FE 530: Introduction to Financial Engineering

Automating Financial Strategies

Scott Russell

Stevens Institute of Technology

December 15, 2016

In today's modern economy, with the advent of metadata and its easy accessibility, one would assume predictions, analysis, and forecasts would be easy. But let us take a moment to picture someone with very little to no financial experience who is handed a company's financial statement. Given the proper amount of time, I believe that person could tell you the company's revenue, cost of sales, and net income for the quarter, but that's about it. They wouldn't be able to make a smart investment decision on the companies future financial growth. To them, the income statement, balance sheet, and cash flow statement are hieroglyphics containing incomprehensible figures. However, to a seasoned investor, those 3 pieces of financial gold are critical to determining the value of a given stock. Now, imagine Joe Smo is given a piece of software that takes, as an input, a company's ticker symbol. The software then gathers and translates the apparent hieroglyphs from the specified company's Financial Statements, as well as the company's competitors, and outputs, based on financial factors, the best company to invest in. For this project, I have engineered a prototype application that compares financial ratios, gathered from public resources, and produces the best investment alternative for an average investor.

This software is designed to help the average person find a company to invest in. Before I could build out the application, I had to learn more about investing myself. I conducted a great deal of research revolved around basic investment strategies, company information, and financial statements. Initially, I thought I could merge numerous types of investment strategies into one magical tool. However, after reading an article by Brian Lund where he says that an investor should not "try to become a jack of all trades"[1], but, instead, "become a Master of One"[1], I decided to focus my research and practical development into a single investment strategy. I began asking my Finance and Quantitative Finance friends which kinds of investment strategies they followed when investing. The majority of them were interested in fundamental analysis; focusing on Macro trends and investing based on where the economy was headed. The others were more interested in technical analysis; looking at current trends and patterns to predict future trends and patterns. Because I have a computer engineering background, and I lacked experience in macroeconomics, I was most comfortable using technical analysis and problem solving for my application. While researching technical strategies on investopedia.com, I uncovered the ability to use financial ratios when determining a company's strength in the market when compared to it's competitors. This lead to the prototype I have created.

I investigated several financial ratios that I felt would be sufficient measurements of a company's financial success. Below is the list of the financial ratios I used, what they mean, how they are calculated, and their importance:

1. **Inventory Turnover Ratio**
    - **Definition:** "ratio showing how many times a company's inventory is sold and replaced over a period of time."[2]

- **Calculation:** Cost of Revenue (Cost of Sales) for a period of time (last 4 quarters) divided by Average Inventory (the sum of Inventory at Beginning and Inventory at End divided by 2).

- **Importance:** A high ratio "implies either strong sales and/or large discounts"(2), where as a low ratio "implies weak sales and, therefore, excess inventory."(2)

2. **Working Capital Turnover**
   • **Definition:** "used to analyze the relationship between the money that funds operations and the sales generated from these operations."(3)

   • **Calculation:** Total Revenue over a period of time (last 4 quarters) divided by Average Working Capital (the difference between Total Current Assets at Beginning minus Total Current Liabilities at Beginning plus the difference between Total Current Assets at End minus Total Current Liabilities at Beginning all divided by 2).

   • **Importance:** A high ratio "shows that management is being very efficient in using a company's short-term assets and liabilities for supporting sales."(3) A low ratio "shows a business is investing in too many accounts receivable (AR) and inventory assets for supporting its sales."(3)

3. **Current Ratio**
   • **Definition:** "liquidity ratio that measures a company's ability to pay short-term and long-term obligations."(4)

   • **Calculation:** Total Current Assets divided by Total Current Liabilities.

   • **Importance:** "The higher the current ratio, the more capable the company is of paying its obligations, as it has a larger proportion of asset value relative to the value of its liabilities."(4) "A ratio under 1 indicates that a company's liabilities are greater than its assets and suggests that the company in question would be unable to pay off its obligations"(4).

4. **Quick Ratio**
   • **Definition:** "measures a company's ability to meet is short-term obligations with its most liquid assets."(5)

   • **Calculation:** Total Current Assets minus Inventory all divided by Total Current Liabilities.

• **Importance:** "The higher the quick ratio, the better the company's liquidity position."(5)

5. **Debt Ratio**
   • **Definition:** "measures the extent of a company's or consumer's leverage."(6) Leverage is another way of saying debt.

   • **Calculation:** Total Debt divided by Total Assets.

   • **Importance:** "The higher this ratio, the more leveraged the company is, implying greater financial risk."(6) The lower, the better.

6. **Financial Leverage Ratio**
   • **Definition:** "look[s] at how much capital comes in the form of debt (loans), or assesses the ability of a company to meet financial obligations."(7)

   • **Calculation:** Total Debt divided by Total Equity.

   • **Importance:** "a ratio greater than 2.0 indicates a risky scenario for the investor, however this yardstick can vary on industry."(7) The lower, the better.

7. **Net Profit Margin (percentage)**
   • **Definition:** "ratio of net profits to revenues for a company or business segment."(8)

   • **Calculation:** Net Income divided by Total Revenue.

   • **Importance:** The higher the ratio, the better, however, a "low profit margin [doesn't] necessarily equate to low profits."(8)

8. **Return on Equity (percentage)**
   • **Definition:** "measures a corporation's profitability by revealing how much profit a company generates with the money shareholders have invested."(9)

   • **Calculation:** Net Income divided by Total Equity.

   • **Importance:** "The ROE is useful for comparing the profitability of a company to that of other firms in the same industry."(9) The higher, the better.

9. **Earnings per Share**
   • **Definition:** "company's profit allocated to each outstanding share of common

stock."(10)

- **Calculation:** Net Income divided by Shares Outstanding.

- **Importance:** The higher, the better. If negative, company is losing money. If two companies generated "the same EPS number, but one could do so with less equity (investment) – that company would be more efficient at using its capital to generate income"(10).

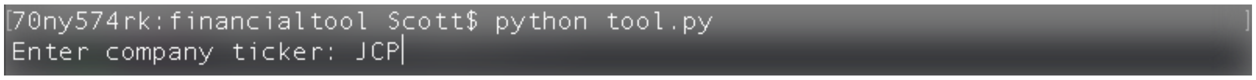
10. **Price to Earnings Ratio**
    - **Definition:** "ratio for valuing a company that measures its current share price relative to its per-share earnings."(11)

    - **Calculation:** Market Price divided by Earnings per Share.

    - **Importance:** "A high P/E suggests that investors are expecting higher earnings growth in the future compared to companies with a lower P/E. A low P/E can indicate either that a company may currently by undervalued or that the company is doing exceptionally well relative to its past trends."(11)

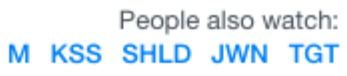
After carrying out the necessary research into the above ratios, I successfully translated them into code and create meaningful comparisons, which I placed into my application.

Using Python as my programming language of choice and Yahoo Finance as my data source, I only had to create two classes to make up my entire application. The main class, "tool.py", acted as the central HUB for the application. Its primary purpose was to first take in a user's input, a company's ticker symbol, gather information for that company, then output a final company that would be the best investment option. The other class, "ratios.py", was made up of functions containing the necessary logic for each individual ratio stated above. The application would begin by asking the user to input a company's ticker symbol:



```
[70ny574rk:financialtool Scott$ python tool.py
Enter company ticker: JCP
```

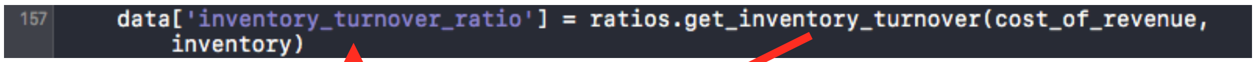
Next, the program would find 5 similar companies by scraping the company's Yahoo Finance page for the ticker symbols located under "People also watch:".



