BA870_Assignment #1_Ji_Qi

March 27, 2022

1 *Name: Ji Qi , Session B1*

2 Income Statement data for Apple Inc (AAPL) for the years 2021, 2020 and 2019.

- Consolidated Statement of Operations
- The SEC URL for this information for Apple is: https://www.sec.gov/Archives/edgar/data/320193/00003201

2.1 Install required libraries

[2]:
```
# Install necessary Python libraries
! pip install requests
! pip install beautifulsoup4
```

```
Requirement already satisfied: requests in /usr/local/lib/python3.7/dist-
packages (2.23.0)
Requirement already satisfied: idna<3,>=2.5 in /usr/local/lib/python3.7/dist-
packages (from requests) (2.10)
Requirement already satisfied: urllib3!=1.25.0,!=1.25.1,<1.26,>=1.21.1 in
/usr/local/lib/python3.7/dist-packages (from requests) (1.24.3)
Requirement already satisfied: chardet<4,>=3.0.2 in
/usr/local/lib/python3.7/dist-packages (from requests) (3.0.4)
Requirement already satisfied: certifi>=2017.4.17 in
/usr/local/lib/python3.7/dist-packages (from requests) (2021.10.8)
Requirement already satisfied: beautifulsoup4 in /usr/local/lib/python3.7/dist-
packages (4.6.3)
```

2.2 Use the "requests" library to download the necessary webpage from the SEC server.

[3]:
```
import requests

# This header information MUST be submitted with your URL reuqest to the SEC␣
↪website.
```

```
# If this header is not included or is incorrect, then you will not be abkle to␣
↪download the data (it will be rejected by the SEC server).

heads = {'Host': 'www.sec.gov', 'Connection': 'close',
         'Accept': 'application/json, text/javascript, */*; q=0.01',␣
↪'X-Requested-With': 'XMLHttpRequest',
         'User-Agent': 'ji97@bu.edu'
         }

URL = 'https://www.sec.gov/Archives/edgar/data/320193/000032019321000105/R2.
↪htm' # Find the specific URL that I need to scrape

# Get data from webpage
page = requests.get(URL, headers=heads)
```

2.3 Use "Beautiful Soup" Library to implement an "html/xml parser"

Here is the link to the documentation: https://www.crummy.com/software/BeautifulSoup/bs4/doc/

In this example we will download the Income Statement for Apple Inc. (AAPL) from the sec website.

[4]:
```
from bs4 import BeautifulSoup

soup = BeautifulSoup(page.content, 'lxml')
```

[ ]:
```
# The Prettify function formats that html/xml text to make it easier to read.
# You would inspect the html code to find marker tags that identifies the␣
↪specific section of text you wish to extract.
print(soup.prettify())
```

2.4 The Next Step is to Find the HTML Tags

- The tag ***<td class="nump">*** precedes any reported numbers AAPL's financial statements

[6]:
```
# Find the tag that identify numbers from Apple's Income Statement
# Extract numbers from AAPL's financial statements using HTML marker: <td␣
↪class="nump">
Numbers = soup.find_all ('td', class_='nump')
```

