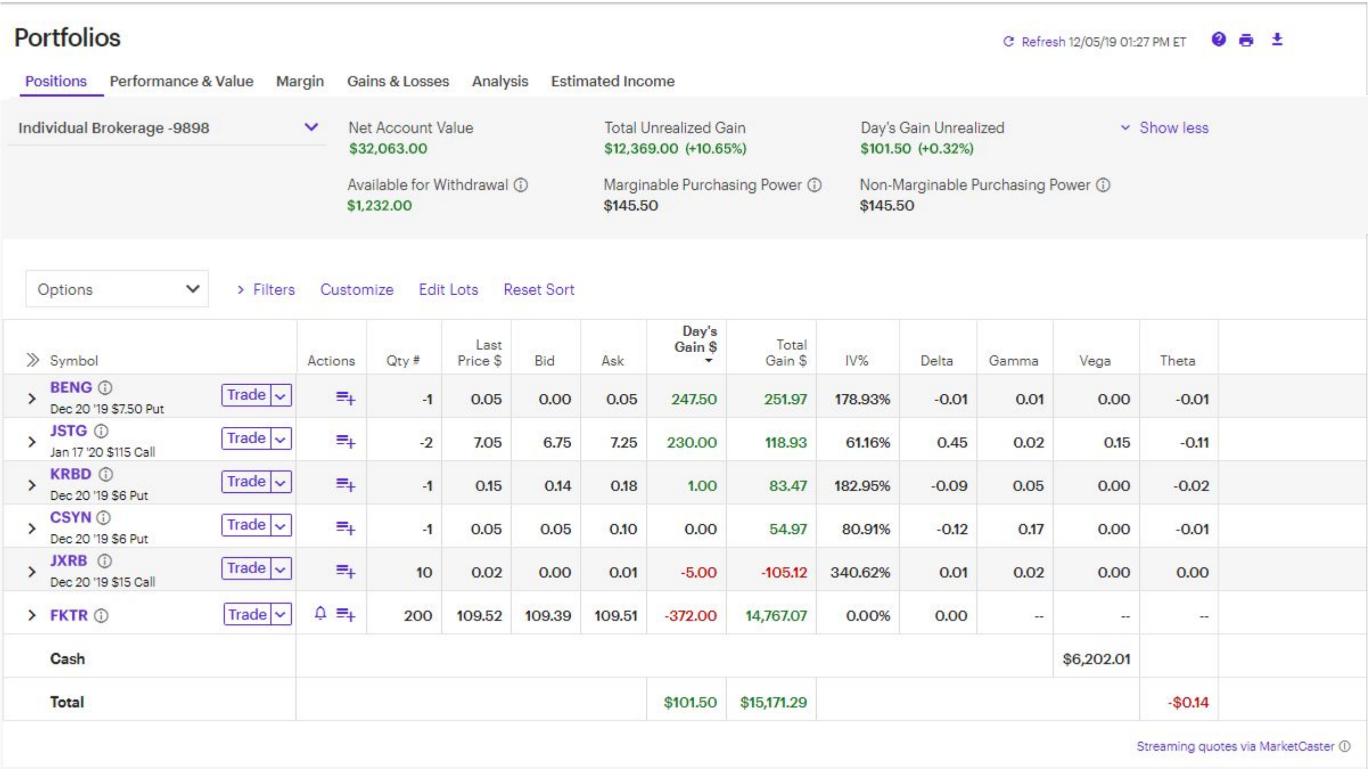
Web Scraping:
Beautiful Soup vs Selenium

Jack Chen
William Frame
Samuel Perebikovsky

Problems Revisited: Yahoo Finance Scraping



Individual Brokerage -9898

Net Account Value
$32,063.00

Total Unrealized Gain
$12,369.00 (+10.65%)

Day's Gain Unrealized
$101.50 (+0.32%)