People also watch:
M KSS SHLD JWN TGT

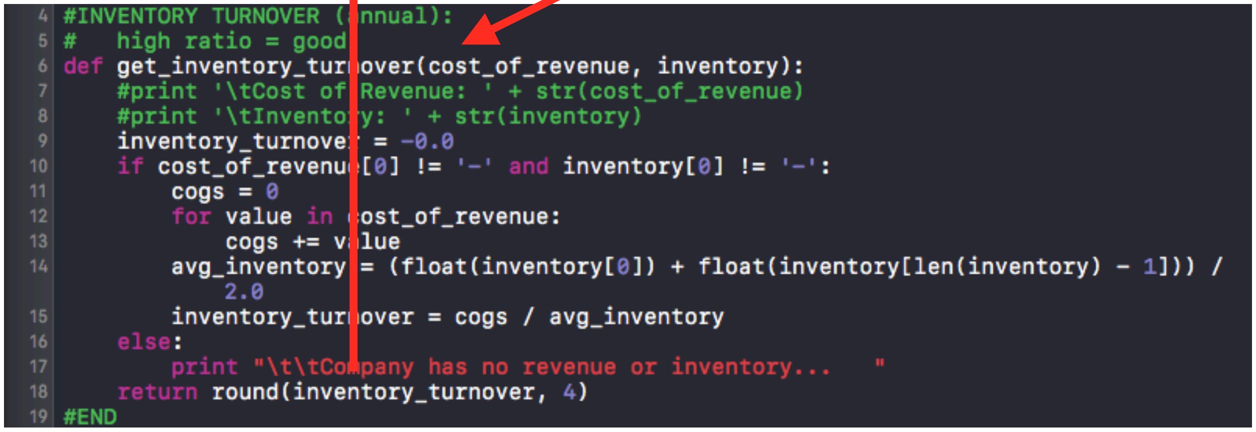
After finding the similar companies, the application would accumulate each company's financial statements (income statement, balance sheet, cash flow statement) by scraping each's Yahoo Finance page's "Financials" tab. The program then calculated each company's financial ratios. For each ratio, the "tool.py" class would make a call to the "ratios.py" class and sending the necessary data. In response, the "ratios.py" class would calculate the ratio with the given data and return the specified ratio value back to "tool.py".

tool.py:



```
    data['inventory_turnover_ratio'] = ratios.get_inventory_turnover(cost_of_revenue,
        inventory)
```

ratios.py:



```
#INVENTORY TURNOVER (annual):
#   high ratio = good
def get_inventory_turnover(cost_of_revenue, inventory):
    #print '\tCost of Revenue: ' + str(cost_of_revenue)
    #print '\tInventory: ' + str(inventory)
    inventory_turnover = -0.0
    if cost_of_revenue[0] != '-' and inventory[0] != '-':
        cogs = 0
        for value in cost_of_revenue:
            cogs += value
        avg_inventory = (float(inventory[0]) + float(inventory[len(inventory) - 1])) /
            2.0
        inventory_turnover = cogs / avg_inventory
    else:
        print "\t\tCompany has no revenue or inventory...    "
    return round(inventory_turnover, 4)
#END
```

Finally, the application takes the collected financial ratio data for each company and outputs everything into a nicely formatted .CSV, which can be easily used for analysis. In the future, I plan to automate the process of selecting the best company out of all six by assigning a weight to each ratio, multiplying it by the ratio value for each company, then adding up the totals. The company with the highest score is the winner.

Outputted Results From tool.py:

	JCP	M	KSS	SHLD	JWN	TGT
Inventory Turnover	2.512	2.4317	2.5799	3.5802	4.3251	5.2902
Working Capital Turnover	8.3852	14.9283	8.4045	55.4899	118.5020	82.4731
Current Ratio	1.4743	1.2177	1.6035	1.0412	1.0444	1.0158
Quick Ratio	0.2193	0.1702	0.2646	0.1630	0.3005	0.2203
Debt Ratio	0.4998	0.3526	0.3285	0.3957	0.3487	0.3323
Financial Leverage Ratio	4.2070	1.9797	0.9054	-1.2719	3.2302	1.1587
Net Profit Margin	-2.0823%	2.6711%	3.1747%	-9.3805%	2.2754%	4.7507%
Return on Equity	-22.98%	18.42%	11.78%	-64.91%	38.72%	30.23%
Earnings Per Share	-0.8498	2.2835	3.3943	-20.4989	1.9211	5.9569
Price to Earnings Ratio	-10.4731	16.3784	16.0092	-0.4995	26.8492	13.0622

The above table is the result of running my application with the original company ticker, JCP (J.C. Penney). M (Macy's), KSS (Kohl's), SHLD (Sears), JWN (Nordstrom), and TGT (Target) are the found "similar companies". The cells highlighted in green are the winners for each financial ratio row. As you can see immediately, TGT should be the winner because the column has won the most rows. And for this particular example, TGT is the best option out of the six listed, however, each ratio carries a different weight of importance which might not yield the same results for who the winner of each row is.

In the Inventory Turnover Ratio row, each company has similar ratios. This is because they are all part of the same industry and all happen to be retail stores. As stated above, Inventory Turnover refers to a company's ability to sell it's inventory then replace it over a certain amount of time. Therefore, if a company's Cost of Revenue (Cost of Sales) is high, but it's Average Inventory is low, then this means the company is selling more of its goods, which yields a higher turnover ratio. The higher the ratio, the better the company is at selling inventory compared to its competitors. In the outputted results, all seem to have a strong and similar Inventory Turnover Ratio and thus it is too difficult to determine the best company. For now, because TGT has the highest ratio, we will say it receives first place with JWN second and SHLD third.

In the Working Capital Turnover Ratio row, the ratios are drastically different from one another. As stated above, Working Capital Turnover can show that management is being either very efficient or inefficient in using a company's Total Current Assets and Liabilities for supporting it's sales. The higher the ratio, the better the company is at utilizing its working capital (Total Current Assets - Total Current Liabilities) to drive Revenue compared to its competitors. However, a ratio above 80 is typically indicates that a company does not have enough working capital to support its sales. After looking at SHLD, JWN, and TGT's Total Revenue and Working Capital, SHLD's and TGT's amount of Revenue help to give them their high ratios of 55.4899 and 82.4731, respectively, where as JWN's high amount of Revenue and low amount of Working Capital helps to produce an inflated ratio of 118.5020. Once again, TGT is first place with SHLD second and JWN third. Regardless, this is still not enough information to make the informed decision that TGT is the right company to invest in.

In the Current Ratio row, all company ratios are above 1.0, which means that their Total Current Assets are greater than their Total Current Liabilities. As stated above, the Current Ratio provides us with an idea of how capable a company is at paying back it's liabilities with its assets. As a rule of thumb, a ratio close to 2.0 is the sweet spot meaning that a company's assets are twice as much as its liabilities. A ratio lower than 1.0 basically means that the company can't pay off its obligations. Referring to the table, KSS has the best ratio of 1.6035 where as TGT has the lowest of 1.0158. Even though TGT is last, its ratio is still greater than 1.0 meaning it could still pay off its obligations if it had to. KSS receives first with JCP second and M third.

In the Quick Ratio row, all ratios are very similar to one another. As stated above, the Quick Ratio measures the dollar amount of current assets available for each dollar of current liabilities, or in other words, the Quick Ratio portrays a company's liquidity position. This ratio is a bit more straight forward as it subtracts the total inventory from the total current assets of the company. The higher the ratio, the better the company's liquidity position compared to its competitors. In the table, JWN has the highest ratio of 0.3005 due to its large amount of Current Assets and minimal amount of Inventory. Therefore, JWN is first place with TGT second and JCP third.

In the Debt Ratio row, all ratios are very close in value to each other. As stated above, the Debt Ratio indicates how much of a company's Assets are provided from debt, or how leveraged a company is. The higher the ratio, the more debt the company has, which implies that the company is at financial risk. A company with a low debt ratio symbolizes that it is using its Assets and Debt effectively and efficiently and therefore would be the better company compared to its competitors. As a rule of thumb, a ratio of less than 0.4 is good. In the table, KSS has the lowest ratio of 0.3285 and TGT has the second lowest of 0.3323. KSS is first place with TGT second and JWN third.

In the Financial Leverage Ratio row, the ratios begin to differentiate a bit. As stated above, the Financial Leverage Ratio indicates how much of a company's Equity is provided from debt. It is very similar to the Debt Ratio, but instead of checking assets, it checks equity. A high ratio means that the company has been aggressive in financing it's growth with debt where as a low ratio means that the company is not utilizing debt to grow its business. As a rule of thumb, a ratio greater than 2.0 presents a risky setting. In the table, KSS has the lowest ratio of 0.9054 and TGT has the second lowest ratio of 1.1587. These are both decent ratios. Not too high and not too low. KSS is first with TGT second and M third.

In the Net Profit Margin row, there are a couple negative percentages, but, mainly, all are similar. As stated above, the Net Profit Margin is an excellent indicator of a company's financial health. It shows how profitable a company is being. Typically, a high Net Profit Margin is great as it shows the company is turning a big profit, however, a low margin isn't a bad thing as long as the company is producing enough Net

Income. When compared to its competitors, the company with the higher Net Profit Margin percentage is the better choice. In the table, TGT has the highest margin of 4.7507%. Even though this may seem low, TGT has a large Net Income that helps to grow it's business annually. TGT is first with KSS second and M third.

In the Return on Equity (ROE) row, the majority of the ratios are similar with the exception of the financially inept companies, like SHLD and JCP. As stated above, Return on Equity is another useful indicator when looking at the profitability of a company. A high Return on Equity means that the company is yielding a big profit compared to the overall money invested into the company. Therefore, when compared to its competitors, the company with the higher ROE is the better investment choice. In the table, JWN has the highest ROE at 0.3872 and TGT has the second highest of 0.3023. JWN is first with TGT second and M third.

