BigData exercises – Week 2

This week, you will start web crawling with bespoke modules/libraries in Python and parsing the crawled data into structured frames. Then the visualization tools will be used in comparison with singly numerical analysis results to gain an insight into its use.

Still, if you are experienced with the web crawling and visualization details, this week you are encouraged to explore more libraries available on Anaconda. DON'T limit yourself to the unit materials.

When you are building your own project, always refer to the bespoke built-in modules and read their documentation first, which will be helpful!!!

Exercise 1:

In this exercise, you will need to crawl the web data with basic operations provided by urllib, requests, scrapy.

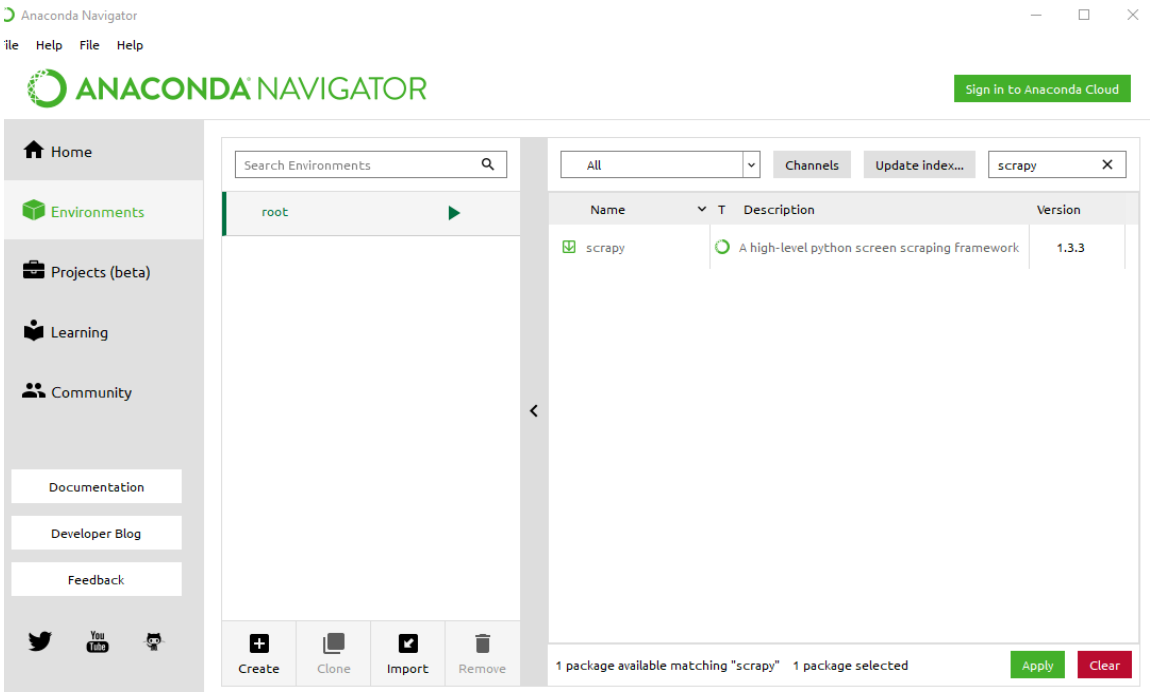
Please (always remember to) refer to official documentation for lib and function details.

urllib (https://docs.python.org/3/library/urllib.html)

requests (https://requests.readthedocs.io/en/master/)

scrapy (https://docs.scrapy.org/en/latest/)

- Check if the libraries can be imported. If not, install them in the Anaconda environment. (Note that the libs in a uni PC are not always the latest version)



```
=========================================================================

#urllib

import urllib

#headers are not included here

urr = urllib.request.urlopen('https://www.port.ac.uk/')

content = urr.read()

urr.close()

html = content.decode()

print(html)

=========================================================================


=========================================================================

#requets.get

import requests

rer = requests.get('https://www.port.ac.uk/')

print(rer.status_code)

html = rer.text

print(html)

=========================================================================
```

- **(Optional)** For Scrapy framework, please refer to the official documentation for crawling details.

