api_data_wrangling_mini_project

November 1, 2020

This exercise will require you to pull some data from the Qunadl API. Qaundl is currently the most widely used aggregator of financial market data.

As a first step, you will need to register a free account on the http://www.quandl.com website.

After you register, you will be provided with a unique API key, that you should store:

```
[1]: # Store the API key as a string - according to PEP8, constants are always named␣
     ↪in all upper case
     API_KEY = 'Ss8nF4i_V2kFAXfdD_VW'
```

Qaundl has a large number of data sources, but, unfortunately, most of them require a Premium subscription. Still, there are also a good number of free datasets.

For this mini project, we will focus on equities data from the Frankfurt Stock Exhange (FSE), which is available for free. We'll try and analyze the stock prices of a company called Carl Zeiss Meditec, which manufactures tools for eye examinations, as well as medical lasers for laser eye surgery: https://www.zeiss.com/meditec/int/home.html. The company is listed under the stock ticker AFX_X.

You can find the detailed Quandl API instructions here: https://docs.quandl.com/docs/time-series

While there is a dedicated Python package for connecting to the Quandl API, we would prefer that you use the *requests* package, which can be easily downloaded using *pip* or *conda*. You can find the documentation for the package here: http://docs.python-requests.org/en/master/

Finally, apart from the *requests* package, you are encouraged to not use any third party Python packages, such as *pandas*, and instead focus on what's available in the Python Standard Library (the *collections* module might come in handy: https://pymotw.com/3/collections/). Also, since you won't have access to DataFrames, you are encouraged to us Python's native data structures - preferably dictionaries, though some questions can also be answered using lists. You can read more on these data structures here: https://docs.python.org/3/tutorial/datastructures.html

Keep in mind that the JSON responses you will be getting from the API map almost one-to-one to Python's dictionaries. Unfortunately, they can be very nested, so make sure you read up on indexing dictionaries in the documentation provided above.

```
[22]: # First, import the relevant modules
      import requests
      import json
      import math
```

```
[5]: # Now, call the Quandl API and pull out a small sample of the data (only one␣
     ↪day) to get a glimpse
     # into the JSON structure that will be returned
     data = requests.get('https://www.quandl.com/api/v3/datasets/FSE/AFX_X?api_key='␣
     ↪+ API_KEY)
     print(data)
```

```
<Response [200]>
```

```
[17]: # Inspect the JSON structure of the object you created, and take note of how␣
      ↪nested it is,
      # as well as the overall structure
      print(json.loads(data.text)["dataset"]["column_names"])
```

```
['Date', 'Open', 'High', 'Low', 'Close', 'Change', 'Traded Volume', 'Turnover',
'Last Price of the Day', 'Daily Traded Units', 'Daily Turnover']
```

These are your tasks for this mini project:

1. Collect data from the Franfurt Stock Exchange, for the ticker AFX_X, for the whole year 2017 (keep in mind that the date format is YYYY-MM-DD).
2. Convert the returned JSON object into a Python dictionary.
3. Calculate what the highest and lowest opening prices were for the stock in this period.
4. What was the largest change in any one day (based on High and Low price)?
5. What was the largest change between any two days (based on Closing Price)?
6. What was the average daily trading volume during this year?
7. (Optional) What was the median trading volume during this year. (Note: you may need to implement your own function for calculating the median.)

```
[18]: year_2017_data = requests.get('https://www.quandl.com/api/v3/datasets/FSE/AFX_X?
      ↪start_date=2017-01-01&end_date=2018-01-01api_key=' + API_KEY)
```

```
[31]: data_json = json.loads(year_2017_data.text)
      dataset = data_json["dataset"]
      data = dataset["data"]
```

```
[33]: #print(dataset)
      highest_opening = 0
      lowest_opening = math.inf
      for row in data:
          if row[1] != None and row[1] > highest_opening:
              highest_opening = row[1]
          if row[1] != None and row[1] < lowest_opening:
              lowest_opening = row[1]

      print(highest_opening, lowest_opening)
```

```
53.11 34.0
```

```
[43]: largest_change_highlow = 0
      for row in data:
          if row[2] - row[3] > largest_change_highlow:
              largest_change_highlow = row[2] - row[3]

      print(largest_change_highlow)
```

```
2.8100000000000023
```

```
[44]: largest_change_close = 0
      for row in data:
          if row[4] != None and row[1] != None and (row[4] - row[1]) >␣
       ↪largest_change_close:
              largest_change_close = (row[4] - row[1])

      print(largest_change_close)
```

```
1.6400000000000006
```

```
[45]: avg_trading_vol = 0
      sum_trading_vol = 0
      for row in data:
          sum_trading_vol += row[6]

      avg_trading_vol = sum_trading_vol / len(list(data))
      print(avg_trading_vol)
```

```
89124.33725490196
```

```
[51]: def median(lst):
          sorted_lst = sorted(lst)
          length = len(sorted_lst)
          if length % 2 == 0:
              return (sorted_lst[int(length/2)] + sorted_lst[int(length/2)+1]) / 2
          else :
              return sorted_lst[int(length/2)]
```

```
[54]: trading_volume = []
      for row in data:
          trading_volume.append(row[6])

      print(median(trading_volume))
```

```
76286.0
```