In the Earnings per Share row, the ratios vary slightly from company to company. As stated above, the Earning per Share refers to the monetary value of income per share of common stock for a particular company. Those companies with the higher Earnings per Share ratio are the better choice as their Net Income must be high compared to the amount of Shares Outstanding the company has. However, something to note is that two companies can have the same Earnings per Share ratio, but the company with less Total Equity would be the better option due to its outstanding ability to use its capital to generate its profit. In the table, TGT has the best Earnings per Share ratio of 5.9569. TGT is first with KSS second and M third.

Lastly, in the Price to Earnings Ratio row, the top three ratios are similar where as the rest vary significantly. As stated above, the Price to Earnings Ratio (P/E Ratio) indicates how much money an investor can expect to pay for one dollar of that company's earnings. In essence, if a company has a P/E Ratio of 5.0, for every $5 an investor pays for, he/she would receive $1 of earnings. The Technology companies typically have higher P/E Ratios where as Utility stocks have lower P/E Ratios. The lower the ratio, the safer and less risky the company. In the table, TGT has the lowest ratio of 13.0622. TGT is first with KSS second and M third.

Each individual ratio is not enough to determine which company is the best, but, together, they can show us a lot about a company. If you were to just look at the Quick Ratio and Return on Equity, JWN would be considered the best company, which isn't the case. JWN isn't the worst, but it isn't the top company either. In order to accurately determine which company is the strong investment choice, you must take into account each ratio. Doing so you will notice that TGT is indeed the top choice and would be outputted by my software. KSS came in second and JWN third. You may be wondering, but TGT didn't win every ratio category. You would be right, except, if you look closer, TGT came in second and third for every other row except the Current Ratio row. Ironically, the company we started with, JCP, turned out to be the worst company to invest in. Because my software calculates and evaluates every financial ratio for each company, I'm confident in stating that with my tool, the average "Joe" investor can make informed decisions on which companies to invest in. However, pundits in financial strategies would argue that software can't tell the whole story with 100% accuracy.

Like Brian Lund stated in his article, "no matter how many factors or rules you use there is one thing I can tell you for sure… there is no Holy Grail!"(1) Further research into my project helped me to discover that my application isn't perfect. In no way is this piece of software a "sure thing" in determining which company is the better investment. A problem I see with my software is that it doesn't take annual or quarterly averages. It only calculates the most recent quarter ratio data, which limits me from recognizing trends and pattens for given companies. With averages, you can predict with better certainty the future of a company. Another problem that I noticed is that my technical analysis financial ratio strategy doesn't take into account the economy and macro trends. This means that my software has no idea if a bubble of some sort is happening, has happened, or will happen. This would be problematic because my software would still output a company even if the market was crashed therefore providing an inaccurate prediction. Given more time on this project and moving forward, I would strongly consider looking into, at the very least, fixing the issue with collecting annual and quarterly averages so that I may make more accurate projections for a company's future.

With the use of my application, I believe the average investor can be guided to make an educated and smart decision for investing in a company. The only thing the user has to do is find a company's ticker symbol and my software will do the rest. I understand there are a few flaws here and there, but I have full confidence in this tool that I have built. By utilizing the worlds public data, my software has the capability to grow and become a life changing instrument to a simple man with two children, a wife, perhaps a dog, and little to no financial experience.

Bibliography

[1] Brian Lund, August 4, 2016, http://www.sparkfin.com/using-data-make-more%E2%80%AFinformed-stock-market-decisions/

[2] Investopedia, http://www.investopedia.com/terms/i/inventoryturnover.asp

[3] Investopedia, http://www.investopedia.com/terms/w/workingcapitalturnover.asp

[4] Investopedia, http://www.investopedia.com/terms/c/currentratio.asp

[5] Investopedia, http://www.investopedia.com/terms/q/quickratio.asp

[6] Investopedia, http://www.investopedia.com/terms/d/debtratio.asp

[7] Investopedia, http://www.investopedia.com/terms/l/leverageratio.asp

[8] Investopedia, http://www.investopedia.com/terms/n/net_margin.asp

[9] Investopedia, http://www.investopedia.com/terms/r/returnonequity.asp

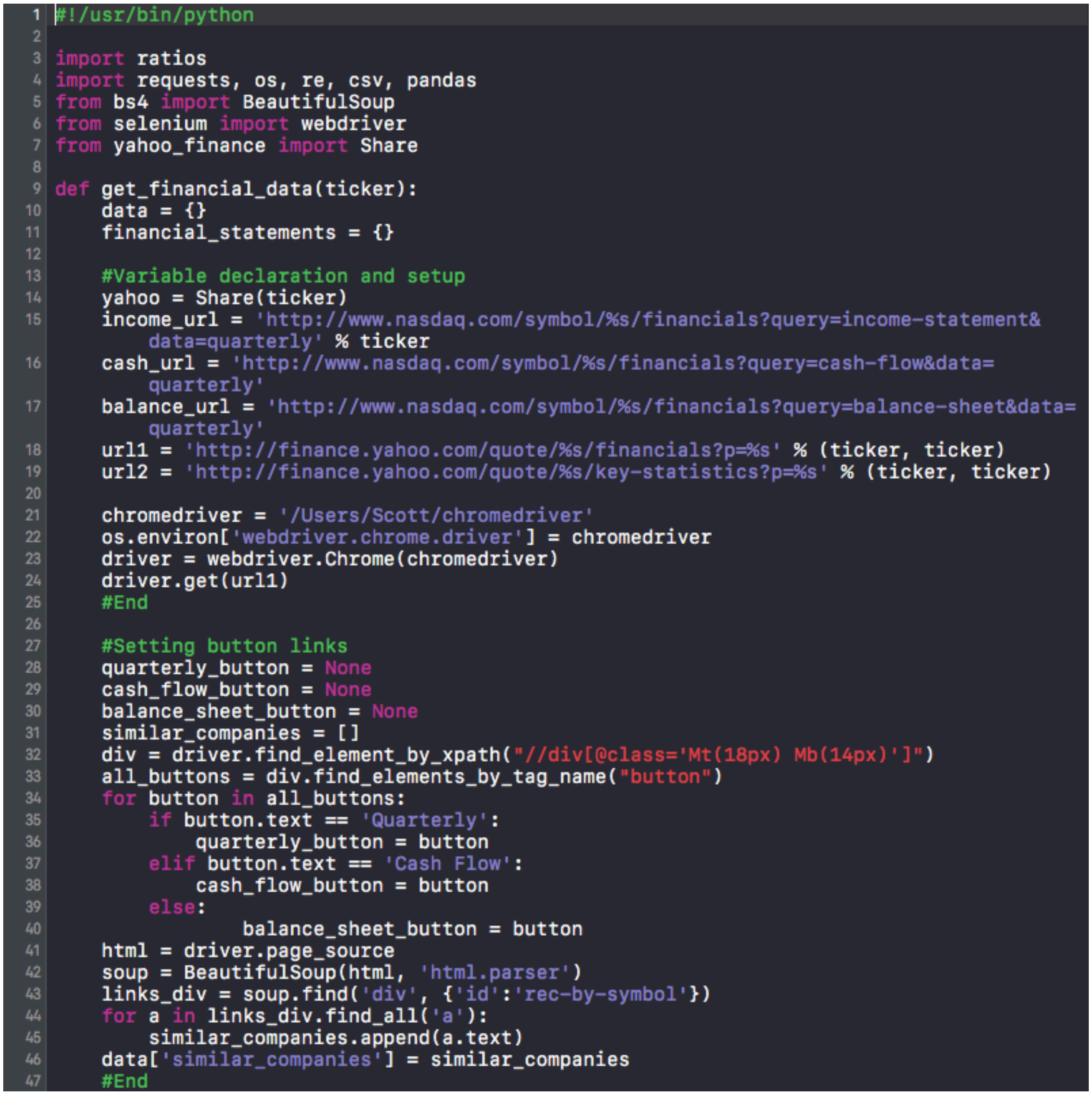
[10] Investopedia, http://www.investopedia.com/terms/e/eps.asp