Available for Withdrawal
$1,232.00

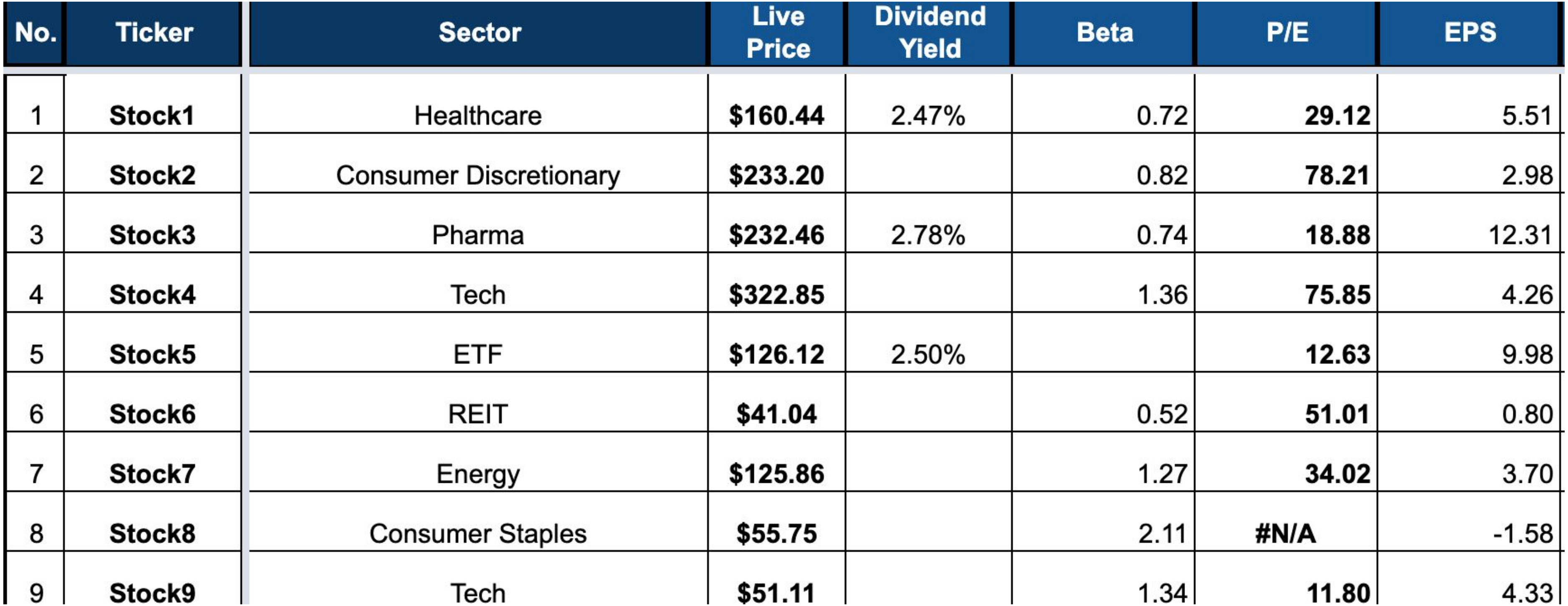
Marginable Purchasing Power
$145.50

Non-Marginable Purchasing Power
$145.50

Symbol	Actions	Qty #	Last Price $	Bid	Ask	Day's Gain $	Total Gain $	IV%	Delta	Gamma	Vega	Theta
BENG Dec 20 '19 $7.50 Put Trade		-1	0.05	0.00	0.05	247.50	251.97	178.93%	-0.01	0.01	0.00	-0.01
JSTG Jan 17 '20 $115 Call Trade		-2	7.05	6.75	7.25	230.00	118.93	61.16%	0.45	0.02	0.15	-0.11
KRBD Dec 20 '19 $6 Put Trade		-1	0.15	0.14	0.18	1.00	83.47	182.95%	-0.09	0.05	0.00	-0.02
CSYN Dec 20 '19 $6 Put Trade		-1	0.05	0.05	0.10	0.00	54.97	80.91%	-0.12	0.17	0.00	-0.01
JXRB Dec 20 '19 $15 Call Trade		10	0.02	0.00	0.01	-5.00	-105.12	340.62%	0.01	0.02	0.00	0.00
FKTR Trade		200	109.52	109.39	109.51	-372.00	14,767.07	0.00%	0.00	--	--	--
Cash											$6,202.01	
Total						$101.50	$15,171.29					-$0.14

