Assignment 5

Goals

The goal of this assignment is to work with scripts and packages in Python.

Instructions

You will be doing your work in Python 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 .py files 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 again be working with data from the [Senate Stock Watcher](https://senatestockwatcher.com/), built by Timothy Carambat, that we first used for [Assignment 3](http://faculty.cs.niu.edu/~dakoop/cs503-2021fa/assignment3.html). That data is located [here](http://faculty.cs.niu.edu/~dakoop/cs503-2021fa/a3/senate-stock-trades.json). Remember that 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 writing Python modules, putting them in a package, and writing a script to help analyze this data. While not required, you may find it useful to create a notebook where you can test the modules and programs. You **may** use other core Python modules (e.g. collections) in this assignment.

Due Date

The assignment is due at 11:59pm on Tuesday, October 19.

Submission

You should submit the completed Python files required for this assignment on [Blackboard](https://webcourses.niu.edu/). Zip the files together; the filename of the zipfile should be a5.zip. You can create an archive on tiger (assuming you created an a5 directory above the package and script that is your current working directory) using the following code in a notebook:

import shutil

shutil.make\_archive('../a5', 'zip', '..', 'a5')

Then, download the a5.zip file to turn in via Blackboard. Make sure your archive contains **both** the senate\_stock\_trades package and the analyze\_trades.py program.

Details

Please make sure to follow instructions to receive full credit. To test your code, you may use the %run magic command in the notebook. For example,

%run analyze\_trades.py

You may also use the Terminal in Jupyter on tiger, but you should activate the correct environment by first running:

$ conda activate py39

0. Name & Z-ID (5 pts)

Since we are using Python files (.py) files for this assignment, add the identifying information to the beginning of your analyze\_trades.py program and the \_\_init\_\_.py file of your package. Minimally, you should have 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. Senate Stock Trades Package (50 pts)

Create three new Python modules, one for reading the dataset, one for analyzing trades by ticker symbol, and one for comparing two senators. Put the three modules (util.py, ticker.py, and compare.py) into a package named senate\_stock\_trades.

1a. Data Utilities (15 pts)

Create a util.py module that has three methods: get\_data, add\_amount\_ranges, and sub\_amount\_ranges.

The get\_data method should read and parse the [senate-stock-trades.json](http://faculty.cs.niu.edu/~dakoop/cs503-2021fa/a3/senate-stock-trades.json) datafile and store it in a module variable. Assume that the data file resides in the same directory as util.py. You can then get its absolute path via the \_\_file\_\_ variable of the module via:

import os

fname = os.path.join(os.path.dirname(\_\_file\_\_),'senate-stock-trades.json')

Use the [json module](https://docs.python.org/3/library/json.html) to load the data from the file. Your get\_data method should only read and parse the file from disk **once**, otherwise returning the pre-loaded data.

The add\_amount\_ranges and sub\_amount\_ranges should add and subtract two amount ranges, respectively. Recall that an amount range is a tuple (*min*, *max*) where *max* can be None indicating it has no upper bound. Two ranges *(a,b)* +/- *(c,d)* = *(a+/-c, b+/-d)* unless at least one of *b* or *d* is None, in which case the result is *(a+/-c, None)*.

Hints

* Initialize the module variable to a sentinel value to indicate when the data has not been read.
* You can use [%autoreload](https://ipython.readthedocs.io/en/stable/config/extensions/autoreload.html) to automatically reload modules as you edit them. Do note, however, that this will mask the effects of trying to not keep reloading the data! You can also use importlib.reload to do this manually.

1b. Ticker Analysis (15 pts)

Create an ticker.py module that has two methods that both take one parameter, the ticker symbol. Use the get\_data method from the data module to obtain the data. The first method, count\_trades, should return a dictionary of the form {<senator>: <count>} with the counts of trades for each senator. The second method, sum\_trades, should return a dictionary of the form {<senator>: (<min\_value>,<max\_value)} with the range of possible trade values. Use the util.add\_amount\_ranges method from Part 1a to add the amount ranges.

Hints

* Make sure to import the util module! You might consider using relative imports to do this from a sibling module.
* You may use [collections.Counter](https://docs.python.org/3/library/collections.html" \l "collections.Counter) for count\_trades.
* You could consider using [collections.defaultdict](https://docs.python.org/3/library/collections.html" \l "collections.defaultdict) to help with sum\_trades. You can use a lambda function as the argument to defaultdict to initialize the key-value pairs with a tuple.

1c. Comparison (15 pts)

Create a compare.py module that calculates comparative information between two senators. Given two senators’ names as parameters, the count\_diff method should return the difference between the **number** of transactions between the two senators, and the amount\_diff method should return the *ranged* difference between the amounts of all trades. This difference should be computed by using the util.sub\_amount\_ranges method from Part 1a.

Hints

* Consider testing the functions via code in a notebook. You may also do this in the modules themselves, but remember to make sure they only run when the module is run as a script.

1d. Package (5 pts)

Make sure all three analysis modules live in a single senate\_stock\_trades package. Add an \_\_init\_\_.py file for completeness. It may contain documentation and the pass keyword.