[11] Investopedia, http://www.investopedia.com/terms/p/price-earningsratio.asp

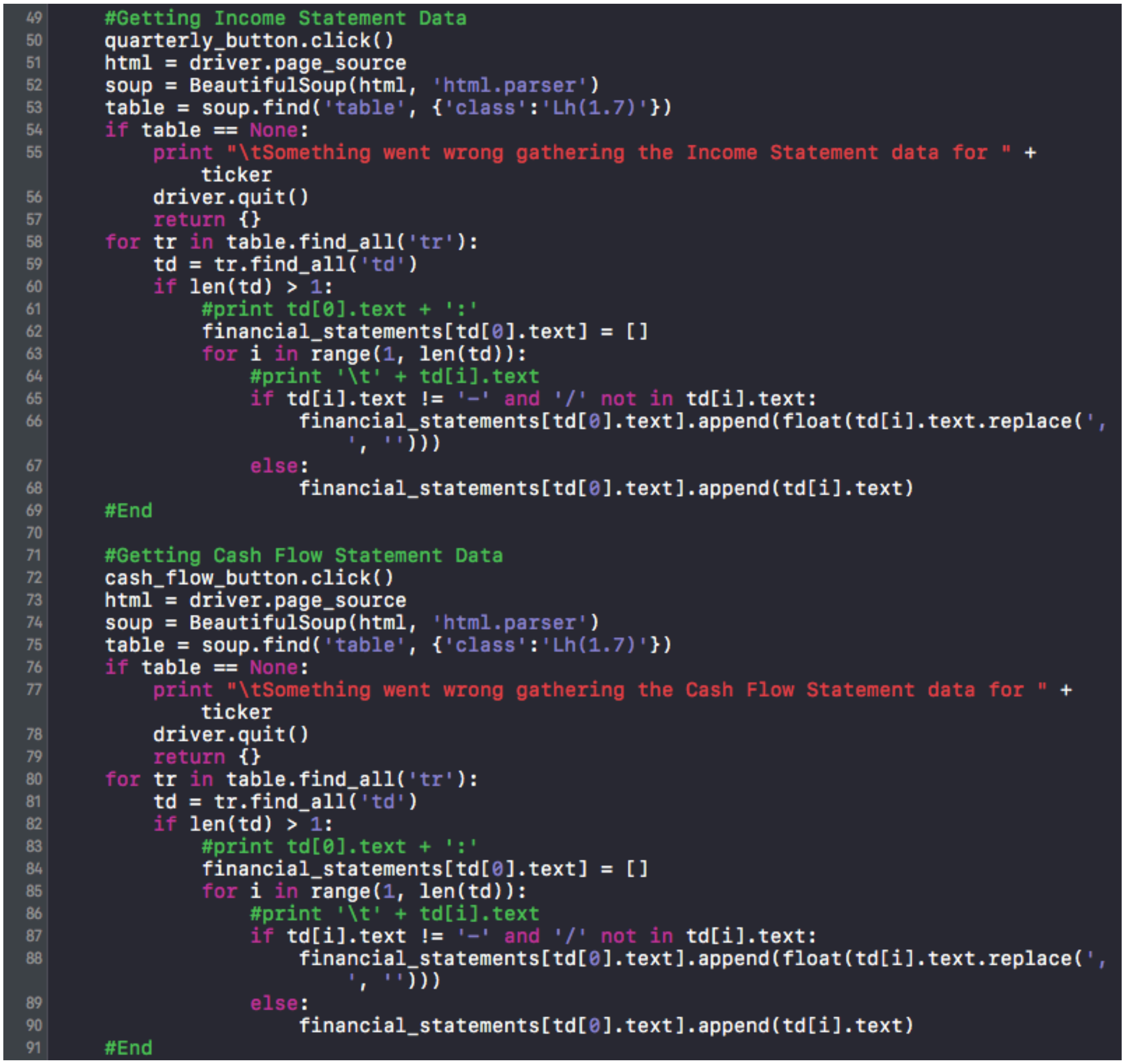
FE 530: Introduction to Financial Engineering