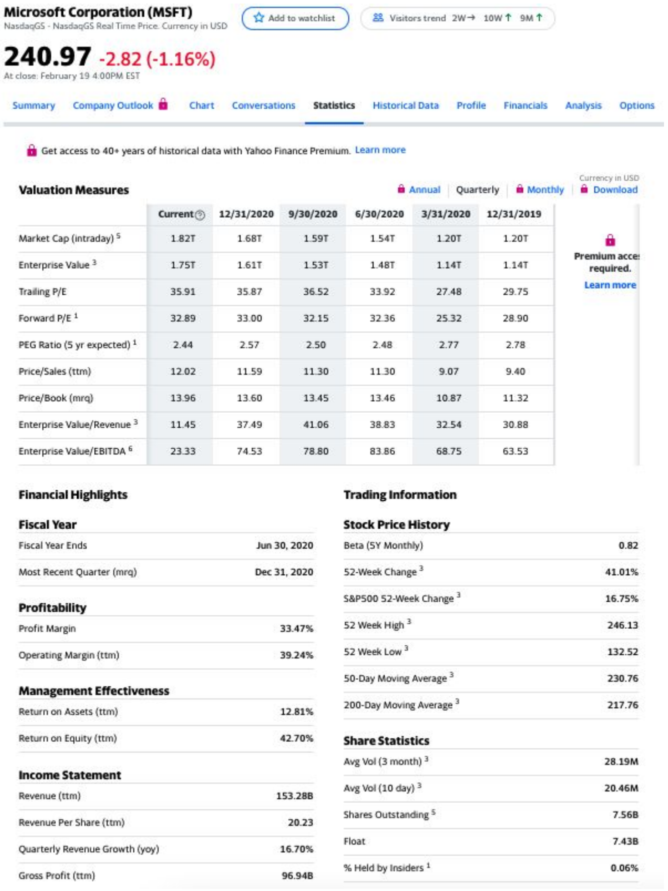
Streaming quotes via MarketCaster

Problems Revisited: Yahoo Finance Scraping



No.	Ticker	Sector	Live Price	Dividend Yield	Beta	P/E	EPS
1	**Stock1**	Healthcare	**$160.44**	2.47%	0.72	**29.12**	5.51
2	**Stock2**	Consumer Discretionary	**$233.20**		0.82	**78.21**	2.98
3	**Stock3**	Pharma	**$232.46**	2.78%	0.74	**18.88**	12.31
4	**Stock4**	Tech	**$322.85**		1.36	**75.85**	4.26
5	**Stock5**	ETF	**$126.12**	2.50%		**12.63**	9.98
6	**Stock6**	REIT	**$41.04**		0.52	**51.01**	0.80
7	**Stock7**	Energy	**$125.86**		1.27	**34.02**	3.70
8	**Stock8**	Consumer Staples	**$55.75**		2.11	**#N/A**	-1.58
9	**Stock9**	Tech	**$51.11**		1.34	**11.80**	4.33

Problems Revisited: Yahoo Finance Scraping



Microsoft Corporation (MSFT)

NasdaqGS - NasdaqGS Real Time Price. Currency in USD

240.97 -2.82 (-1.16%)

At close: February 19 4:00PM EST

Get access to 40+ years of historical data with Yahoo Finance Premium. Learn more

Currency in USD

Valuation Measures

	Current	12/31/2020	9/30/2020	6/30/2020	3/31/2020	12/31/2019
Market Cap (intraday)[5]	1.82T	1.68T	1.59T	1.54T	1.20T	1.20T
Enterprise Value[3]	1.75T	1.61T	1.53T	1.48T	1.14T	1.14T
Trailing P/E	35.91	35.87	36.52	33.92	27.48	29.75
Forward P/E[1]	32.89	33.00	32.15	32.36	25.32	28.90
PEG Ratio (5 yr expected)[1]	2.44	2.57	2.50	2.48	2.77	2.78
Price/Sales (ttm)	12.02	11.59	11.30	11.30	9.07	9.40
Price/Book (mrq)	13.96	13.60	13.45	13.46	10.87	11.32
Enterprise Value/Revenue[3]	11.45	37.49	41.06	38.83	32.54	30.88
Enterprise Value/EBITDA[6]	23.33	74.53	78.80	83.86	68.75	63.53

