プログラミング体験(Python)
WEBスクレイピング

WEBスクレイピングとは

- 「Scrape（こする・削る・かき出す）」が由来
- Webサイトから特定の情報を自動的に抽出するコンピュータソフトウェア技術のこと
- Webスクレイピングを使えば、インターネット上に存在するWebサイトを探り、大量のデータの中から余分なデータを削ぎ落とし、特定のデータのみを抽出することが可能

WEBスクレイピングする際の注意点

- 法律に触れる可能性がある
  ⬇
  対象のサイトの利用規約を遵守したり、
  負荷をかけない工夫が必要

- Webページの構造変化への対応が必要

WEBスクレイピングのための Pythonライブラリ・フレームワーク例

- requests + Beautiful Soup
- Selenium
- Scrapy

⋮

WEBスクレイピングの手順

スクレイピングするサイトのHTMLを検証モードで表示

⬇

スクレイピングしたい箇所の構造を確認

⬇

その箇所の要素を抽出するようにプログラミング

⬇

プログラムを実行し必要な情報をスクレイピング

実際にスクレイピングしてみよう

Beautiful Soupの基本的なメソッド

- select("要素")
  指定した要素に当てはまる部分を全て取得

- select_one("要素")
  指定した要素に当てはまる部分の最初の1つを取得

- get_text()
  直前の要素のテキストを取得

- attrs["属性"]
  直前の要素の指定した属性(href, title等)を取得

プログラミングに必要なライブラリをインポート

```
1    # 必要なライブラリをインポート
2    import requests
3    from bs4 import BeautifulSoup
4    import csv
```

- **requests**
  サイトからHTML情報を取得するのに使用

- **BeautifulSoup**
  HTML情報から必要なデータを抽出するのに使用

- **csv**
  CSVファイルの読み書きに使用

サイトのHTML情報を取得

```
6   # スクレイピング先を選択するフラグ  True : オフラインのサイト  False : オンラインのサイト
7   is_offline = True
8
9   # サイトから情報をスクレイピング
10  if is_offline:
11      soup = BeautifulSoup(open('./sample_site/yahoo_finance_dividend_yield_ranking.html', encoding='utf-8'), "html.parser")
12  else:
13      url = 'https://finance.yahoo.co.jp/stocks/ranking/dividendYield'
14      response = requests.get(url)
15      soup = BeautifulSoup(response.content, "html.parser")
16
17  # 1. 変数soupの確認 確認後下2行をコメントアウト
18  print(soup)
19  exit()
```

確認1 プログラムを実行して変数soupを中身を確認
確認できたら18,19行目をコメントアウト(ctrl + /)

```
17  # 1. 変数soupの確認 確認後下2行をコメントアウト
18  # print(soup)
19  # exit()
```

検証モードでサイトのHTML情報を確認

株式1つ分の情報が<tr class="_1GwpkGwB">の中身
selectメソッドを使ってこの塊を取得

株式情報を取得する

```
21    # 高配当株式の行のデータを取得
22    stock_rows = soup.select('スクレイピングするクラス')
23
24    # 2. 変数stock_rowsの確認 確認後下2行をコメントアウト
25    print(stock_rows)
26    exit()
```