Resources

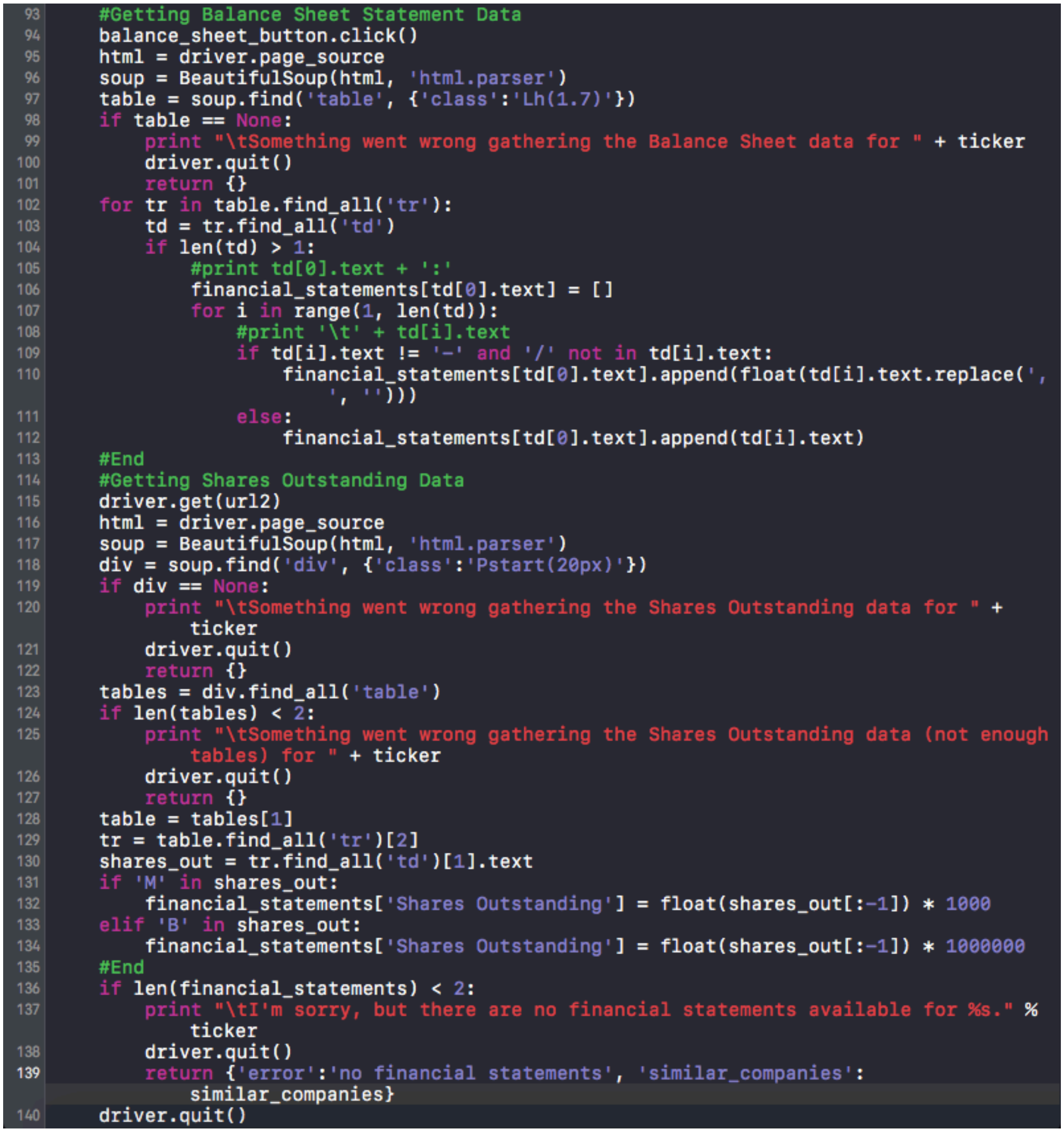
Tool.py



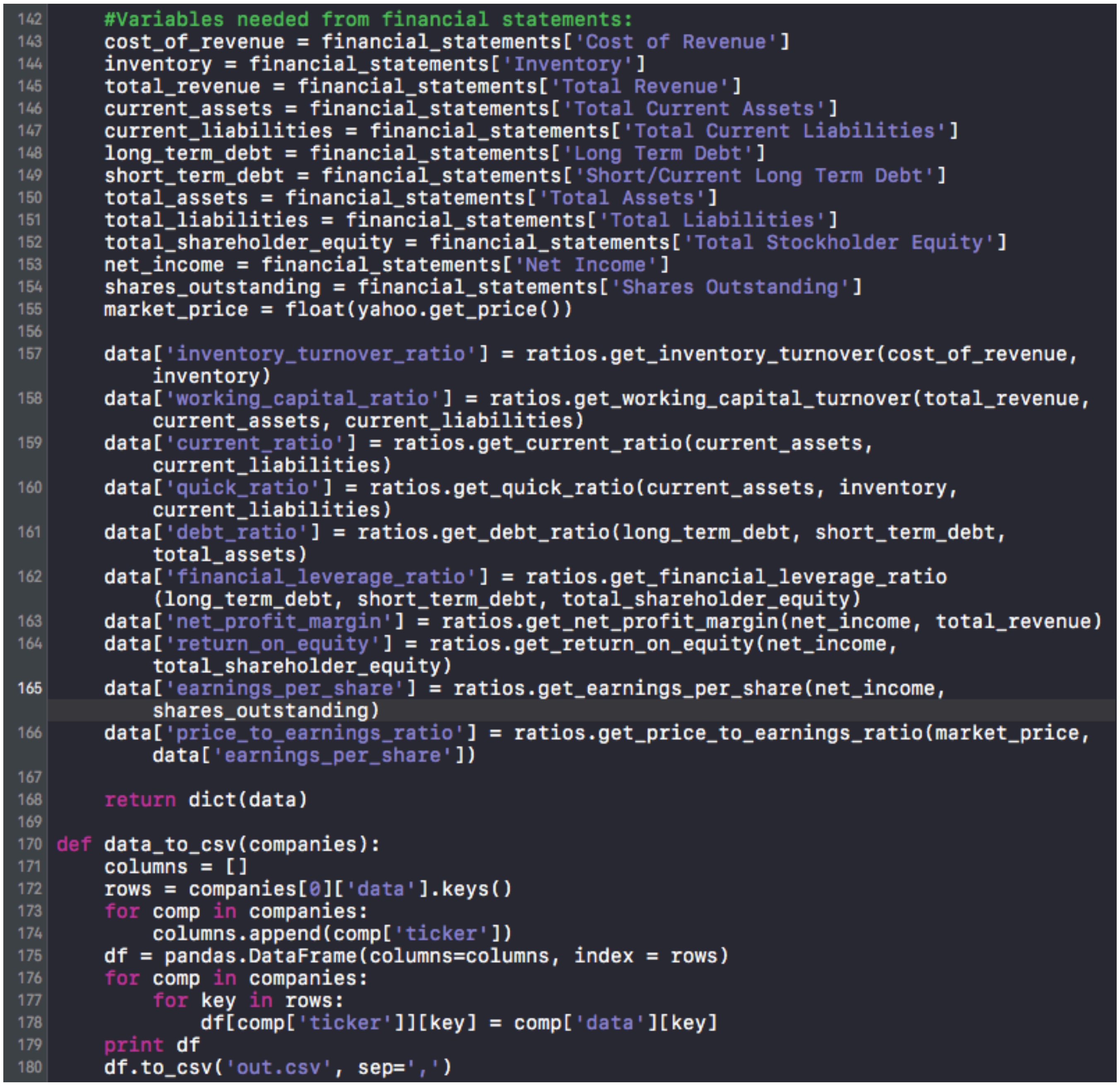
```python
#!/usr/bin/python

import ratios
import requests, os, re, csv, pandas
from bs4 import BeautifulSoup
from selenium import webdriver
from yahoo_finance import Share

def get_financial_data(ticker):
    data = {}
    financial_statements = {}

    #Variable declaration and setup
    yahoo = Share(ticker)
    income_url = 'http://www.nasdaq.com/symbol/%s/financials?query=income-statement&data=quarterly' % ticker
    cash_url = 'http://www.nasdaq.com/symbol/%s/financials?query=cash-flow&data=quarterly'
    balance_url = 'http://www.nasdaq.com/symbol/%s/financials?query=balance-sheet&data=quarterly'
    url1 = 'http://finance.yahoo.com/quote/%s/financials?p=%s' % (ticker, ticker)
    url2 = 'http://finance.yahoo.com/quote/%s/key-statistics?p=%s' % (ticker, ticker)

    chromedriver = '/Users/Scott/chromedriver'
    os.environ['webdriver.chrome.driver'] = chromedriver
    driver = webdriver.Chrome(chromedriver)
    driver.get(url1)
    #End

    #Setting button links
    quarterly_button = None
    cash_flow_button = None
    balance_sheet_button = None
    similar_companies = []
    div = driver.find_element_by_xpath("//div[@class='Mt(18px) Mb(14px)']")
    all_buttons = div.find_elements_by_tag_name("button")
    for button in all_buttons:
        if button.text == 'Quarterly':
            quarterly_button = button
        elif button.text == 'Cash Flow':
            cash_flow_button = button
        else:
                balance_sheet_button = button
    html = driver.page_source
    soup = BeautifulSoup(html, 'html.parser')
    links_div = soup.find('div', {'id':'rec-by-symbol'})
    for a in links_div.find_all('a'):
        similar_companies.append(a.text)
    data['similar_companies'] = similar_companies
    #End
```



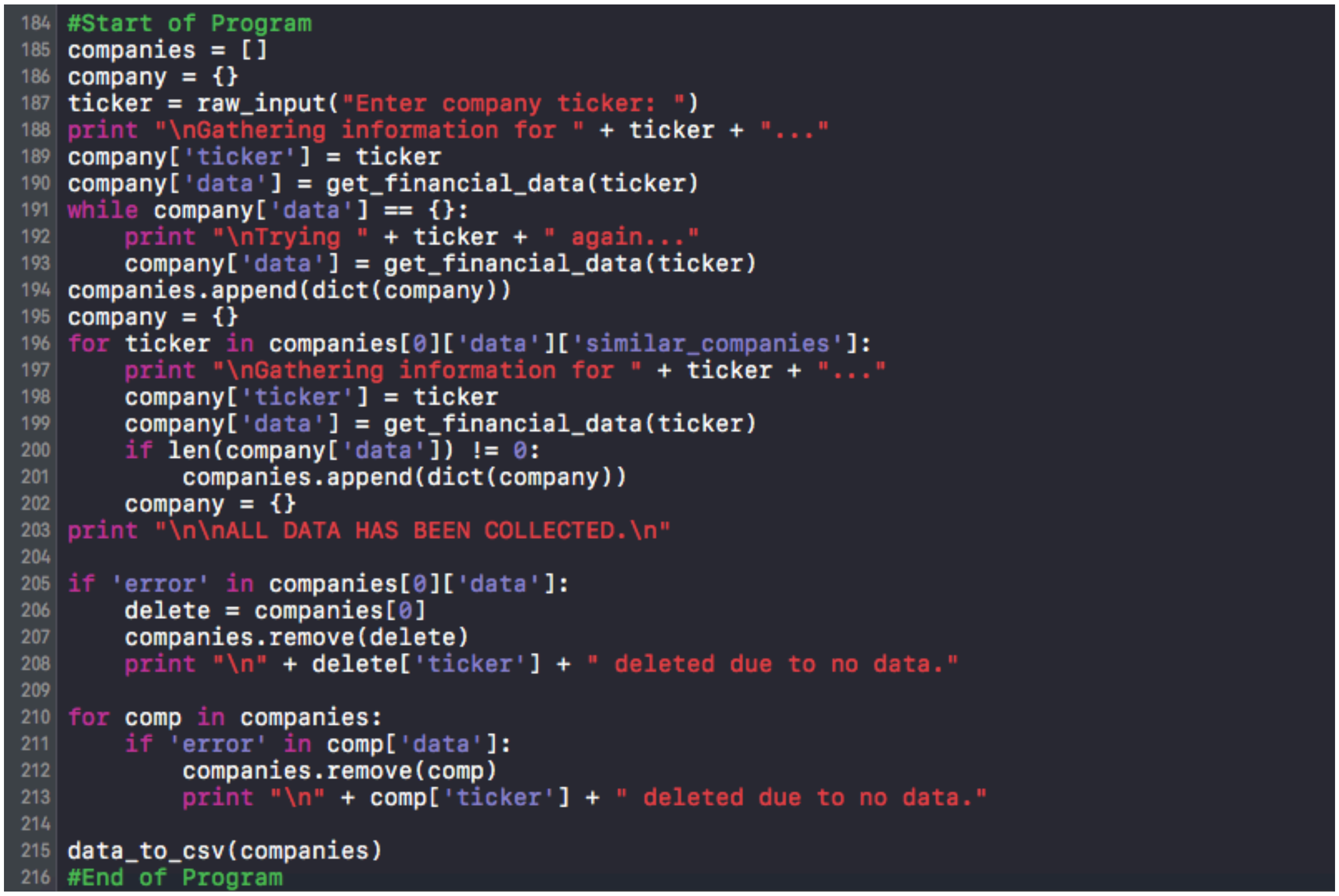
```
    #Getting Income Statement Data
    quarterly_button.click()
    html = driver.page_source
    soup = BeautifulSoup(html, 'html.parser')
    table = soup.find('table', {'class':'Lh(1.7)'})
    if table == None:
        print "\tSomething went wrong gathering the Income Statement data for " + ticker
        driver.quit()
        return {}
    for tr in table.find_all('tr'):
        td = tr.find_all('td')
        if len(td) > 1:
            #print td[0].text + ':'
            financial_statements[td[0].text] = []
            for i in range(1, len(td)):
                #print '\t' + td[i].text
                if td[i].text != '-' and '/' not in td[i].text:
                    financial_statements[td[0].text].append(float(td[i].text.replace(',', '')))
                else:
                    financial_statements[td[0].text].append(td[i].text)
    #End

    #Getting Cash Flow Statement Data
    cash_flow_button.click()
    html = driver.page_source
    soup = BeautifulSoup(html, 'html.parser')
    table = soup.find('table', {'class':'Lh(1.7)'})
    if table == None:
        print "\tSomething went wrong gathering the Cash Flow Statement data for " + ticker
        driver.quit()
        return {}
    for tr in table.find_all('tr'):
        td = tr.find_all('td')
        if len(td) > 1:
            #print td[0].text + ':'
            financial_statements[td[0].text] = []
            for i in range(1, len(td)):
                #print '\t' + td[i].text
                if td[i].text != '-' and '/' not in td[i].text:
                    financial_statements[td[0].text].append(float(td[i].text.replace(',', '')))
                else:
                    financial_statements[td[0].text].append(td[i].text)
    #End
```



```
    #Getting Balance Sheet Statement Data
    balance_sheet_button.click()
    html = driver.page_source
    soup = BeautifulSoup(html, 'html.parser')
    table = soup.find('table', {'class':'Lh(1.7)'})
    if table == None:
        print "\tSomething went wrong gathering the Balance Sheet data for " + ticker
        driver.quit()
        return {}
    for tr in table.find_all('tr'):
        td = tr.find_all('td')
        if len(td) > 1:
            #print td[0].text + ':'
            financial_statements[td[0].text] = []
            for i in range(1, len(td)):
                #print '\t' + td[i].text
                if td[i].text != '-' and '/' not in td[i].text:
                    financial_statements[td[0].text].append(float(td[i].text.replace(',', '')))
                else:
                    financial_statements[td[0].text].append(td[i].text)
    #End
    #Getting Shares Outstanding Data
    driver.get(url2)
    html = driver.page_source
    soup = BeautifulSoup(html, 'html.parser')
    div = soup.find('div', {'class':'Pstart(20px)'})
    if div == None:
        print "\tSomething went wrong gathering the Shares Outstanding data for " + ticker
        driver.quit()
        return {}
    tables = div.find_all('table')
    if len(tables) < 2:
        print "\tSomething went wrong gathering the Shares Outstanding data (not enough tables) for " + ticker
        driver.quit()
        return {}
    table = tables[1]
    tr = table.find_all('tr')[2]
    shares_out = tr.find_all('td')[1].text
    if 'M' in shares_out:
        financial_statements['Shares Outstanding'] = float(shares_out[:-1]) * 1000
    elif 'B' in shares_out:
        financial_statements['Shares Outstanding'] = float(shares_out[:-1]) * 1000000
    #End
    if len(financial_statements) < 2:
        print "\tI'm sorry, but there are no financial statements available for %s." % ticker
        driver.quit()
        return {'error':'no financial statements', 'similar_companies': similar_companies}
    driver.quit()
```

