

Metadata

Course: DS 5100
Module: 07 Python Classes
Topic: HW 07 Stock Class
Author: R.C. Alvarado
Date: 7 July 2023

Student Info

- Name: Rachel Holman
- Net UD: dnw9qk
- URL of this file in GitHub: <https://github.com/rachel-holman/DS5100-dnw9qk/blob/main/lessons/M07/hw07.ipynb>

Instructions

In your **private course repo on Rivanna**, use this Jupyter notebook and the data file described to write code that performs the tasks below.

Save your notebook in the **M07** directory.

Remember to add and commit these files to your repo.

Then push your commits to your repo on GitHub.

Be sure to fill out the **Student Info** block above.

To submit your homework, save the notebook as a PDF and upload it to GradeScope, following the instructions.

TOTAL POINTS: 12

Overview

In this assignment you will define a class and use it to perform the requested tasks.

Before answering the questions, read the market data from **apple_data.csv** into a Pandas dataframe. The file is in the HW for this module in the course repo.

Setting Up

```
In [1]: import pandas as pd
import numpy as np
```

Prepare the Data

Read in the dataset from the attached file `apple_data.csv` using `pd.read_csv()`.

```
In [2]: apple = pd.read_csv('apple_data.csv')
apple
```

```
Out[2]:
```

	date	adj_close
0	2020-01-02	298.829956
1	2020-01-03	295.924713
2	2020-01-06	298.282715
3	2020-01-07	296.879883
4	2020-01-08	301.655548
...
130	2020-07-09	383.010010
131	2020-07-10	383.679993
132	2020-07-13	381.910004
133	2020-07-14	388.230011
134	2020-07-15	390.899994

135 rows × 2 columns

Task 1

(5 PTS)

Define a class with these features:

Class Name: `Stock`

Attributes:

- `ticker`: a string to hold the stock symbol
- `sector`: a string to hold the sector name
- `prices`: a dataframe to hold the prices for the stock

Methods:

- `print_sector` to just print out the sector string.

- `get_row_count` to count the number of rows in the price dataframe. Set an attribute "price_records" equal to this count.
- `__init__` to build objects. Initialize with the three attribute values passed to the constructor.

```
In [3]: class Stock:

    def __init__(self, ticker, sector, prices):
        self.ticker = ticker
        self.sector = sector
        self.prices = prices

    def print_sector(self):
        print(self.sector)

    def get_row_count(self):
        self.price_records = self.prices.shape[0]
```

Task 2

(1 PT)

Create an instance of your class with the these initial values:

- `ticker`: 'AAPL'
- `sector`: 'technology'
- `prices`: *the imported price dataframe*

Then Use the dot operator to print the stock's ticker.

```
In [4]: newstock = Stock('AAPL', 'technology', apple)
newstock.ticker
```

```
Out[4]: 'AAPL'
```

Task 3

(1 PT)

Use the `print_sector()` method to print the sector.

```
In [5]: newstock.print_sector()

technology
```

Task 4

(2 PTS)

Use the `get_row_count()` method to compute the number of price records and set `price_records`.

Use the dot operator to access the stock's `price_records`, printing the result.

```
In [6]: newstock.get_row_count()
newstock.price_records
```

Out[6]: 135

Task 5

(1 PT)

Add a new column called `'month'` to the `prices` attribute and put the month number there.

Hint: You can use `.apply()` with a lambda function to split the month string and keep the second element.

```
In [7]: newstock.prices['month'] = newstock.prices['date'].str[5:7].astype(int)
newstock.prices
```

Out[7]:

	date	adj_close	month
0	2020-01-02	298.829956	1
1	2020-01-03	295.924713	1
2	2020-01-06	298.282715	1
3	2020-01-07	296.879883	1
4	2020-01-08	301.655548	1
...
130	2020-07-09	383.010010	7
131	2020-07-10	383.679993	7
132	2020-07-13	381.910004	7
133	2020-07-14	388.230011	7
134	2020-07-15	390.899994	7

135 rows × 3 columns

```
In [8]: # Alternative method using .apply()
# newstock.prices['month'] = newstock.prices['date'].apply(lambda x: x[5:7]).as
```

Task 6

(1 PT)