2.5 Print out the scraped Numbers that had the class tag "nump" from Apple's Income Statement

```
[7]: print(Numbers)
```

```
[<td class="nump">$ 365,817<span></span>
</td>, <td class="nump">$ 274,515<span></span>
</td>, <td class="nump">$ 260,174<span></span>
</td>, <td class="nump">212,981<span></span>
</td>, <td class="nump">169,559<span></span>
</td>, <td class="nump">161,782<span></span>
</td>, <td class="nump">152,836<span></span>
</td>, <td class="nump">104,956<span></span>
</td>, <td class="nump">98,392<span></span>
</td>, <td class="nump">21,914<span></span>
</td>, <td class="nump">18,752<span></span>
</td>, <td class="nump">16,217<span></span>
</td>, <td class="nump">21,973<span></span>
</td>, <td class="nump">19,916<span></span>
</td>, <td class="nump">18,245<span></span>
</td>, <td class="nump">43,887<span></span>
</td>, <td class="nump">38,668<span></span>
</td>, <td class="nump">34,462<span></span>
</td>, <td class="nump">108,949<span></span>
</td>, <td class="nump">66,288<span></span>
</td>, <td class="nump">63,930<span></span>
</td>, <td class="nump">258<span></span>
</td>, <td class="nump">803<span></span>
</td>, <td class="nump">1,807<span></span>
</td>, <td class="nump">109,207<span></span>
</td>, <td class="nump">67,091<span></span>
</td>, <td class="nump">65,737<span></span>
</td>, <td class="nump">14,527<span></span>
</td>, <td class="nump">9,680<span></span>
</td>, <td class="nump">10,481<span></span>
</td>, <td class="nump">$ 94,680<span></span>
</td>, <td class="nump">$ 57,411<span></span>
</td>, <td class="nump">$ 55,256<span></span>
</td>, <td class="nump">$ 5.67<span></span>
</td>, <td class="nump">$ 3.31<span></span>
</td>, <td class="nump">$ 2.99<span></span>
</td>, <td class="nump">$ 5.61<span></span>
</td>, <td class="nump">$ 3.28<span></span>
</td>, <td class="nump">$ 2.97<span></span>
</td>, <td class="nump">16,701,272<span></span>
</td>, <td class="nump">17,352,119<span></span>
</td>, <td class="nump">18,471,336<span></span>
</td>, <td class="nump">16,864,919<span></span>
</td>, <td class="nump">17,528,214<span></span>
</td>, <td class="nump">18,595,651<span></span>
</td>, <td class="nump">$ 297,392<span></span>
</td>, <td class="nump">$ 220,747<span></span>
</td>, <td class="nump">$ 213,883<span></span>
</td>, <td class="nump">192,266<span></span>
</td>, <td class="nump">151,286<span></span>
</td>, <td class="nump">144,996<span></span>
</td>, <td class="nump">68,425<span></span>
</td>, <td class="nump">53,768<span></span>
</td>, <td class="nump">46,291<span></span>
</td>, <td class="nump">$ 20,715<span></span>
</td>, <td class="nump">$ 18,273<span></span>
</td>, <td class="nump">$ 16,786<span></span>
</td>]
```