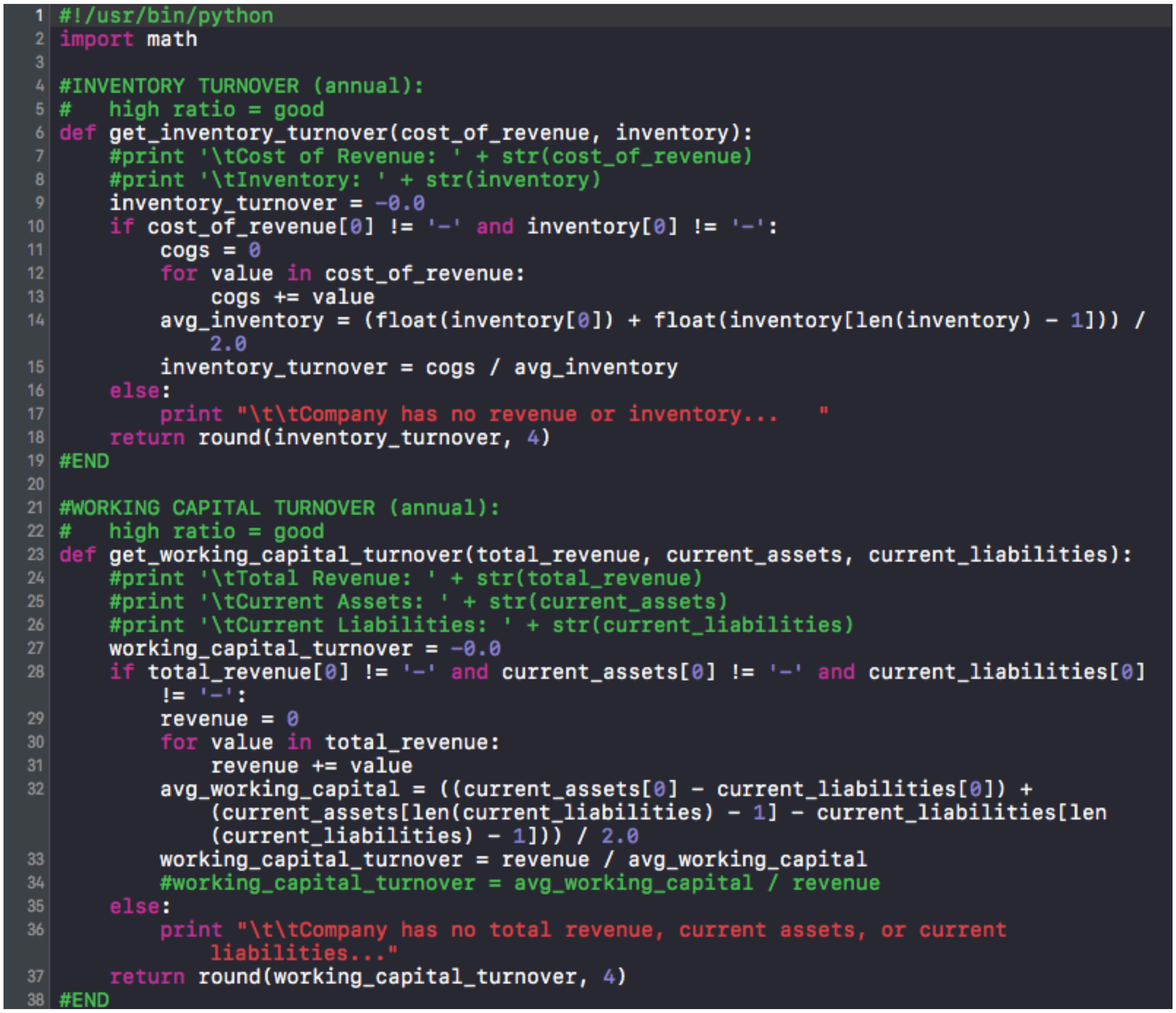


```python
    #Variables needed from financial statements:
    cost_of_revenue = financial_statements['Cost of Revenue']
    inventory = financial_statements['Inventory']
    total_revenue = financial_statements['Total Revenue']
    current_assets = financial_statements['Total Current Assets']
    current_liabilities = financial_statements['Total Current Liabilities']
    long_term_debt = financial_statements['Long Term Debt']
    short_term_debt = financial_statements['Short/Current Long Term Debt']
    total_assets = financial_statements['Total Assets']
    total_liabilities = financial_statements['Total Liabilities']
    total_shareholder_equity = financial_statements['Total Stockholder Equity']
    net_income = financial_statements['Net Income']
    shares_outstanding = financial_statements['Shares Outstanding']
    market_price = float(yahoo.get_price())

    data['inventory_turnover_ratio'] = ratios.get_inventory_turnover(cost_of_revenue,
        inventory)
    data['working_capital_ratio'] = ratios.get_working_capital_turnover(total_revenue,
        current_assets, current_liabilities)
    data['current_ratio'] = ratios.get_current_ratio(current_assets,
        current_liabilities)
    data['quick_ratio'] = ratios.get_quick_ratio(current_assets, inventory,
        current_liabilities)
    data['debt_ratio'] = ratios.get_debt_ratio(long_term_debt, short_term_debt,
        total_assets)
    data['financial_leverage_ratio'] = ratios.get_financial_leverage_ratio
        (long_term_debt, short_term_debt, total_shareholder_equity)
    data['net_profit_margin'] = ratios.get_net_profit_margin(net_income, total_revenue)
    data['return_on_equity'] = ratios.get_return_on_equity(net_income,
        total_shareholder_equity)
    data['earnings_per_share'] = ratios.get_earnings_per_share(net_income,
        shares_outstanding)
    data['price_to_earnings_ratio'] = ratios.get_price_to_earnings_ratio(market_price,
        data['earnings_per_share'])

    return dict(data)

def data_to_csv(companies):
    columns = []
    rows = companies[0]['data'].keys()
    for comp in companies:
        columns.append(comp['ticker'])
    df = pandas.DataFrame(columns=columns, index = rows)
    for comp in companies:
        for key in rows:
            df[comp['ticker']][key] = comp['data'][key]
    print df
    df.to_csv('out.csv', sep=',')
```

