AM02 Group Assignment 1
Introduction to Python for Data Science

Please ensure to only use the topics we have covered so far (i.e. no Pandas lib please). If in doubt please double check with me by messaging me on Slack or via email.

Q1) Moving Average Cross-over Trading Strategy

This qn is aimed to prepare you for the moving-average trading strategy programming task. Before progressing onto using NumPy and Pandas data structures for solving this task, you should become familiar with how this could be done using lists. Often in Python we end up using the commands which are contained inside a particular library, but we may not get a feel for how to design such code ourselves. Therefore let us solidify our fundamentals of coding.

A **simple n-period moving average** is calculated for a time series using an average of the last `n` values, where for our time series we use prices:

$$MA_n(t) = \frac{p_t + p_{t-1} + \ldots + p_{t-(n-1)}}{n}$$

Note: for a 3-period moving average, the first 2 MA values are `None`, and the 3rd value is the average of the first 3 values.

Over the Covid-19 period airline stocks have taken a huge hit. The famous Berkshire Heathaway, run by Warren Buffett, has sold its US airline shares back in May (article). Let's examine Boeing share price behaviour over the last 5 year period, using Daily Adjusted Closing prices.

a) Import the data for Boeing stock prices from file called `BA.csv` and prepare it by creating a list called `prices`. Use the following steps (do not use Pandas library):
   - Use `readlines()` function to read in the columns of the csv file into a variable called `data`.
   - Split away the header from the rest of the data, hence forming 2 variables `header` and `allData`.
   - Obtain the number of observations and store it into variable called, `N`.
   - Next use a `for` loop to process each line of the csv file such that you obtain: a list of `dates` and a list of `prices`.
     Hint: use a `for` loop to access each element of the data in the `allData` list, process each string and element by element build the `dates` and `prices` lists.

b) Using the definition of Moving Average above, write code to calculate an n-period moving average list, called `ma`. Ensure that your code is flexible enough to accommodate any period `n`, not just a hard-coded single value.

   Reminder of variable meaning:

`prices`	: (list) prices data to be used to obtain moving average
`n`	: (int) number defining moving-average lag (e.g. 252)
`ma`	: (list) moving average data of length N, where first (n-1) values are None

   Hint: Your code will likely utilise nested `for` loops.

c) Perform a simple test of your code to ensure that it is working correctly using 'dummy' `prices` list containing only a few entries, and use a moving average period of n = 3:
   `prices = [2,3,4,5,8,5,4,3,2,1]; n = 3`

You should obtain:

```
[None, None, 3.0, 4.0, 5.666666666666667, 6.0, 5.666666666666667,
4.0, 3.0, 2.0]
```

d) Now obtain the moving-average series for Boeing prices using `n = 252` days which is typically used in the financial industry, store the result into `ma_252` variable. Ensure that the length of your `ma` list is the same as the time series length and store `None` into time steps without the moving average readings.

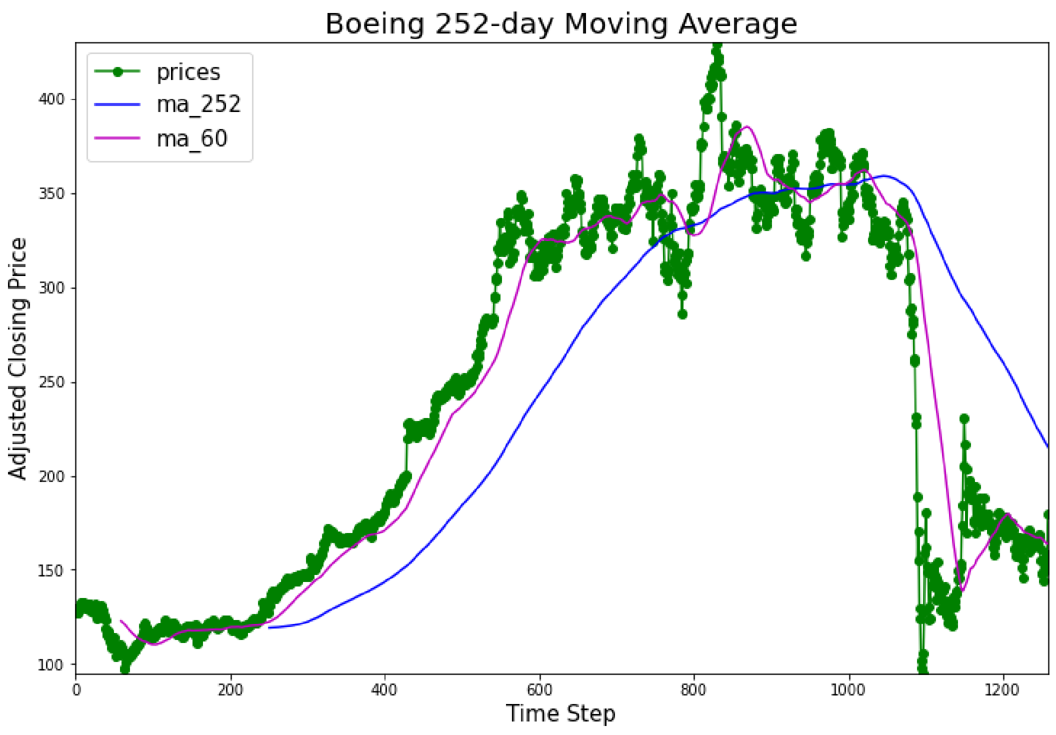
e) Use your knowledge of list slicing to obtain elements in position 251, 252. What is the first moving average value in your series?

f) Obtain another moving average of 60-day period, call it `ma_60`.