Premium acce
required.
Learn more

Financial Highlights

Fiscal Year

Fiscal Year Ends	Jun 30, 2020
Most Recent Quarter (mrq)	Dec 31, 2020

Profitability

Profit Margin	33.47%
Operating Margin (ttm)	39.24%

Management Effectiveness

Return on Assets (ttm)	12.81%
Return on Equity (ttm)	42.70%

Income Statement

Revenue (ttm)	153.28B
Revenue Per Share (ttm)	20.23
Quarterly Revenue Growth (yoy)	16.70%
Gross Profit (ttm)	96.94B

Trading Information

Stock Price History

Beta (5Y Monthly)	0.82
52-Week Change[3]	41.01%
S&P500 52-Week Change[3]	16.75%
52 Week High[3]	246.13
52 Week Low[3]	132.52
50-Day Moving Average[3]	230.76
200-Day Moving Average[3]	217.76

Share Statistics

Avg Vol (3 month)[3]	28.19M
Avg Vol (10 day)[3]	20.46M
Shares Outstanding[5]	7.56B
Float	7.43B
% Held by Insiders[1]	0.06%

What is Beautiful Soup?

- A module for pulling data out of HTML and XML documents
- Pros:
  - Beginner Friendly, very good documentation and a friendly user community
  - Most popular web scraping Python library
- Drawbacks:
  - Relies on other libraries to work (requesters and parser) → More dependencies
  - Lack of JavaScript handling by itself

What is Selenium?

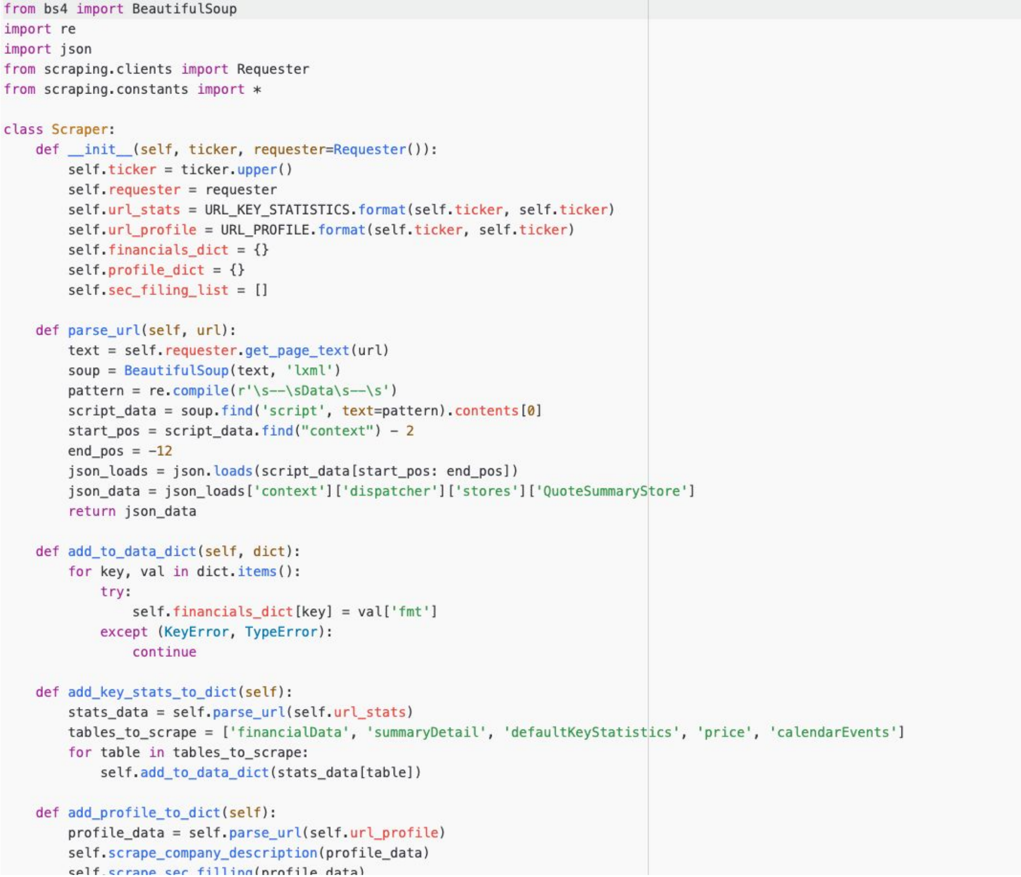
- Web browser automation tool, developed for web testing
- Pros:
  - Incorporates requesters and parsers
  - Can interact with web controls and invoke JavaScript
- Drawbacks:
  - Much longer time and more expensive to run the script: Selenium needs to download the full contents of the web page, not just HTML, resulting in a slower process and higher CPU and memory usage
  - More potential for reliability issues