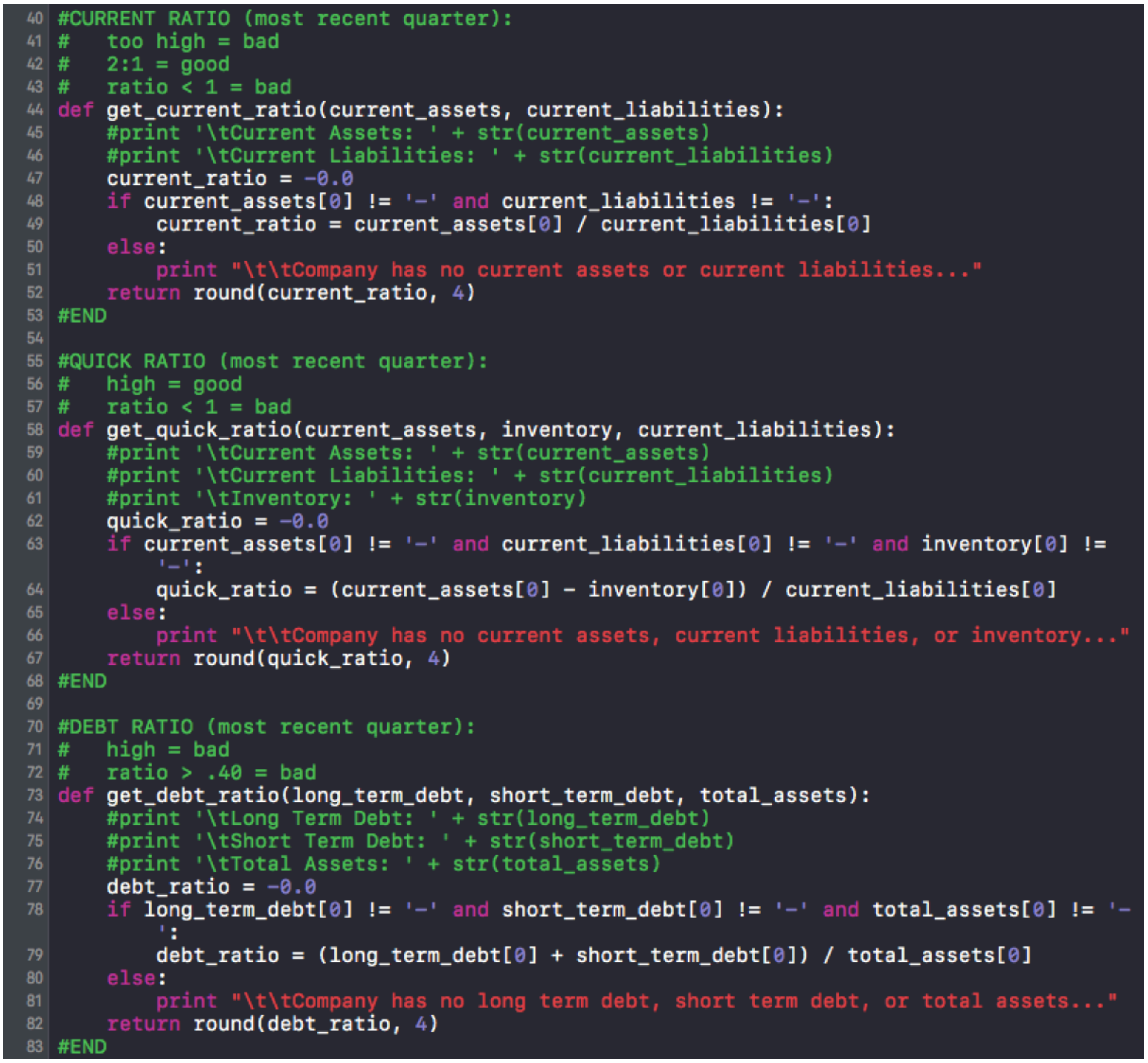


```
#Start of Program
companies = []
company = {}
ticker = raw_input("Enter company ticker: ")
print "\nGathering information for " + ticker + "..."
company['ticker'] = ticker
company['data'] = get_financial_data(ticker)
while company['data'] == {}:
    print "\nTrying " + ticker + " again..."
    company['data'] = get_financial_data(ticker)
companies.append(dict(company))
company = {}
for ticker in companies[0]['data']['similar_companies']:
    print "\nGathering information for " + ticker + "..."
    company['ticker'] = ticker
    company['data'] = get_financial_data(ticker)
    if len(company['data']) != 0:
        companies.append(dict(company))
    company = {}
print "\n\nALL DATA HAS BEEN COLLECTED.\n"

if 'error' in companies[0]['data']:
    delete = companies[0]
    companies.remove(delete)
    print "\n" + delete['ticker'] + " deleted due to no data."

for comp in companies:
    if 'error' in comp['data']:
        companies.remove(comp)
        print "\n" + comp['ticker'] + " deleted due to no data."

data_to_csv(companies)
#End of Program
```