[EXTRA, not graded:] Use the following code for a quick visual of your results:

```
import matplotlib.pyplot as plt
plt.figure(figsize=(12, 8))
plt.plot(prices, 'go-', label = "prices")
plt.plot(ma_252, 'b-', label = "ma_252")
plt.plot(ma_60, 'm-', label = "ma_60")
plt.title('Boeing 252-day Moving Average', fontsize = 20)
plt.xlabel('Time Step', fontsize = 15)
plt.ylabel('Adjusted Closing Price', fontsize = 15)
plt.autoscale(enable = True, tight = True) # or for specific
axis axis='x'
plt.legend(loc = "upper left", fontsize = 15)
plt.show()
```

It should look like this (note that moving average cross over is used as a buy/sell signal in technical analysis. In this case the short term moving average of 60 days crossed and went below the long term moving average of 252 days, signalling a 'sell'):



The validity of such a strategy should be backtested before being utilised. We will do so later using Numpy and Pandas libraries.

Q2. Key Financial Statistics Calculations

If you perform quantitative analysis for a firm, it is likely that you would be asked to calculate key financial statistics of an instrument. Your task is to perform basic financial analysis on Facebook stock (note it does not pay dividends), using only `for` loops and functions available with base distribution (i.e. not part of imported libraries), such that you obtain the following:

`R`	: (float) sum total of simple daily returns over time series
`muR`	: (float) mean of simple daily returns
`sigmaR`	: (float) std dev of simple daily returns
`highPt`	: (float) highest time series point of prices
`lowPt`	: (float) lowest time series point of prices
`muR_annual`	: (float) mean of simple daily returns (annualised)
`sigmaR_annual`	: (float) volatility of simple daily returns (annualised)
`SR_annual`	: (float) Sharpe Ratio calculated using annualised statistics

Useful Information

Returns:
Log returns are calculated using the following formula:

$$R_t = \log\left(\frac{P_t}{P_{t-1}}\right)$$

where P_t is price at time step t.

Mean:
Annualised mean is obtained using the formula:

$$\mu_{anualised} = \mu_{daily} * 252$$

where μ_{daily} is the daily simple returns mean.

Volatility:
It is common to assume that there are 252 business days of the year when trading can happen. Therefore the annualised realised volatility (note this is a random quantity) is calculated as: (for further info visit the Financial Volatility Wiki Page)

$$\sigma_{anualised} = \sigma_{daily} * \sqrt{252}$$

where the daily volatility is calculated as:

$$\sigma_{daily} = \sqrt{\frac{1}{T-1}\sum_{t=1}^{T}(R_t - \bar{R})^2}$$

where $\bar{R}$ is the average daily return, and T is the number of days for which returns were calculated.

Sharpe Ratio:
Is a measure of excess return of the instrument / strategy / portfolio over risk-free rate adjusted by its volatility (all measures annualised), calculated as follows:

$$SR = \frac{\bar{R}_{anualised} - rf_{anualised}}{\sigma_{anualised}}$$

Note that the returns are assumed to be normally distributed, which is often not the case. The Sharpe ratio therefore has a shortcoming, since it only holds for normal returns.

Risk Free Rate:
Note: we will use the annualised risk free rate of 1.1555% (i.e. 0.011555), which is approximately the average of the FED Base Rate over period of 01-11-2015 to 01-10-2020 which is the period of our data set (data obtained from FRED Economic Data).

A measure which would typically be used in the industry is the 1-month London Interbank Offered Rate for USD (USD1MTD156N), which is produced daily at 11am. These values are quoted in an annualised format.

Another option is to use the 3-month Treasury yield taken from monthly data from the Federal Reserve Bank of St. Louis.

Instructions

Let us continue working on Boeing data which you currently have stored in `prices` list, however this time we will perform simple financial data analysis on the data.

a) Initialise variables business days to `252` (call it `busDays`), and risk-free rate to `0.011555` (call it `rf`) i.e. 1.1555%.

b) Calculate log returns and store results into a list (call it `R`) using a `for` loop. You are allowed to use math function `log()`.

c) Next, use a `for` loop to calculate the daily mean (call it `muR`), and following that convert it to the annualised mean (call it `muR_annual`).

d) Use a `for` loop to calculate the daily variance of the log returns (call it `varR`). Following this calculate the daily volatility (call it `sigmaR`) and the annualised volatility (call it `sigmaR_annual`).

e) Use a `for` loop to obtain the highest and lowest **price** time series points (call variables: `highPt`, `lowPt`).

f) Calculate the Sharpe Ratio of the time series using annualised statistics (call it `SR`).

g) Print the result for `muR`, `muR_annual`, `sigmaR`, `sigmaR_annual`, `SR`. Given the Sharpe Ratio you found, do you believe this is a good stock to invest in (see link)?

   Also a note: remember that historically investing in an index (such as the S&P500) yields approx. 6% annual return on investment.

h) Use a docstring to report the results, i.e. the bottom of your code should contain:

```
'''
muR: your result
muR_annual: your result
etc…
'''
```