please make sure to follow instructions to receive full credit. Use a markdown cell to **Label** each part of the assignment with the number of the section you are completing. You may put the code for each part into one or more cells.

0. Name & Z-ID (5 pts)

The first cell of your notebook should be a markdown cell with a line for your name and a line for your Z-ID. If you wish to add other information (the assignment name, a description of the assignment), you may do so after these two lines.

1. Owner Types (10 pts)

Find all of the possible values for owner types. List each type only **once**!

Hints

* Iterate through all of the list elements, and extract the owner type from each element
* Use a set

2. Largest Transaction (10 pts)

Write code to find the trade in the dataset that involved the most money (amount). Output the **name** of the senator who was involved in that trade. Remember that you will need to iterate through each element of the list, and each element is a dictionary which has various keys including amount\_range and name.

3. Transaction Counts (10 pts)

Write code to create a dictionary that keeps track of how many **sales** (full or partial) transactions each senator has been involved in.

Hints

* Make sure to filter to only capture sales.

4. Sales Sums (15 pts)

Write code to create a dictionary that keeps track of the sum of the **sales** that each senator has made. Since we only have ranges, your output should also be a **range**. For example, if a senator has two sales of [1001,15000] and [100001, 250000], the result will be [101002, 265000]. Note that one senator (the same one who has the largest transaction), will have an upper bound of None (indicating no upper bound). Your result should be a dictionary whose keys are the senators’ names and whose values are their sales sums.

Hints

* Check specifically for a None value in the upper part of the range, and set the sum of two values to None whenever either value is None.

5. Median Transaction by Ticker Symbol (15 pts)

Write a **function** get\_ticker\_median that, given a ticker symbol, returns the **median** transaction range for that ticker symbol. Recall the median is the middle value. For a **sorted** list of values [1, 3, 4, 7, 21], the median is 4; for [1, 3, 4, 7, 13, 21], it is the average of the two middle values 4 and 7 = 5.5. The median range, unlike the sum, will be the middle range (after sorting) if we have an odd number of ranges for a particular ticker, and the union of two middle ranges (the lower bound from lower middle range and upper bound from higher middle range) if we have an even number of ranges. For example, the median of [[0,1], [1,3], [4,7]] is [1,3] while the median of [[0,1], [1,3], [4,7], [8,15]] is [1,7].

For example,

get\_ticker\_median('AAPL') *# returns [1001, 15000]*

get\_ticker\_median('NVDA') *# returns [15001, 50000]*

6. [CSCI 503 Only] Filter by Date (20 pts)

*Only CSCI 503 students need to complete this part. CSCI 490 students may complete it for extra credit.*

Write a function transactions\_in\_range that will filter the sales by date (inclusive). Specifically, given a start date and an end date, return the transactions that fall in that range. Note that you will need to parse the date strings, and then compare the dates in the correct order. Do not use a python library for this, but rather create a tuple that encodes the date and makes the comparison operators work as desired. For example, transactions\_in\_range('9/30/2016','10/3/2016') returns

[{'transaction\_date': '10/03/2016',

'owner': 'Child',

'ticker': 'ASIX',

'asset\_type': 'Stock',

'type': 'Exchange',

'amount\_range': [1001, 15000],

'senator': 'Sheldon Whitehouse'},

{'transaction\_date': '10/03/2016',

'owner': 'Joint',

'ticker': 'ASIX',

'asset\_type': 'Stock',

'type': 'Exchange',

'amount\_range': [1001, 15000],

'senator': 'Sheldon Whitehouse'},

{'transaction\_date': '10/03/2016',

'owner': 'Self',

'ticker': 'ASIX',

'asset\_type': 'Stock',

'type': 'Exchange',

'amount\_range': [1001, 15000],

'senator': 'Sheldon Whitehouse'},

{'transaction\_date': '09/30/2016',

'owner': 'Self',

'ticker': 'DIS',

'asset\_type': 'Stock',

'type': 'Purchase',

'amount\_range': [50001, 100000],

'senator': 'James M Inhofe'},

{'transaction\_date': '09/30/2016',

'owner': 'Self',

'ticker': 'VZ',

'asset\_type': 'Corporate Bond',

'type': 'Purchase',

'amount\_range': [1001, 15000],

'senator': 'Gary C Peters'}]

Hints

* [str.split](https://docs.python.org/3/library/stdtypes.html#str.split) will help
* To compare dates, compare the year, then month, then day
* Make sure to convert strings to ints