Module 7: Data Wrangling with Pandas

CPE311 Computational Thinking with Python

Submitted by: Masangkay, Frederick Performed on: 04-06-2025 Submitted on:

Submitted to: Engr. Roman M. Richard

Exercise 1

We want to look at data for the Facebook, Apple, Amazon, Netflix, and Google (FAANG) stocks, but we were given each as a separate CSV file. Combine them into a single file and store the dataframe of the FAANG data as faang for the rest of the exercises:

1. Read each file in.
2. Add a column to each dataframe, called ticker, indicating the ticker symbol it is for (Apple's is AAPL, for example). This is how you look up a stock. Each file's name is also the ticker symbol, so be sure to capitalize it.
3. Append them together into a single dataframe.
4. Save the result in a CSV file called faang.csv.

In [2]:
```
import pandas as pd
```

In [3]:
```
# reading all the uploaded csv files
aapl = pd.read_csv('./sample-data/aapl.csv')
amzn = pd.read_csv('./sample-data/amzn.csv')
fb = pd.read_csv('./sample-data/fb.csv')
goog = pd.read_csv('./sample-data/goog.csv')
nflx = pd.read_csv('./sample-data/nflx.csv')
```

In [4]:
```
# add appropriate ticker to each dataframe
aapl['ticker'] = 'AAPL'
amzn['ticker'] = 'AMZN'
fb['ticker'] = 'FB'
goog['ticker'] = 'GOOG'
nflx['ticker'] = 'NFLX'
```

In [5]:
```
# Merging multiple CSV files
df = pd.concat([aapl, amzn, goog, fb, nflx], ignore_index=True)
df
```

Out[5]:

	date	open	high	low	close	volume	ticker
0	2018-01-02	166.9271	169.0264	166.0442	168.9872	25555934	AAPL
1	2018-01-03	169.2521	171.2337	168.6929	168.9578	29517899	AAPL
2	2018-01-04	169.2619	170.1742	168.8106	169.7426	22434597	AAPL
3	2018-01-05	170.1448	172.0381	169.7622	171.6751	23660018	AAPL
4	2018-01-08	171.0375	172.2736	170.6255	171.0375	20567766	AAPL
...	...	...	...	...	...	...	...
1250	2018-12-24	242.0000	250.6500	233.6800	233.8800	9547616	NFLX
1251	2018-12-26	233.9200	254.5000	231.2300	253.6700	14402735	NFLX
1252	2018-12-27	250.1100	255.5900	240.1000	255.5650	12235217	NFLX
1253	2018-12-28	257.9400	261.9144	249.8000	256.0800	10987286	NFLX
1254	2018-12-31	260.1600	270.1001	260.0000	267.6600	13508920	NFLX

1255 rows × 7 columns

In [6]:
```
df.to_csv('faang.csv') # convert to csv
```

Exercise 2

- With `faang`, use type conversion to change the `date` column into a datetime and the `volume` column into integers. Then, sort by `date` and `ticker`.
- Find the seven rows with the highest value for `volume`.
- Right now, the data is somewhere between long and wide format. Use `melt()` to make it completely long format.
  Hint: `date` and `ticker` are our ID variables (they uniquely identify each row). We need to melt the rest so that we don't have separate columns for `open`, `high`, `low`, `close`, and `volume`.

In [7]:
```
faang = pd.read_csv('./faang.csv')
```

In [8]:
```
faang['date'] = pd.to_datetime(faang['date']) # change data type to datetime
print(faang['date'].dtype)
```

```
datetime64[ns]
```

In [9]:
```
faang['volume'].dtype # the data type of colume is already int
```

Out[9]:
```
dtype('int64')
```

In [10]:
```
# Find the seven rows with the highest value for volume.
faang.sort_values(by = 'volume')
faang.head(7)
```

Out[10]:

	Unnamed: 0	date	open	high	low	close	volume	ticker
0	0	2018-01-02	166.9271	169.0264	166.0442	168.9872	25555934	AAPL
1	1	2018-01-03	169.2521	171.2337	168.6929	168.9578	29517899	AAPL
2	2	2018-01-04	169.2619	170.1742	168.8106	169.7426	22434597	AAPL
3	3	2018-01-05	170.1448	172.0381	169.7622	171.6751	23660018	AAPL
4	4	2018-01-08	171.0375	172.2736	170.6255	171.0375	20567766	AAPL
5	5	2018-01-09	171.2337	171.7340	170.1154	171.0179	21583997	AAPL
6	6	2018-01-10	169.8701	170.9884	169.7131	170.9786	23959895	AAPL

In [11]:
```
pd.melt(faang, id_vars=['date'], value_vars=['open', 'high', 'low', 'close', 'volum
```

Out[11]:

	date	variable	value
0	2018-01-02	open	1.669271e+02
1	2018-01-03	open	1.692521e+02
2	2018-01-04	open	1.692619e+02
3	2018-01-05	open	1.701448e+02
4	2018-01-08	open	1.710375e+02
...	...	...	...
6270	2018-12-24	volume	9.547616e+06
6271	2018-12-26	volume	1.440274e+07
6272	2018-12-27	volume	1.223522e+07
6273	2018-12-28	volume	1.098729e+07
6274	2018-12-31	volume	1.350892e+07

6275 rows × 3 columns

Exercise 3

- Using web scraping, search for the list of the hospitals, their address and contact information. Save the list in a new CSV file, `hospitals.csv`.

- Using the generated `hospitals.csv`, convert the CSV file into a pandas dataframe. Prepare the data using the necessary preprocessing techniques.

In [24]:
```
import requests
from bs4 import BeautifulSoup
import pandas as pd
```

In [25]:
```
url = "https://shop.philcare.com.ph/accredited-hospitals"
```

In [26]:
```
page = requests.get(url)
```

In [27]:
```
soup = BeautifulSoup(page.text, 'html')
```

In [60]:
```
hospitals_table = soup.find_all('table')[0]
```

In [61]:
```
hospital_header = hospitals_table.find_all('th')
hospital_header
```

Out[61]:
```
[<th>Provider Name</th>,
 <th>Complete Address</th>,
 <th>City</th>,
 <th>Province</th>,
 <th>Region</th>,
 <th>Area</th>,
 <th>Contact No.</th>]
```

In [62]:
```
hospital_header = ['Provider Name', 'Complete Address', 'Contact No']
print(hospital_header)
```

```
['Provider Name', 'Complete Address', 'Contact No']
```

In [63]:
```
hospital_header = hospitals_table.find_all('tr')
```

In [52]:
```
# Save to csv
hospitals.to_csv('hospitals.csv')
```

Conclusion

On this activity, I've had a hard time identifying the right html tags to look for in order to scrape the data from the website. On the other hand, I spend a lot of time looking for a website to scrape initially. I think there's a better way to do this rather than manually searching contents under its html elements, maybe a better site has already prepared the data for web scraping. Overall, it's a challenging activity due to BeautifulSoup being new to me.