2. Stock Analysis Program (25 pts)

Create a analyze\_trades.py program that uses the package from Part 1 to identify trades of interest and compare senators. The script should process two subcommands; the first is “ticker” and the second is “compare”. The first subcommand prints the results from the count\_trades and sum\_trades methods, and the second subcommand takes the names of two senators as arguments and prints the results from the count\_diff and amount\_diff methods. You can test your script via the IPython magic command %run analyze\_trades.py ... or via the shell command !/opt/conda/envs/py39/bin/python analyze\_trades.py ... (you will need to adjust the path if not using tiger). Make sure to print a usage method if the user misses or provides incorrect arguments. Some sample output:

%run analyze\_trades.py

Usage: python analyze\_trades.py [ticker <ticker> | compare <senator1> <senator2>]

%run analyze\_trades.py ticker

Usage: python analyze\_trades.py [ticker <ticker> | compare <senator1> <senator2>]

%run analyze\_trades.py ticker NFLX

Number of trades:

Pat Roberts: 43

Ron L Wyden: 8

Sheldon Whitehouse: 8

David A Perdue , Jr: 4

Christopher A Coons: 4

John Hoeven: 1

Sum of trade values:

Pat Roberts: (1343043, 3420000)

Ron L Wyden: (71008, 240000)

Sheldon Whitehouse: (50008, 225000)

David A Perdue , Jr: (60004, 200000)

John Hoeven: (50001, 100000)

Christopher A Coons: (4004, 60000)

%run analyze\_trades.py compare "Kelly Loeffler" "David A Perdue , Jr"

Kelly Loeffler has -2318 trades with value +(10969682, 68780000) than David A Perdue , Jr

%run analyze\_trades.py compare "Kelly Loeffler" "James M Inhofe"

Kelly Loeffler has +50 trades with value -(12291950, None) than James M Inhofe

Hints

* Create a usage method that can be called whenever there is trouble
* You will need to pass the names in quotes because of the spaces which the shell will otherwise decompose into separate arguments.
* You will need a different number of arguments depending on which subcommand is called
* Note that showing the +/- sign for the range will require figuring out whether the amounts are less than or greater than zero

3. [CSCI 503 Only] Add Date Filtering (20 pts)

For this part, you will add date filtering to the package and program you wrote in Parts 1 and 2. This should not require significant changes to the overall logic, and your final package and library should work both for unfiltered data and for filtered data. Turn in this final package and program with the additional filtering added.

3a. Add Filtering to the Package (10 pts)

Add the ability to restrict the calculations by date to the compare.py module. Thus, count\_diff and amount\_diff should take **two** optional parameters that set the start date and end date, respectively. If they are not set, the start date is the earliest date in the data, and the end date is the latest date. The range is inclusive. The methods should now return the differences for the senators for only the trades between those dates, **inclusive**.

You may choose to parse the transaction\_date in the data to a tuple (as in Assignment 3) or to a date object using the datetime.date library. You may also choose which format the count\_diff and amount\_diff methods take (e.g. a tuple *(year, month, day)* or a date object), but you need to document this as a docstring in those methods. You will be passing arguments that adhere to your format in Part 3b.

Hints

* Use a sentinel value to indicate when the start or end date are not set. You can use an or expression to check if either date is unset or the criteria is satisfied.

3b. Add Filtering to the Program (10 pts)

Add support for the date filtering in your script. To do so, we will require that the user specify the argument **after** the senators’ names and specify the date range in the format YYYY-mm-dd:YYYY-mm-dd. If the date range is specified, pass the **parsed** individual dates to the correct parameters of count\_diff and amount\_diff. You will need to parse the string to split the two dates and convert them to the form required by Part 3a. Some sample output:

%run analyze\_trades.py compare

Usage: python analyze\_trades.py [ticker <ticker> | compare <senator1> <senator2> [start-date:end-date]]

%run analyze\_trades.py compare "David A Perdue , Jr" "Kelly Loeffler" 2016-08-01:2017-07-01

David A Perdue , Jr has +296 trades with value +(2330296, 8800000) than Kelly Loeffler

%run analyze\_trades.py compare "Kelly Loeffler" "James M Inhofe" 2018-01-01:

Kelly Loeffler has +180 trades with value +(42140180, 176475000) than James M Inhofe

Hints:

* Use the split function multiple times to parse the date string
* Make sure to parse the extracted strings to integers
* Make sure your code still runs if the entire date range is **not** specified
* Make sure to detect when either the start or end date is not specified
* Remember to update the usage string

Extra Credit

* [20 pts] CSCI 490 Students may complete Part 3 for extra credit
* [5 pts] Use the [operator package](https://docs.python.org/3/library/operator.html) to refactor the add\_amount\_ranges and sub\_amount\_ranges methods to use a common shared method.
* [5 pts] Add support for matching senators’ names even if they are only partially specified (e.g. just a last name). Make sure that ambiguous cases generate an error.
* [5 pts] Extend Part 3b to allow the date range to be specified as a -d argument that can be before or after the senators names. For example,

$ python analyze\_trades.py -d :2018-07-01 "Kelly Loeffler" "David A Perdue , Jr"

* [5 pts] Add support for date ranges to specify only the year or year and month. In these cases, the program should assume that the start date begins on the first day of the month/year, and the end date ends on the **last** day of the month/year.