Se
Selenium
EFFICIENT SELENIUM SCRIPTING AND TROUBLESHOOT SCENARIOS
© www.SoftwareTestingHelp.com

Winner: Beautiful Soup

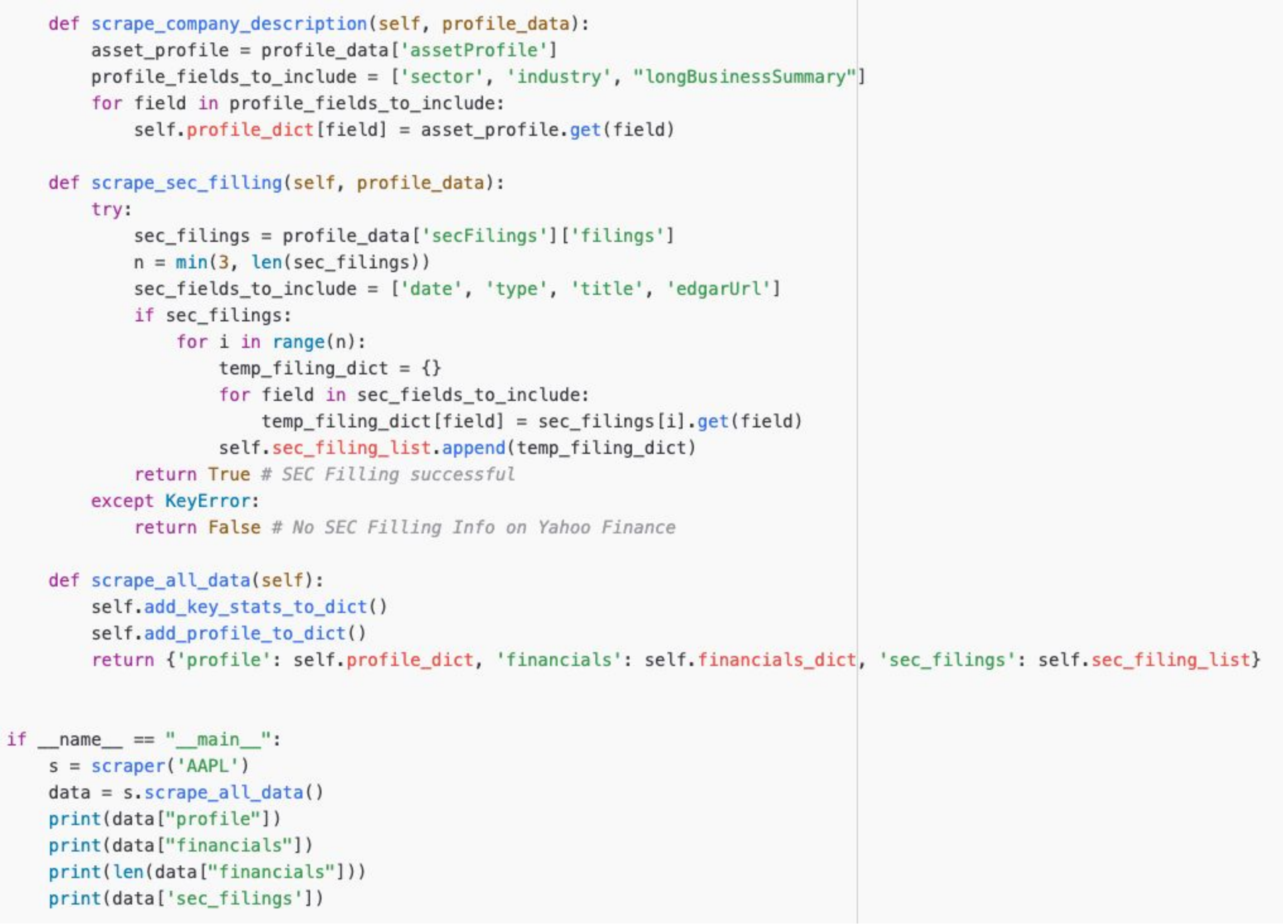
- Beginner friendly and the amount of resources and libraries available are the main reasons that we chose Beautiful Soup.
- Beautiful Soup tends to need more concise code than Selenium when scraping the same data.
- Time Complexity is also considered, especially when we want the users to have the ability to scrape data for multiple stock tickers at a time