Exercise 2:

In this exercise, you will refer to the example from the official documentation to parse the sample html data with beautifulsoup and extract the information you need with regular expression re.

re (https://docs.python.org/3/library/re.html)

beautifulsoup (https://www.crummy.com/software/BeautifulSoup/bs4/doc)

```
==========================================================================

html_doc = """
      <html><head><title>The Dormouse's story</title></head>
      <body>
      <p class="title"><b>The Dormouse's story</b></p>
      <p class="story">Once upon a time there were three little sisters; and their names were
      <a href="http://example.com/elsie" class="sister" id="link1">Elsie</a>,
      <a href="http://example.com/lacie" class="sister" id="link2">Lacie</a> and
      <a href="http://example.com/tillie" class="sister" id="link3">Tillie</a>;
      and they lived at the bottom of a well.</p>
      <p class="story">...</p>
"""
from bs4 import BeautifulSoup as bs
import re
soup = bs(html_doc, 'lxml')
print(soup.prettify())
for link in soup.find_all('a'):
      print(link.get('href'))
print(soup.get_text())
for tag in soup.find_all(re.compile("^b")):
       print(tag.name)
==========================================================================
```

Exercise 3:

Using bespoke libs to get the current weather of Portsmouth through API.

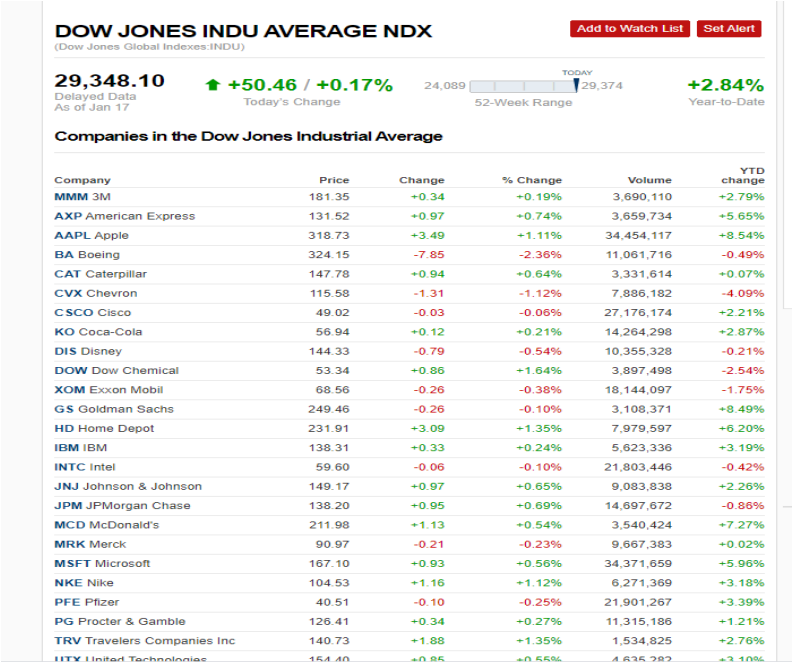
- The API description page link is https://openweathermap.org/current
- You may need to register for a free API key and include it in the request.

json https://docs.python.org/3/library/json.html

Exercise 4:

Using bespoke libs to crawl the Dow Jones Global Index from CNN and output the code, name and price of the 30 companies in a list.

- The page link is https://money.cnn.com/data/dow30/



DOW JONES INDU AVERAGE NDX

(Dow Jones Global Indexes:INDU)