Use `.groupby()` to compute the mean `adj_close` by month. Save your result as a dataframe, not a series.

```
In [9]: means = pd.DataFrame({'mean_adj_close': newstock.prices.groupby('month')['adj_c
means
```

```
Out[9]:      mean_adj_close
```

month	
1	310.337596
2	310.271843
3	261.735581
4	271.650839
5	309.785164
6	345.806360
7	378.385999

```
In [10]: # Another method
means2 = newstock.prices.groupby('month').agg({'adj_close': 'mean'})
means2
```

```
Out[10]:      adj_close
```

month	
1	310.337596
2	310.271843
3	261.735581
4	271.650839
5	309.785164
6	345.806360
7	378.385999

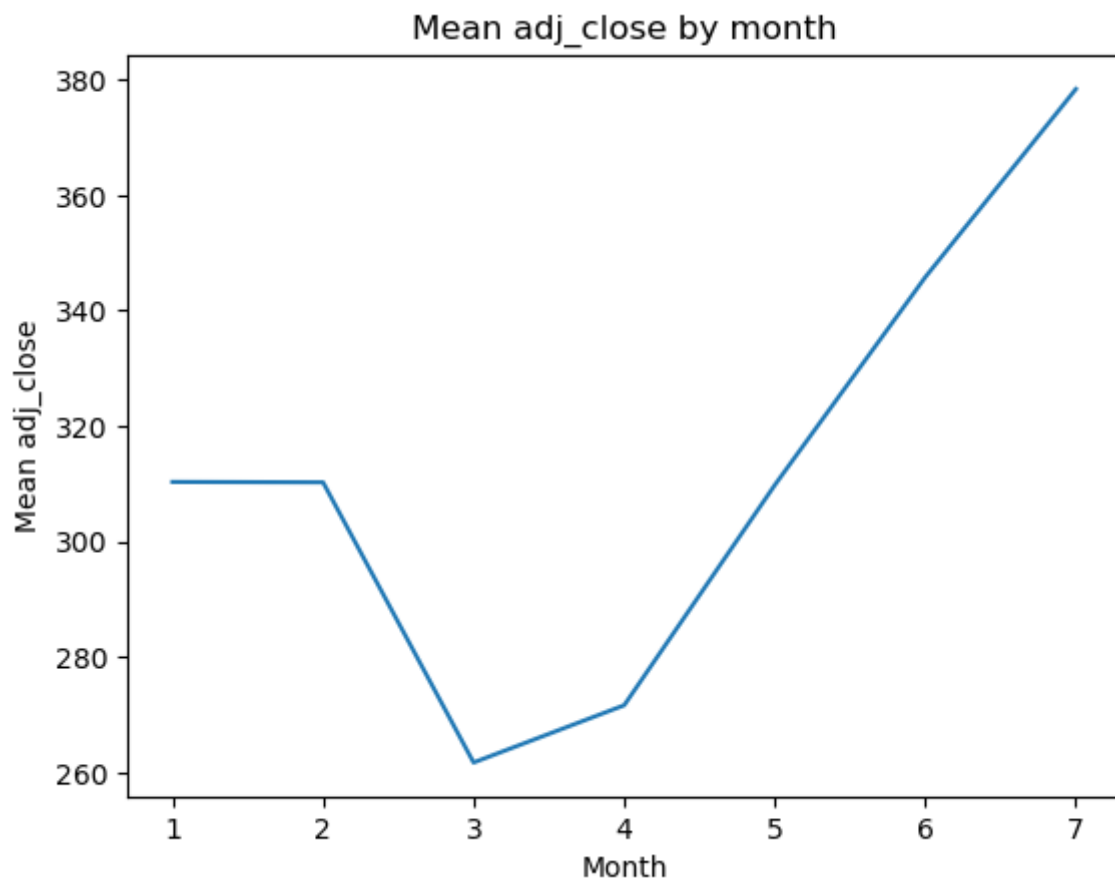
Task 7

(1 PT)

Plot the mean `adj_close` by month using a simple line plot.

```
In [11]: import matplotlib.pyplot as plt
plt.plot(means)
plt.xlabel('Month')
plt.ylabel('Mean adj_close')
plt.title('Mean adj_close by month')
```

```
Out[11]: Text(0.5, 1.0, 'Mean adj_close by month')
```



```
In [ ]:
```