Code



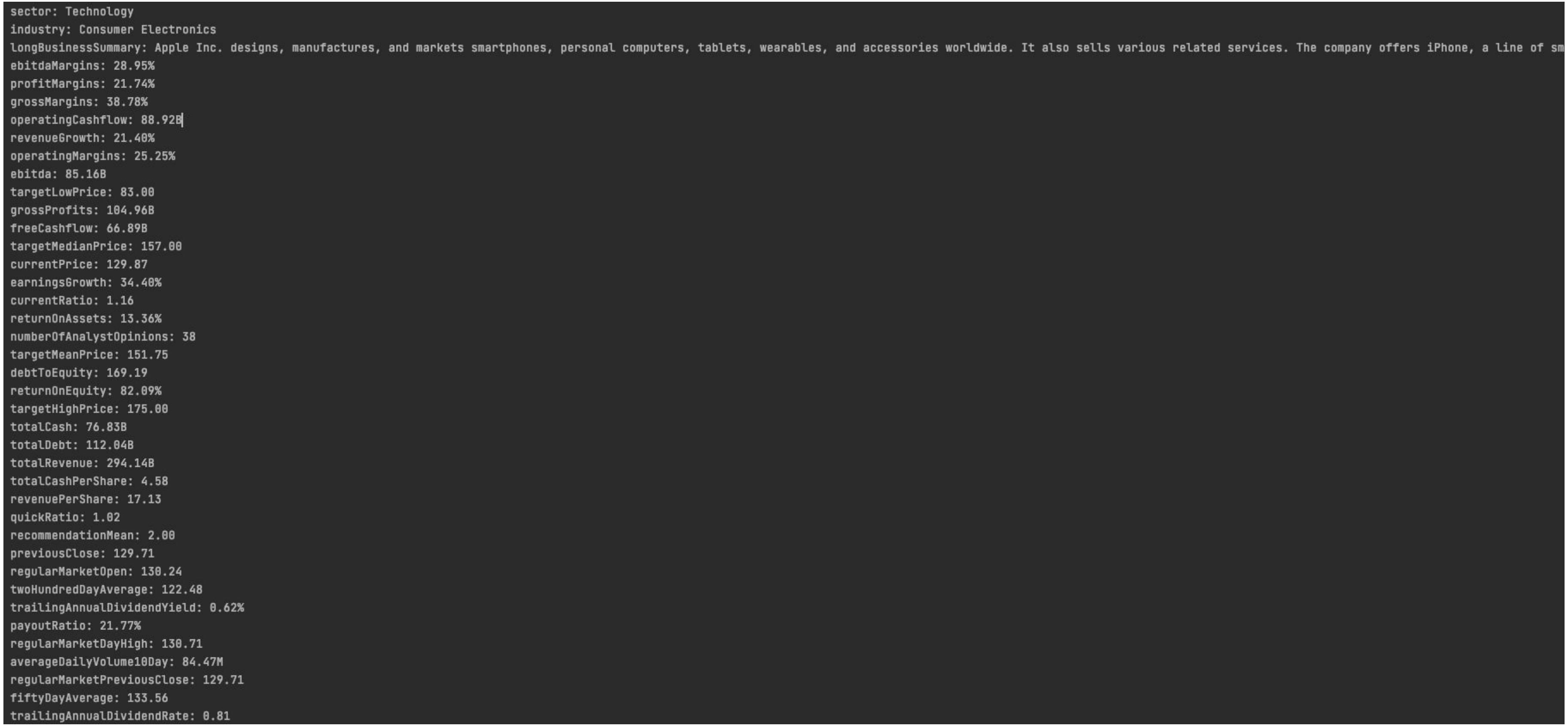
```python
from bs4 import BeautifulSoup
import re
import json
from scraping.clients import Requester
from scraping.constants import *

class Scraper:
    def __init__(self, ticker, requester=Requester()):
        self.ticker = ticker.upper()
        self.requester = requester
        self.url_stats = URL_KEY_STATISTICS.format(self.ticker, self.ticker)
        self.url_profile = URL_PROFILE.format(self.ticker, self.ticker)
        self.financials_dict = {}
        self.profile_dict = {}
        self.sec_filing_list = []

    def parse_url(self, url):
        text = self.requester.get_page_text(url)
        soup = BeautifulSoup(text, 'lxml')
        pattern = re.compile(r'\s--\sData\s--\s')
        script_data = soup.find('script', text=pattern).contents[0]
        start_pos = script_data.find("context") - 2
        end_pos = -12
        json_loads = json.loads(script_data[start_pos: end_pos])
        json_data = json_loads['context']['dispatcher']['stores']['QuoteSummaryStore']
        return json_data

    def add_to_data_dict(self, dict):
        for key, val in dict.items():
            try:
                self.financials_dict[key] = val['fmt']
            except (KeyError, TypeError):
                continue

    def add_key_stats_to_dict(self):
        stats_data = self.parse_url(self.url_stats)
        tables_to_scrape = ['financialData', 'summaryDetail', 'defaultKeyStatistics', 'price', 'calendarEvents']
        for table in tables_to_scrape:
            self.add_to_data_dict(stats_data[table])

    def add_profile_to_dict(self):
        profile_data = self.parse_url(self.url_profile)
        self.scrape_company_description(profile_data)
        self.scrape_sec_filling(profile_data)
```