Add to Watch List | Set Alert

29,348.10 — Delayed Data As of Jan 17

+50.46 / +0.17% — Today's Change

24,089 — TODAY 29,374 — 52-Week Range

+2.84% — Year-to-Date

Companies in the Dow Jones Industrial Average

Company	Price	Change	% Change	Volume	YTD change
MMM 3M	181.35	+0.34	+0.19%	3,690,110	+2.79%
AXP American Express	131.52	+0.97	+0.74%	3,659,734	+5.65%
AAPL Apple	318.73	+3.49	+1.11%	34,454,117	+8.54%
BA Boeing	324.15	-7.85	-2.36%	11,061,716	-0.49%
CAT Caterpillar	147.78	+0.94	+0.64%	3,331,614	+0.07%
CVX Chevron	115.58	-1.31	-1.12%	7,886,182	-4.09%
CSCO Cisco	49.02	-0.03	-0.06%	27,176,174	+2.21%
KO Coca-Cola	56.94	+0.12	+0.21%	14,264,298	+2.87%
DIS Disney	144.33	-0.79	-0.54%	10,355,328	-0.21%
DOW Dow Chemical	53.34	+0.86	+1.64%	3,897,498	-2.54%
XOM Exxon Mobil	68.56	-0.26	-0.38%	18,144,097	-1.75%
GS Goldman Sachs	249.46	-0.26	-0.10%	3,108,371	+8.49%
HD Home Depot	231.91	+3.09	+1.35%	7,979,597	+6.20%
IBM IBM	138.31	+0.33	+0.24%	5,623,336	+3.19%
INTC Intel	59.60	-0.06	-0.10%	21,803,446	-0.42%
JNJ Johnson & Johnson	149.17	+0.97	+0.65%	9,083,838	+2.26%
JPM JPMorgan Chase	138.20	+0.95	+0.69%	14,697,672	-0.86%
MCD McDonald's	211.98	+1.13	+0.54%	3,540,424	+7.27%
MRK Merck	90.97	-0.21	-0.23%	9,667,383	+0.02%
MSFT Microsoft	167.10	+0.93	+0.56%	34,371,659	+5.96%
NKE Nike	104.53	+1.16	+1.12%	6,271,369	+3.18%
PFE Pfizer	40.51	-0.10	-0.25%	21,901,267	+3.39%
PG Procter & Gamble	126.41	+0.34	+0.27%	11,315,186	+1.21%
TRV Travelers Companies Inc	140.73	+1.88	+1.35%	1,534,825	+2.76%
UTX United Technologies	154.40	+0.86	+0.55%	4,635,282	+3.10%

- If you are using requests lib, you will need to use requests.get()
- If you are using re lib, you probably need to use re.compile() and re.findall().

The output list looks like

```
[('MMM', '3M', '181.35'), ('AXP', 'American Express', '131.52'), ('AAPL', 'Apple', '318.73'), ('BA', 'Boeing', '324.15'),
('CAT', 'Caterpillar', '147.78'), ('CVX', 'Chevron', '115.58'), ('CSCO', 'Cisco', '49.02'), ('KO', 'Coca-Cola', '56.94'),
('DIS', 'Disney', '144.33'), ('DOW', 'Dow Chemical', '53.34'), ('XOM', 'Exxon Mobil', '68.56'), ('GS', 'Goldman Sachs',
'249.46'), ('HD', 'Home Depot', '231.91'), ('IBM', 'IBM', '138.31'), ('INTC', 'Intel', '59.60'), ('JNJ', 'Johnson & Johnson',
'149.17'), ('JPM', 'JPMorgan Chase', '138.20'), ('MCD', "McDonald's", '211.98'), ('MRK', 'Merck', '90.97'), ('MSFT',
'Microsoft', '167.10'), ('NKE', 'Nike', '104.53'), ('PFE', 'Pfizer', '40.51'), ('PG', 'Procter & Gamble', '126.41'), ('TRV',
'Travelers Companies Inc', '140.73'), ('UTX', 'United Technologies', '154.40'), ('UNH', 'UnitedHealth', '298.47'), ('VZ',
'Verizon', '60.13'), ('V', 'Visa', '204.70'), ('WMT', 'Wal-Mart', '114.96'), ('WBA', 'Walgreen', '54.41')]
```

(Optional) Exercise 5:

Read the html file of https://www.volleyball.world/en/vnl/2018/women/results-and-ranking/round1 first. Then crawl the information of Nation and Total_No, Won_No, Lost_No. and output them in a list.

- Still, if you are using requests lib, you will need to use requests.get().

- And If you are also using re lib, you probably need to use re.compile() and re.findall().

