This exercise will require you to pull some data from https://data.nasdaq.com/ (formerly Quandl API).

As a first step, you will need to register a free account on the https://data.nasdaq.com/ website.

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

Note: Use a `.env` file and put your key in there and `python-dotenv` to access it in this notebook.

The code below uses a key that was used when generating this project but has since been deleted. Never submit your keys to source control. There is a `.env-example` file in this repository to illusrtate what you need. Copy that to a file called `.env` and use your own api key in that `.env` file. Make sure you also have a `.gitignore` file with a line for `.env` added to it.

The standard Python gitignore is here you can just copy that.

In [5]:
```
# get api key from your .env file
import os
from dotenv import load_dotenv

load_dotenv()
API_KEY = os.getenv('NASDAQ_API_KE# Y')

print(API_KEY)
```

```
None
```

Nasdaq Data 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 Nasdaq Data API instructions here: https://docs.data.nasdaq.com/docs/in-depth-usage

While there is a dedicated Python package for connecting to the Nasdaq 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.

In [12]:
```
# First, import the relevant modules
import requests
```

Note: API's can change a bit with each version, for this exercise it is reccomended to use the nasdaq api at `https://data.nasdaq.com/api/v3/`. This is the same api as what used to be quandl so `https://www.quandl.com/api/v3/` should work too.

Hint: We are looking for the `AFX_X` data on the `datasets/FSE/` dataset.

In [14]:
```
# Now, call the Nasdaq API and pull out a small sample of the data (only one day) t
# into the JSON structure that will be returned
```

In [15]:
```
url = "https://data.nasdaq.com/api/v3/datasets/FSE/AFX_X.json"
params = {
    "start_date": "2020-11-02",
    "end_date": "2020-11-02",
    "api_key": API_KEY
}
r = requests.get(url, params=params)
json_data = r.json()
```

In [16]:
```
# Inspect the JSON structure of the object you created, and take note of how nested
# as well as the overall structure
```

In [17]:
```
print(json_data)
```

```
{'dataset': {'id': 10095370, 'dataset_code': 'AFX_X', 'database_code': 'FSE', 'nam
e': 'Carl Zeiss Meditec (AFX_X)', 'description': 'Stock Prices for Carl Zeiss Medite
c (2020-11-02) from the Frankfurt Stock Exchange.<br><br>Trading System: Xetra<br><b
r>ISIN: DE0005313704', 'refreshed_at': '2020-12-01T14:48:09.907Z', 'newest_available
_date': '2020-12-01', 'oldest_available_date': '2000-06-07', 'column_names': ['Dat
e', 'Open', 'High', 'Low', 'Close', 'Change', 'Traded Volume', 'Turnover', 'Last Pri
ce of the Day', 'Daily Traded Units', 'Daily Turnover'], 'frequency': 'daily', 'typ
e': 'Time Series', 'premium': False, 'limit': None, 'transform': None, 'column_inde
x': None, 'start_date': '2020-11-02', 'end_date': '2020-11-02', 'data': [['2020-11-0
2', 111.3, 111.6, 108.1, 108.1, None, 362.0, 39844.5, None, None, None]], 'collaps
e': None, 'order': None, 'database_id': 6129}}
```

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.)

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).

```
In [20]: # use url with parameters for 2017
url = "https://data.nasdaq.com/api/v3/datasets/FSE/AFX_X.json"
params = {
    "start_date": "2017-01-01",
    "end_date": "2017-12-31",
    "api_key": API_KEY
}
r_2017 = requests.get(url, params=params)
```

2. Convert the returned JSON object into a Python dictionary.

```
In [22]: data_2017 = r_2017.json()
```

3. Calculate what the highest and lowest opening prices were for the stock in this period.

```
In [24]: # check keys to see where to find the open price
data_2017['dataset']['column_names']
```

```
Out[24]: ['Date',
 'Open',
 'High',
 'Low',
 'Close',
 'Change',
 'Traded Volume',
 'Turnover',
 'Last Price of the Day',
 'Daily Traded Units',
 'Daily Turnover']
```

```
In [25]: # find the index of the opening prices
open_index = data_2017['dataset']['column_names'].index('Open')
```

In [26]:
```
# use list comprehension to gather all open prices
open_price = [day[open_index] for day in data_2017['dataset']['data']]
```

In [27]:
```
# some are type None, so filter those out
open_price = [price for price in open_price if price is not None]
```

In [28]:
```
# find min and max values and print those out
min_price = "%.2f" % (min(open_price))
max_price = "%.2f" % (max(open_price))
print('The highest opening price in 2017 was $' + str(max_price))
print('The lowest opening price in 2017 was $' + str(min_price))
```

```
The highest opening price in 2017 was $53.11
The lowest opening price in 2017 was $34.00
```

4. What was the largest change in any one day (based on High and Low price)?

In [30]:
```
# find the index of the high and low prices
high_index = data_2017['dataset']['column_names'].index('High')
low_index = data_2017['dataset']['column_names'].index('Low')
```

In [31]:
```
# use list comprehension to gather the difference in high and low prices for each d
diff_price = [day[high_index] - day[low_index] for day in data_2017['dataset']['dat
```

In [32]:
```
# find the largest change and print it out
largest_diff = "%.2f" % (max(diff_price))
print('The largest change in any one day based on high and low price in 2017 was $'
```

```
The largest change in any one day based on high and low price in 2017 was $2.81
```

5. What was the largest change between any two days (based on Closing Price)?

In [34]:
```
# find the index of the close prices
close_index = data_2017['dataset']['column_names'].index('Close')
```

In [35]:
```
# use list comprehension to gather closing prices for each day
closing = [day[close_index] for day in data_2017['dataset']['data']]
```

In [36]:
```
# use list comprehension to gather absolute value of difference in closing price be
close_diff = [abs(closing[index]-closing[index-1]) for index, value in enumerate(cl
```

In [37]:
```
# find the largest value and print it out
largest_close_diff = "%.2f" % (max(close_diff))
print('The largest change between any two days based on closing price in 2017 was $
```

```
The largest change between any two days based on closing price in 2017 was $2.56
```

6. What was the average daily trading volume during this year?

```
In [39]: # find the index of the trading volume
         trading_index = data_2017['dataset']['column_names'].index('Traded Volume')
```

```
In [40]: # use list comprehension to gather trading volume for each day
         trading_volume = [day[trading_index] for day in data_2017['dataset']['data']]
```

```
In [41]: # find the average trading volume and print it out
         average_volume = "%.2f" % (sum(trading_volume)/len(trading_volume))
         print('The average daily trading volume in 2017 was $' + str(average_volume))

         The average daily trading volume in 2017 was $89124.34
```

7. (Optional) What was the median trading volume during this year. (Note: you may need to implement your own function for calculating the median.)

```
In [43]: # sort values
         trading_volume = sorted(trading_volume)
```

```
In [44]: # calculate median
         if len(trading_volume) % 2 == 1:
             median_volume = trading_volume[len(trading_volume)//2]
         else:
             median_volume = (trading_volume[len(trading_volume)//2] + trading_volume[len(tr
         median_volume =  "%.2f" % (median_volume)
         print('The median trading volume in 2017 was $' + str(median_volume))

         The median trading volume in 2017 was $76286.00
```

```
In [ ]:
```