

Metadata

Course: DS 5100
Module: 07 Python Classes
Topic: HW 07 Stock Class
Author: R.C. Alvarado
Date: 29 June 2022

Student Info

- Name:
- Net UD:
- URL of this file in GitHub:

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.

Setting Up

```
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()`.

CODE HERE

```
df = pd.read_csv("apple_data.csv")
```

```
df
```

	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 x 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.

CODE HERE

```
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 = len(self.prices)

```

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.

CODE HERE

```
my_stock = Stock('AAPL', 'technology', df)
```

Task 3

(1 PT)

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

CODE HERE

```
my_stock.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.

CODE HERE

```
my_stock.get_row_count()
my_stock.price_records
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.

```
# CODE HERE
my_stock.prices['month'] = my_stock.prices['date'].apply(lambda x: x.split('-')[1])
# Another method
# my_stock.prices['month'] = my_stock.prices['date'].str.split('-').str[1]
```

Task 6

(1 PT)

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

```
# CODE HERE
acm_month = my_stock.prices.groupby('month').adj_close.mean().to_frame()
acm_month
      adj_close
month
01    310.337596
02    310.271843
03    261.735581
04    271.650839
05    309.785164
06    345.806360
07    378.385999

# Another method
# my_stock.prices.groupby('month').agg({'adj_close': 'mean'})
```

Task 7

(1 PT)

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

CODE HERE

```
acm_month.plot();
```