ROUND ROBIN

Rankings | Results

RANK	TEAMS	MATCHES TOTAL	MATCHES WON	MATCHES LOST	RESULT DETAILS 3-0	3-1	3-2	2-3	1-3	0-3	POINTS	SETS WON	SETS LOST	SETS RATIO	POINTS WON	POINTS LOST	POINTS RATIO
1	USA	15	13	2	11	2	0	1	1	0	40	42	8	5.250	1227	997	1.230
2	SERBIA	15	12	3	7	4	1	2	1	0	37	41	15	2.733	1324	1141	1.160
3	BRAZIL	15	12	3	3	7	2	1	2	0	35	40	20	2.000	1376	1229	1.119
4	NETHERLANDS	15	12	3	6	3	3	1	1	1	34	39	18	2.166	1327	1176	1.128
5	TURKEY	15	11	4	5	5	1	3	1	0	35	40	19	2.105	1351	1244	1.086
6	ITALY	15	10	5	6	1	3	2	0	3	29	34	22	1.545	1230	1136	1.082
7	RUSSIA	15	8	7	2	4	2	1	0	6	23	26	29	0.896	1194	1198	0.996
8	POLAND	15	8	7	3	2	3	1	3	3	22	29	29	1.000	1298	1211	1.071
9	CHINA	15	7	8	5	1	1	2	4	2	22	29	27	1.074	1237	1178	1.050

The output list looks like

```
[('USA', '15', '13', '2'), ('Serbia', '15', '12', '3'), ('Brazil', '15', '12', '3'), ('Netherlands', '15', '12', '3'),
('Turkey', '15', '11', '4'), ('Italy', '15', '10', '5'), ('Russia', '15', '8', '7'), ('Poland', '15', '8', '7'), ('China',
'15', '7', '8'), ('Japan', '15', '7', '8'), ('Germany', '15', '5', '10'), ('Korea', '15', '5', '10'), ('Belgium', '15', '4',
'11'), ('Dominican Republic', '15', '3', '12'), ('Thailand', '15', '2', '13'), ('Argentina', '15', '1', '14')]
```

So you have successfully implemented the basic data collection step. Now we go on with the visualization tools.

Exercise 6:

Please import from seaborn the famous Anscombe's quartet. Then plot them with matplot. And calculate their means, variances correlations and linear fitting coefficients. For linear regression, you can use the sklearn lib. Can you have a more concise way to plot the data?

matplotlib (https://matplotlib.org/)

seaborn (https://seaborn.pydata.org/)

sklearn (https://scikit-learn.org/stable/)

==========

```
import seaborn as sns

import matplotlib.pyplot as plt

from sklearn import linear_model

anscombe = sns.load_dataset("anscombe")
```

```
print(anscombe)

# create subsets and subplots of the anscombe data

dataset_1 = anscombe[anscombe['dataset'] == 'I']

dataset_2= anscombe[anscombe['dataset'] == 'II']

dataset_3 = anscombe[anscombe['dataset'] == 'III']

dataset_4 = anscombe[anscombe['dataset'] == 'IV']

fig = plt.figure()

axes1 = fig.add_subplot(2, 2, 1)

axes2 = fig.add_subplot(2, 2, 2)

axes3 = fig.add_subplot(2, 2, 3)

axes4 = fig.add_subplot(2, 2, 4)

axes1.plot(dataset_1['x'], dataset_1['y'], 'o')

axes2.plot(dataset_2['x'], dataset_2['y'], 'o')

axes3.plot(dataset_3['x'], dataset_3['y'], 'o')

axes4.plot(dataset_4['x'], dataset_4['y'], 'o')

#linear regression model

regr = linear_model.LinearRegression()

regr.fit(dataset_1['x'].values.reshape(-1,1), dataset_1['y'].values.reshape(-1,1))

axes1.plot(dataset_1['x'].values.reshape(-1,1), regr.predict(dataset_1['x'].values.reshape(-1,1)), 'r')
```

==========

(Optional) Exercise 7:

Consider the problem of Boston Dataset you have seen in Week1, visualize the analysis results to give a more intuitive understanding. (Plot 'y_predict' and 'y' from 'bostonPythonRefresh.py' that you have executed in the same figure).

In the example, you were given the model of 13-dimension input and 1-dimension output. Can you put them all in a 2-d or 3-d plot?) If not, can you think of an alternative?

(A plausible and tricky solution: reduce the dimensionality. We will cover it next week, but you can first refer* to https://scikit-learn.org/stable/modules/decomposition.html#pca)

***Always remember to refer to the official documentation of libraries, particularly the data science tools/models. They are well established tools, and detailed explanations are available in the library.**