Code Continued



```python
    def scrape_company_description(self, profile_data):
        asset_profile = profile_data['assetProfile']
        profile_fields_to_include = ['sector', 'industry', "longBusinessSummary"]
        for field in profile_fields_to_include:
            self.profile_dict[field] = asset_profile.get(field)

    def scrape_sec_filling(self, profile_data):
        try:
            sec_filings = profile_data['secFilings']['filings']
            n = min(3, len(sec_filings))
            sec_fields_to_include = ['date', 'type', 'title', 'edgarUrl']
            if sec_filings:
                for i in range(n):
                    temp_filing_dict = {}
                    for field in sec_fields_to_include:
                        temp_filing_dict[field] = sec_filings[i].get(field)
                    self.sec_filing_list.append(temp_filing_dict)
            return True # SEC Filling successful
        except KeyError:
            return False # No SEC Filling Info on Yahoo Finance

    def scrape_all_data(self):
        self.add_key_stats_to_dict()
        self.add_profile_to_dict()
        return {'profile': self.profile_dict, 'financials': self.financials_dict, 'sec_filings': self.sec_filing_list}


if __name__ == "__main__":
    s = scraper('AAPL')
    data = s.scrape_all_data()
    print(data["profile"])
    print(data["financials"])
    print(len(data["financials"]))
    print(data['sec_filings'])
```

Results



```
sector: Technology
industry: Consumer Electronics
longBusinessSummary: Apple Inc. designs, manufactures, and markets smartphones, personal computers, tablets, wearables, and accessories worldwide. It also sells various related services. The company offers iPhone, a line of sm
ebitdaMargins: 28.95%
profitMargins: 21.74%
grossMargins: 38.78%
operatingCashflow: 88.92B
revenueGrowth: 21.40%
operatingMargins: 25.25%
ebitda: 85.16B
targetLowPrice: 83.00
grossProfits: 104.96B
freeCashflow: 66.89B
targetMedianPrice: 157.00
currentPrice: 129.87
earningsGrowth: 34.40%
currentRatio: 1.16
returnOnAssets: 13.36%
numberOfAnalystOpinions: 38
targetMeanPrice: 151.75
debtToEquity: 169.19
returnOnEquity: 82.09%
targetHighPrice: 175.00
totalCash: 76.83B
totalDebt: 112.04B
totalRevenue: 294.14B
totalCashPerShare: 4.58
revenuePerShare: 17.13
quickRatio: 1.02
recommendationMean: 2.00
previousClose: 129.71
regularMarketOpen: 130.24
twoHundredDayAverage: 122.48
trailingAnnualDividendYield: 0.62%
payoutRatio: 21.77%
regularMarketDayHigh: 130.71
averageDailyVolume10Day: 84.47M
regularMarketPreviousClose: 129.71
fiftyDayAverage: 133.56
trailingAnnualDividendRate: 0.81
```