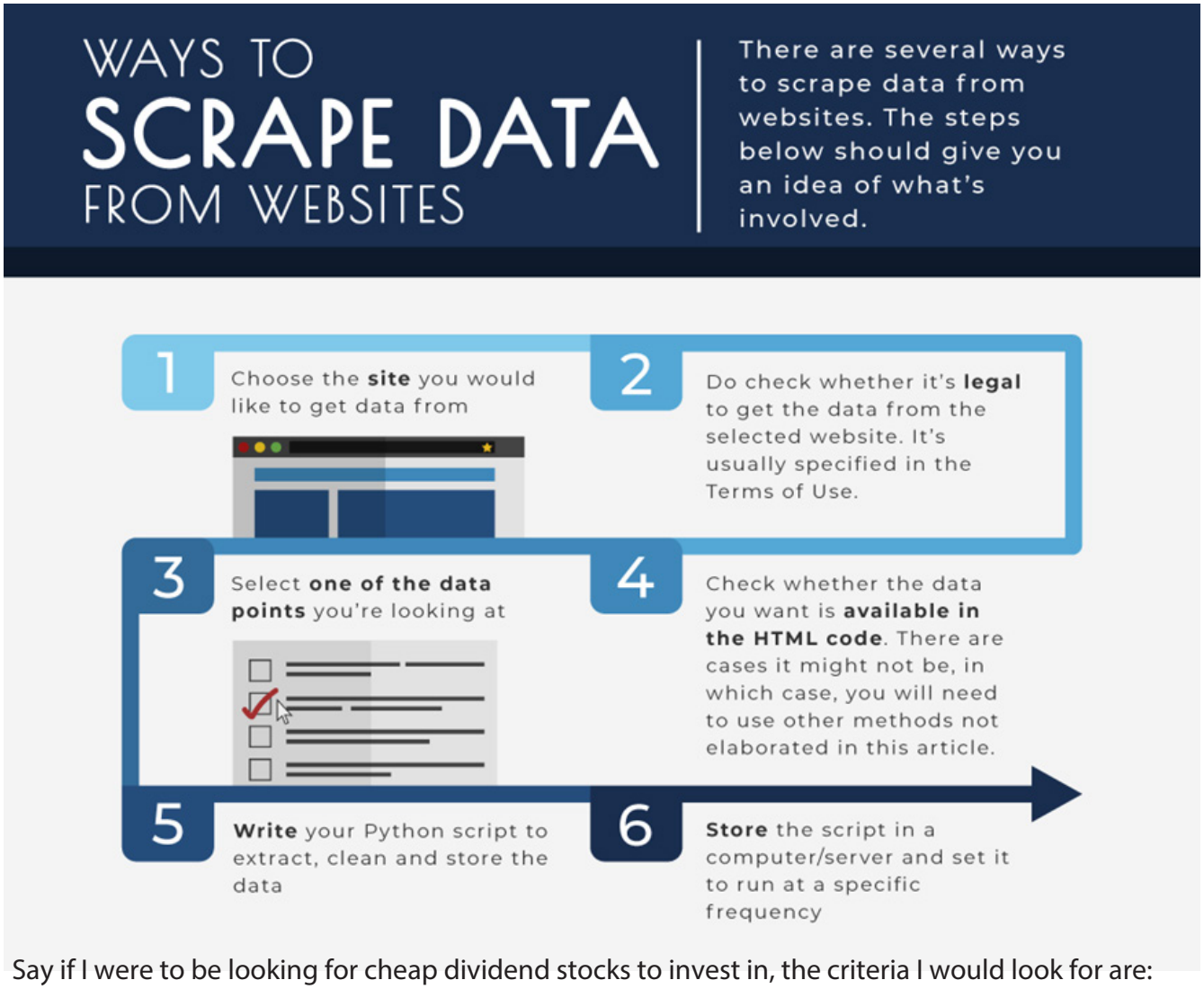
ASIAN BANKING SCHOOL

Let's Talk Digital Series #9

Applications and a Guide to Web Scraping

This article is about making use of external data to complement your organisation's data sources by web-scraping. Most organisations like financial institutions have rich datasets on their users: they are able to identify who are the large spenders, who are likely to register for exclusive credit cards, who are likely to go into default when given a loan etc. These organisations are not likely to have data to answer questions like what customers think of their products and services, which companies are likely to perform well in the stock market in the near future, and how live updates on critical events can directly impact businesses. The answers to those questions lie scattered all over the web on sites like Lowyat.NET, Reddit, The Star Online, The Edge Markets and so on. Most of the data needed can be obtained by using automated systems to harvest data from selected sites.