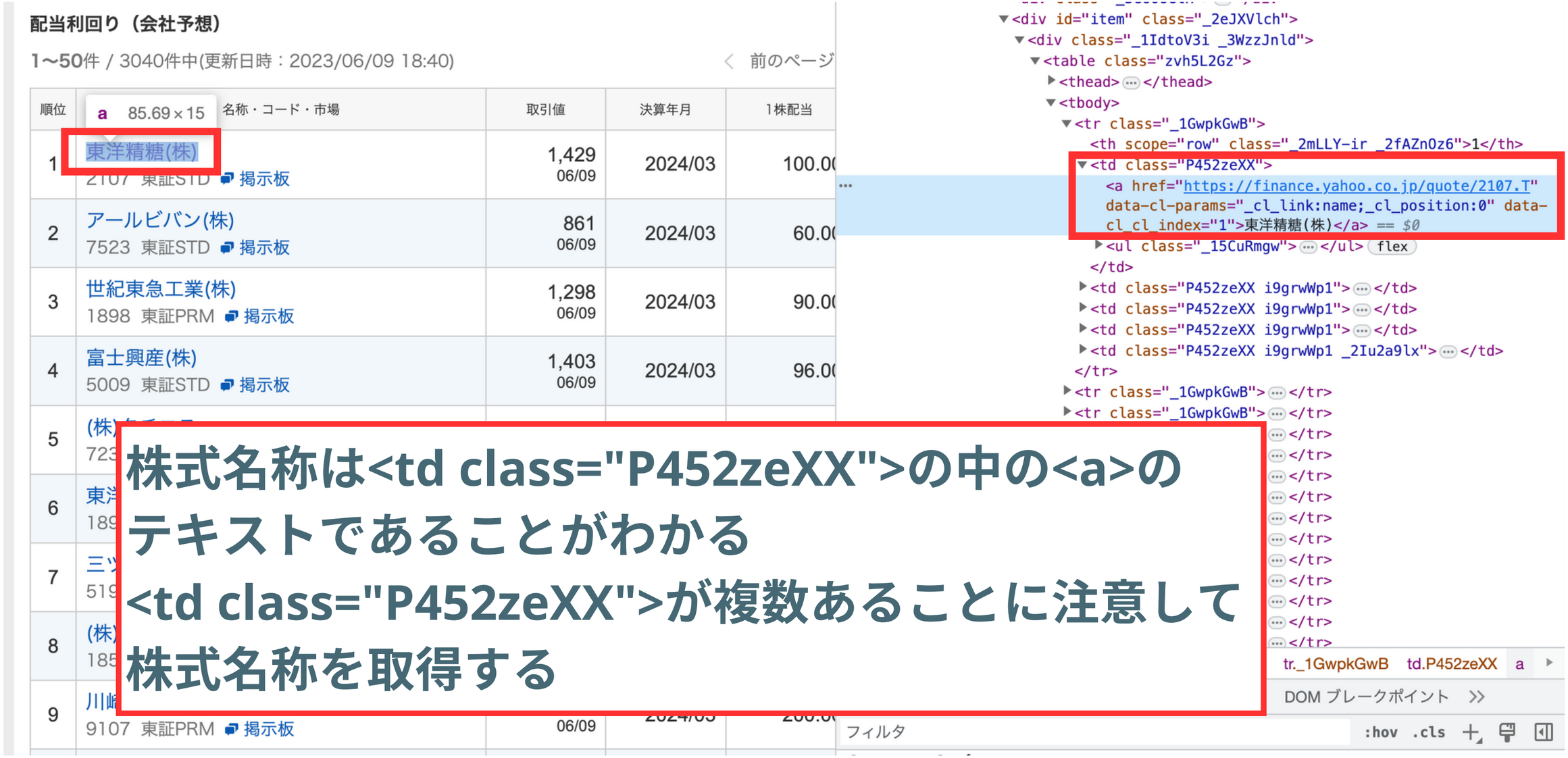
classが"_1GwpkGwB"の要素を取得したいので、22行目を以下のように埋める(classは"."、idは"#"を名前の前につける)

```
21    # 高配当株式の行のデータを取得
22    stock_rows = soup.select('._1GwpkGwB')
```

確認2 プログラムを実行して変数stock_rowsを中身を確認 確認できたら25,26行目はコメントアウト

```
24    # 2. 変数stock_rowsの確認 確認後下2行をコメントアウト
25    # print(stock_rows)
26    # exit()
```

検証モードでサイトの株式名称情報の構造を確認



株式名称情報を取得

```
33      # 株式名称をスクレイピング
34      stock_name = stock_row.select('スクレイピングするクラス')[0].select_one('スクレイピングするタグ').get_text()
35
36      # 3. 変数stock_nameの確認 確認後下2行をコメントアウト
37      print(stock_name)
38      exit()
```

classが"P452zeXX"の要素の1番目を抽出したいので、34行目を以下のように埋める

```
33      # 株式名称をスクレイピング
34      stock_name = stock_row.select('P452zeXX')[0].select_one('スクレイピングするタグ').get_text()
```

aタグのテキストを取得したいので、34行目を以下のように埋める

```
33      # 株式名称をスクレイピング
34      stock_name = stock_row.select('P452zeXX')[0].select_one('a').get_text()
```

確認3 プログラムを実行して変数stock_nameを中身を確認
出力が確認できたら37,38行目はコメントアウト

その他の株式情報を取得

同じように41,44,47行目を穴埋めして以下を取得してみよう

- 株式価格
- 1株当たり配当金
- 配当利回り

```
40      # 株式価格をスクレイピング
41      stock_price = stock_row.select('スクレイピングするクラス')[1].select_one('スクレイピングするクラス').get_text()
42
43      # 1株当たり配当金をスクレイピング
44      dividend_per_share = stock_row.select('スクレイピングするクラス')[3].select_one('スクレイピングするクラス').get_text()
45
46      # 配当利回りをスクレイピング
47      dividend_yield = stock_row.select('スクレイピングするクラス')[4].select_one('スクレイピングするクラス').get_text()
```

全て穴埋めしてコードを実行すると、スクレイピング結果をターミナルに出力するとともに、CSVファイルに保存します

コードを自由に書き換えて好きな所をスクレイピングしてみよう

例えば...

- 順位
- 決算年月
- 株式コード
- 株式会社のリンク
- 「次へ」のリンク

まとめ

- スクレイピングを使うことでWEBサイトの
  必要な情報のみを取得可能

- スクレイピングする際は法律等に触れないように注意

- 対象サイトで検証モードを使い、抽出する要素を
  確認しながらプログラムを実装