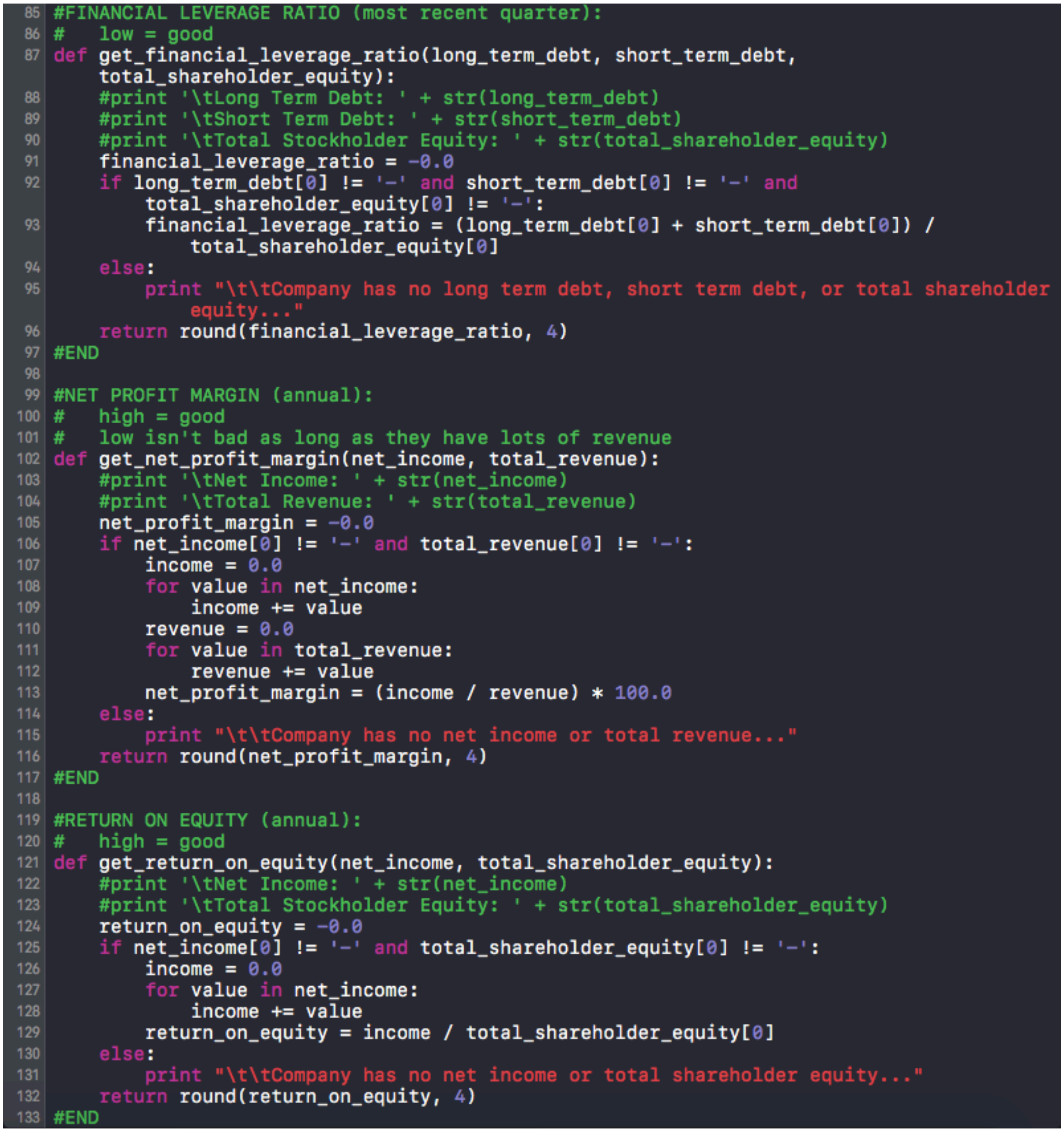
Ratios.py



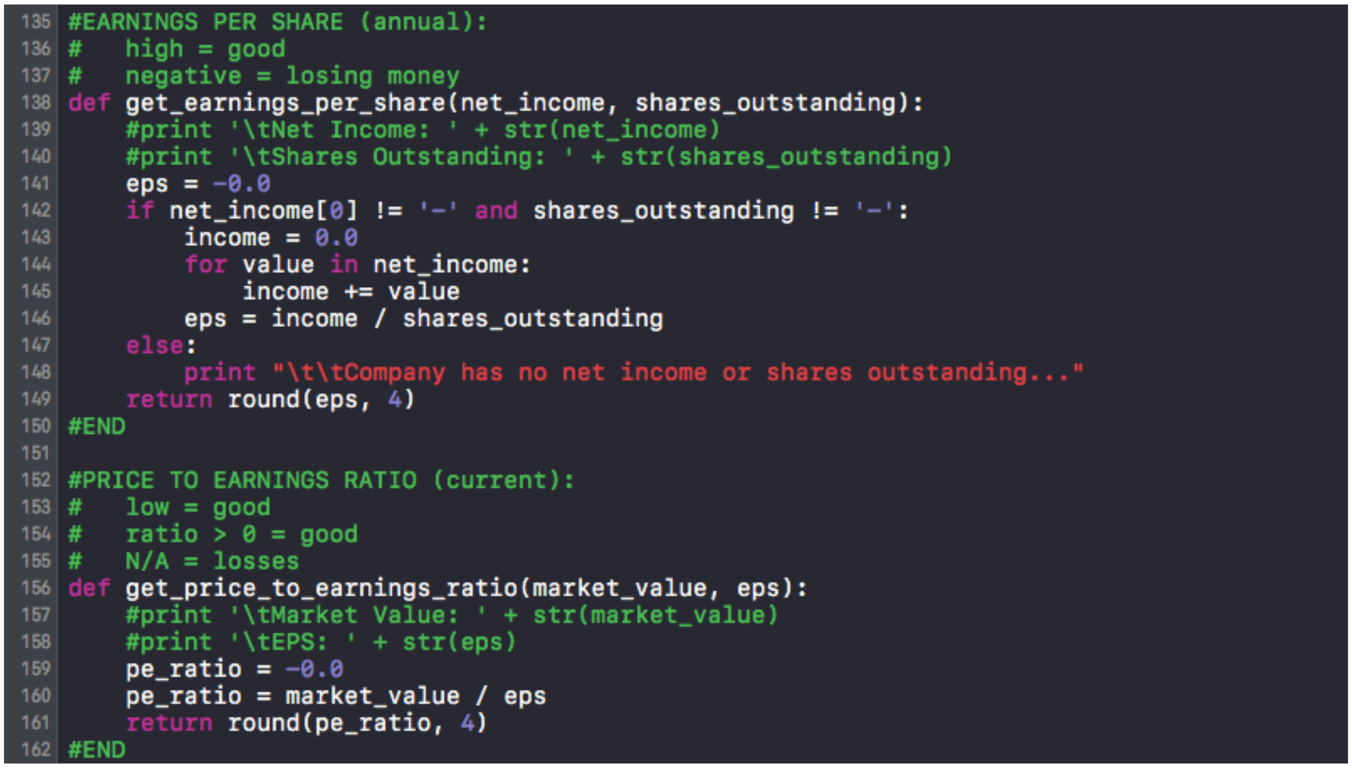
```python
#!/usr/bin/python
import math

#INVENTORY TURNOVER (annual):
#   high ratio = good
def get_inventory_turnover(cost_of_revenue, inventory):
    #print '\tCost of Revenue: ' + str(cost_of_revenue)
    #print '\tInventory: ' + str(inventory)
    inventory_turnover = -0.0
    if cost_of_revenue[0] != '-' and inventory[0] != '-':
        cogs = 0
        for value in cost_of_revenue:
            cogs += value
        avg_inventory = (float(inventory[0]) + float(inventory[len(inventory) - 1])) / 2.0
        inventory_turnover = cogs / avg_inventory
    else:
        print "\t\tCompany has no revenue or inventory...   "
    return round(inventory_turnover, 4)
#END

#WORKING CAPITAL TURNOVER (annual):
#   high ratio = good
def get_working_capital_turnover(total_revenue, current_assets, current_liabilities):
    #print '\tTotal Revenue: ' + str(total_revenue)
    #print '\tCurrent Assets: ' + str(current_assets)
    #print '\tCurrent Liabilities: ' + str(current_liabilities)
    working_capital_turnover = -0.0
    if total_revenue[0] != '-' and current_assets[0] != '-' and current_liabilities[0] != '-':
        revenue = 0
        for value in total_revenue:
            revenue += value
        avg_working_capital = ((current_assets[0] - current_liabilities[0]) + (current_assets[len(current_liabilities) - 1] - current_liabilities[len(current_liabilities) - 1])) / 2.0
        working_capital_turnover = revenue / avg_working_capital
        #working_capital_turnover = avg_working_capital / revenue
    else:
        print "\t\tCompany has no total revenue, current assets, or current liabilities..."
    return round(working_capital_turnover, 4)
#END
```



```
#CURRENT RATIO (most recent quarter):
#    too high = bad
#    2:1 = good
#    ratio < 1 = bad
def get_current_ratio(current_assets, current_liabilities):
    #print '\tCurrent Assets: ' + str(current_assets)
    #print '\tCurrent Liabilities: ' + str(current_liabilities)
    current_ratio = -0.0
    if current_assets[0] != '-' and current_liabilities != '-':
        current_ratio = current_assets[0] / current_liabilities[0]
    else:
        print "\t\tCompany has no current assets or current liabilities..."
    return round(current_ratio, 4)
#END

#QUICK RATIO (most recent quarter):
#    high = good
#    ratio < 1 = bad
def get_quick_ratio(current_assets, inventory, current_liabilities):
    #print '\tCurrent Assets: ' + str(current_assets)
    #print '\tCurrent Liabilities: ' + str(current_liabilities)
    #print '\tInventory: ' + str(inventory)
    quick_ratio = -0.0
    if current_assets[0] != '-' and current_liabilities[0] != '-' and inventory[0] !=
        '-':
        quick_ratio = (current_assets[0] - inventory[0]) / current_liabilities[0]
    else:
        print "\t\tCompany has no current assets, current liabilities, or inventory..."
    return round(quick_ratio, 4)
#END

#DEBT RATIO (most recent quarter):
#    high = bad
#    ratio > .40 = bad
def get_debt_ratio(long_term_debt, short_term_debt, total_assets):
    #print '\tLong Term Debt: ' + str(long_term_debt)
    #print '\tShort Term Debt: ' + str(short_term_debt)
    #print '\tTotal Assets: ' + str(total_assets)
    debt_ratio = -0.0
    if long_term_debt[0] != '-' and short_term_debt[0] != '-' and total_assets[0] != '-
        ':
        debt_ratio = (long_term_debt[0] + short_term_debt[0]) / total_assets[0]
    else:
        print "\t\tCompany has no long term debt, short term debt, or total assets..."
    return round(debt_ratio, 4)
#END
```



```python
#FINANCIAL LEVERAGE RATIO (most recent quarter):
#    low = good
def get_financial_leverage_ratio(long_term_debt, short_term_debt,
    total_shareholder_equity):
    #print '\tLong Term Debt: ' + str(long_term_debt)
    #print '\tShort Term Debt: ' + str(short_term_debt)
    #print '\tTotal Stockholder Equity: ' + str(total_shareholder_equity)
    financial_leverage_ratio = -0.0
    if long_term_debt[0] != '-' and short_term_debt[0] != '-' and
        total_shareholder_equity[0] != '-':
        financial_leverage_ratio = (long_term_debt[0] + short_term_debt[0]) /
            total_shareholder_equity[0]
    else:
        print "\t\tCompany has no long term debt, short term debt, or total shareholder
            equity..."
    return round(financial_leverage_ratio, 4)
#END

#NET PROFIT MARGIN (annual):
#    high = good
#    low isn't bad as long as they have lots of revenue
def get_net_profit_margin(net_income, total_revenue):
    #print '\tNet Income: ' + str(net_income)
    #print '\tTotal Revenue: ' + str(total_revenue)
    net_profit_margin = -0.0
    if net_income[0] != '-' and total_revenue[0] != '-':
        income = 0.0
        for value in net_income:
            income += value
        revenue = 0.0
        for value in total_revenue:
            revenue += value
        net_profit_margin = (income / revenue) * 100.0
    else:
        print "\t\tCompany has no net income or total revenue..."
    return round(net_profit_margin, 4)
#END

#RETURN ON EQUITY (annual):
#    high = good
def get_return_on_equity(net_income, total_shareholder_equity):
    #print '\tNet Income: ' + str(net_income)
    #print '\tTotal Stockholder Equity: ' + str(total_shareholder_equity)
    return_on_equity = -0.0
    if net_income[0] != '-' and total_shareholder_equity[0] != '-':
        income = 0.0
        for value in net_income:
            income += value
        return_on_equity = income / total_shareholder_equity[0]
    else:
        print "\t\tCompany has no net income or total shareholder equity..."
    return round(return_on_equity, 4)
#END
```



```
#EARNINGS PER SHARE (annual):
#    high = good
#    negative = losing money
def get_earnings_per_share(net_income, shares_outstanding):
    #print '\tNet Income: ' + str(net_income)
    #print '\tShares Outstanding: ' + str(shares_outstanding)
    eps = -0.0
    if net_income[0] != '-' and shares_outstanding != '-':
        income = 0.0
        for value in net_income:
            income += value
        eps = income / shares_outstanding
    else:
        print "\t\tCompany has no net income or shares outstanding..."
    return round(eps, 4)
#END

#PRICE TO EARNINGS RATIO (current):
#    low = good
#    ratio > 0 = good
#    N/A = losses
def get_price_to_earnings_ratio(market_value, eps):
    #print '\tMarket Value: ' + str(market_value)
    #print '\tEPS: ' + str(eps)
    pe_ratio = -0.0
    pe_ratio = market_value / eps
    return round(pe_ratio, 4)
#END
```