Say if I were to be looking for cheap dividend stocks to invest in, the criteria I would look for are:

1) low or moderate Price-to-Earnings (PE)
2) high Dividend Yield (DE)
3) high Return on Equity (ROE)

Bursa Malaysia's Equities Prices[1] page doesn't have this information, but I did find another site which has on malaysiastock.biz[2]. The ratios I'm looking for are available, but the stocks are listed alphabetically, so I would not be able to see the ratios across the entire stock market. This is where web scraping can help, and the steps below show how to accomplish this.

1) Import the relevant Python libraries

```
import pandas as pd
import requests
import urllib.request
import time
from bs4 import BeautifulSoup ##other options are scrapy and Selenium
```

2) Scrape malaysiastock.biz

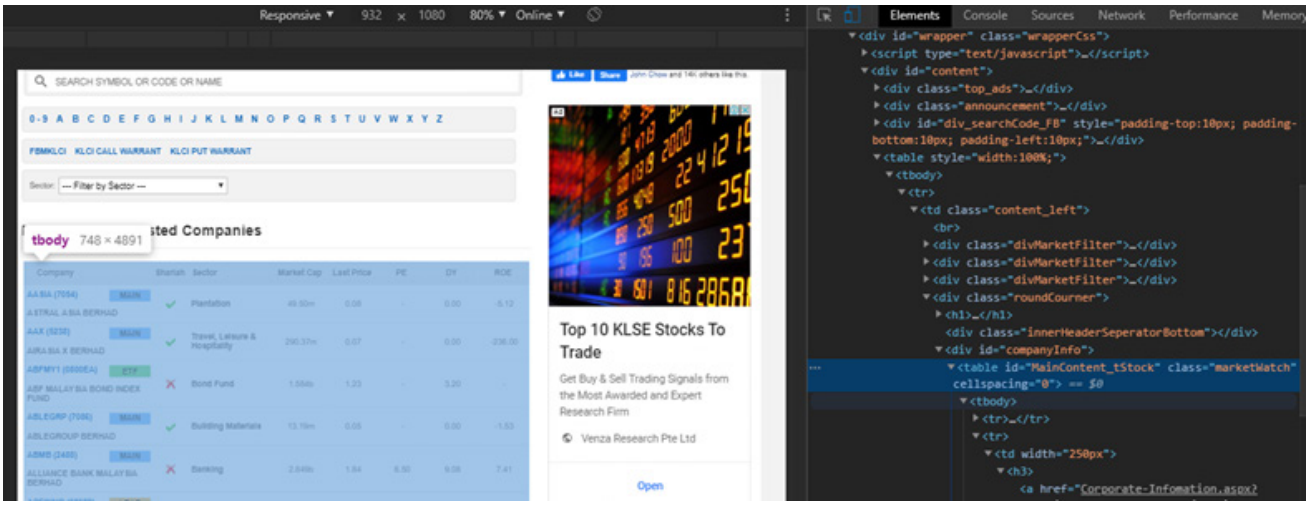
[1] https://www.bursamalaysia.com/market_information/equities_prices
[2] https://www.malaysiastock.biz/Listed-Companies.aspx?type=A&value=A

```
url = 'https://www.malaysiastock.biz/Listed-
Companies.aspx?type=A&value=A' ##scrape only stocks starting with A
response = requests.get(url)
response
```

```
<Response [200]> ##this confirms the connection is successful
```

```
soup = BeautifulSoup(response.text, "html.parser")
##now use BeautifulSoup's html.parser to display the data
table = soup.findAll('table', {'class': 'marketWatch'})
##and look for a table called marketWatch
table
```

Using Chrome, if you were to right click on **AASIA (7054)**, and click **Inspect**, you will be able to see the html code of the webpage, as snapshot below. Since the data on the website is displayed in a table, it would be reasonable to look for a table within the html code. While parsing the html code, notice the table on the webpage is highlighted when you hover your cursor over **<table id="MainContent_tStock" class="marketWatch" cellspacing="0">**.



3) The next part is the pre-processing bit. This is usually deemed the most laborious part of this process. Most data which comes from a html.parser is usually in html format, and it not easily converted into a neat table.

```
example = list(table[0])  ##converts table into a list
example2 = [x for x in example if x != "\n"]  ##get rid of empty lines

prices = []  ##creates an empty array to store data

for i in range(1, len(example2)):
entry = example2[i].get_text(separator="\n").split("\n")
entry_filtered = [x for x in entry if x]
prices.append(entry_filtered)

prices
```

The for loop is for me to separate the data using the delimiter \n , get rid of any more delimiters \n, and to append the result in the prices array. A snapshot of the prices array is to the right.

```
[['AASIA (7054)',
  'MAIN',
  'ASTRAL ASIA BERHAD',
  'Plantation ',
  '49.50m',
  '0.08',
  '-',
  '0.00',
  '-5.12'],
 ['AAX (5238)',
  'MAIN',
  'AIRASIA X BERHAD',
```

rid of away.

4) Once your data is in an array, it's easy to convert it into a table and to rename the columns. A snapshot of the resulting table is below.

```
complete_table = pd.DataFrame(prices)
complete_table.rename(columns = {0: "Company Code", 1: "Market", 2:
"Company Name", 3: "Sector", 4: "Market Cap", 5: "Last Price",
                                 6: "PE", 7: "DY", 8: "ROE"})
```

	Company Code	Market	Company Name	Sector	Market Cap	Last Price	PE	DY	ROE
0	AASIA (7054)	MAIN	ASTRAL ASIA BERHAD	Plantation	49.50m	0.08	-	0.00	-5.12
1	AAX (5238)	MAIN	AIRASIA X BERHAD	Travel, Leisure & Hospitality	290.37m	0.07	-	0.00	-236.00
2	ABFMY1 (0800EA)	ETF	ABF MALAYSIA BOND INDEX FUND	Bond Fund	1.584b	1.23	-	3.20	-
3	ABLEGRP (7086)	MAIN	ABLEGROUP BERHAD	Building Materials	13.19m	0.05	-	0.00	-1.53
4	ABMB (2488)	MAIN	ALLIANCE BANK MALAYSIA BERHAD	Banking	2.849b	1.84	6.50	9.08	7.41
...	...	...	...	...	...	...	...	...	...
74	AXIATA (6888)	MAIN	AXIATA GROUP BERHAD	Telecommunications Service Providers	29.325b	3.20	21.48	2.97	8.42
75	AXREIT (5106)	MAIN	AXIS REAL ESTATE INVESTMENT TRUST	Real Estate Investment Trusts	2.596b	1.80	12.41	5.14	10.02
76	AYER (2305)	MAIN	AYER HOLDINGS BERHAD	Property	415.43m	5.55	31.20	0.90	2.55
77	AYS (5021)	MAIN	AYS VENTURES BERHAD	Building Materials	60.87m	0.16	-	6.25	-2.75
78	AZRB (7078)	MAIN	AHMAD ZAKI RESOURCES BERHAD	Construction	101.65m	0.17	-	5.88	-

79 rows × 9 columns

5) The resulting table only contains the statistics of stocks starting with A. The remaining tasks are to construct a for loop, to run this algorithm for stocks starting with B, C, D, and so on. This can be achieved by running the algorithm above and changing the url variable to: `'https://www.malaysiastock.biz/Listed-Companies.aspx?type=A&value=B'` for stocks starting with B, and doing the same for other alphabets. The full code can be accessed via the QR on the right or the link below[3].

SCAN ME

To deploy this code:
1) Adjust the output directory of the code to point to a data warehouse
2) Store this script in a Docker container, and set the cron job to run it every 15 minutes

Web scraping is useful for other sorts of data too:

1) To gauge what customers think of your organisation's products and services based on comments from public forums like Lowyat.NET and Reddit using sentiment analysis
2) To find out what topics are trending on Twitter and Instagram for product development using topic modelling
3) To gather information from news sites which may impact your organisation's products and services like news coverage about competitor's products

One important takeaway is to identify the problem your business is trying to solve. If it involves web scraping, it's vital to understand the structure of the website on which you would like to scrape the information. Scraping different sites will involve multiple pre-processing methods, which means someone in your team will need to maintain the scraper if the target site's structure changes.

[3]https://github.com/atlas-github/malaysiastockbiz_scraper/blob/master/malaysiastock_biz_scraper.ipynb

ASIAN BANKING SCHOOL

This article is part of the Digital Banking Learning Series, 'Let's Talk Digital', an initiative by the ABS Center for Digital Banking. It is written by industry practitioners and are aimed at educating the general public on the intricacies of digital applications in banking and other related industries, including the latest insights and trends of Digital Banking.

As the industry's preferred partner in learning and development, ABS offers relevant training programmes that covers a comprehensive list of banking areas that are designed and developed in-house by our Specialist Training Consultancy Team or in collaboration with strategic learning partners that includes some of the top business schools in the world. It also provides specialised consulting services and tailored learning solutions to meet the specific needs of its clients.

For more information, visit our website at **www.asianbankingschool.com** or email us at **digitalbanking@asianbankingschool.com**