2.6 Clean & Merge Data and Return the Income Statement DataFrame

[8]:
```
# import regex library to extract only digit numbers from "the scraped Numbers"
import re
l = []
for i in Numbers[:]:
    x = re.findall("([0-9]+.[0-9]+)", str(i)) # return the string format e.g.␣
 ↪"7.8" or '887,99'
    if x == []: # nothing matched in the first two clauses
      x = re.findall("([0-9]+)", str(i)) # return the string format e.g. "788"

    try:
      x = int(x[0].replace(',','')) # for x, replace the ',' with '' and␣
 ↪convert 'string' into 'int'
    except:
      x = float(x[0]) # otherwise, for x, convert 'string' to 'floating point'
    l.append(x) # store each number into a list 'l'



# import numpy
import numpy as np
Num = np.round(np.array(l).reshape(int(len(l)/3),3),2) # convert the list 'l'␣
 ↪into numpy array 'Num', reshape the array 'Num' into the dimension: 24 X 3,␣
 ↪round all numbers to 2 decimal places
Num = np.insert(Num, [3,11,13,15,17], np.nan, axis=0 ) # insert 5 rows of Nan␣
 ↪values before the row index 3,11,13,15,17



# Extract the income statement items using the HTML Tag <a class="a">
```

```
coln = []
for i in range(len(soup.find_all ('a', class_='a'))):
  name = str(soup.find_all ('a', class_='a')[i])
  name = re.findall('>.+?<',name)[0] # return the string format e.g. ">ABc<"
  name = re.sub('[<,>]','', name) # replace the '<' , ',' , '>' with  ''
  name = re.sub('strong','', name) # replace the 'strong' with  ''
  coln.append(name) # store each name into a list 'coln'



coln = np.array(coln).reshape(24,1) # reshape the coln name into the dimension:␣
 ↪24 X 1
coln_Num = np.concatenate((coln, Num), axis=1) # concat 'coln' and 'Num'


# Extract the income statement headers using the HTML Tag   <th class="th">
header = []
for i in range(len(soup.find_all('th', class_='th'))):
  name = str(soup.find_all('th', class_='th')[i])
  name = re.findall('>.+?<',name)[0] # return the string format e.g. ">ABc<"
  name = re.sub('[<,>]','', name) # replace the '<' , ',' , '>' with  ''
  name = re.sub('div','',name) # replace the 'div' with  ''
  header.append(name) # store each name into a list 'header'
header.pop(0) # exclude the first element '12 Months Ended'
header.insert(0,'Income Statement Item for Apple') # insert the 'Income␣
 ↪Statement Item for Apple' at the index 0


import pandas as pd
df_apple = pd.DataFrame(coln_Num, columns = header) # Convert the income␣
 ↪statement into DataFrame
print(df_apple)
```

```
                  Income Statement Item for Apple Sep. 25 2021 Sep. 26 2020  \
0                                       Net sales     365817.0     274515.0
1                                   Cost of sales     212981.0     169559.0
2                                    Gross margin     152836.0     104956.0
3                             Operating expenses:          nan          nan
4                        Research and development      21914.0      18752.0
5             Selling general and administrative      21973.0      19916.0
6                        Total operating expenses      43887.0      38668.0
7                                Operating income     108949.0      66288.0
8                        Other income/(expense) net      258.0        803.0
9        Income before provision for income taxes     109207.0      67091.0
10                     Provision for income taxes      14527.0       9680.0
11                                     Net income      94680.0      57411.0
12                            Earnings per share:          nan          nan
```

```
13                  Basic (in dollars per share)         5.67         3.31
14                Diluted (in dollars per share)         5.61         3.28
15  Shares used in computing earnings per share:          nan          nan
16                            Basic (in shares)       16701.0      17352.0
17                          Diluted (in shares)       16864.0      17528.0
18                                     Products           nan          nan
19                                    Net sales      297392.0     220747.0
20                                Cost of sales      192266.0     151286.0
21                                     Services           nan          nan
22                                    Net sales       68425.0      53768.0
23                                Cost of sales       20715.0      18273.0

   Sep. 28 2019
0      260174.0
1      161782.0
2       98392.0
3           nan
4       16217.0
5       18245.0
6       34462.0
7       63930.0
8        1807.0
9       65737.0
10      10481.0
11      55256.0
12          nan
13         2.99
14         2.97
15          nan
16      18471.0
17      18595.0
18          nan
19     213883.0
20     144996.0
21          nan
22      46291.0
23      16786.0
```

3 Income Statement data for Microsoft (MSFT) for the years 2021, 2020 and 2019.

- Consolidated Statement of Operations
- The SEC URL for this information for Microsoft is: https://www.sec.gov/Archives/edgar/data/789019/000156459021039151/R2.htm

3.1 Install required libraries

[9]:
```
# Install necessary Python libraries
! pip install requests
! pip install beautifulsoup4
```

```
Requirement already satisfied: requests in /usr/local/lib/python3.7/dist-
packages (2.23.0)
Requirement already satisfied: certifi>=2017.4.17 in
/usr/local/lib/python3.7/dist-packages (from requests) (2021.10.8)
Requirement already satisfied: urllib3!=1.25.0,!=1.25.1,<1.26,>=1.21.1 in
/usr/local/lib/python3.7/dist-packages (from requests) (1.24.3)
Requirement already satisfied: chardet<4,>=3.0.2 in
/usr/local/lib/python3.7/dist-packages (from requests) (3.0.4)
Requirement already satisfied: idna<3,>=2.5 in /usr/local/lib/python3.7/dist-
packages (from requests) (2.10)
Requirement already satisfied: beautifulsoup4 in /usr/local/lib/python3.7/dist-
packages (4.6.3)
```

3.2 Use the ”requests” library to download the necessary webpage from the SEC server.

[10]:
```
import requests

# This header information MUST be submitted with your URL reuqest to the SEC␣
↪website.
# If this header is not included or is incorrect, then you will not be abkle to␣
↪download the data (it will be rejected by the SEC server).

heads = {'Host': 'www.sec.gov', 'Connection': 'close',
         'Accept': 'application/json, text/javascript, */*; q=0.01',␣
↪'X-Requested-With': 'XMLHttpRequest',
         'User-Agent': 'ji97@bu.edu'
         }

URL = 'https://www.sec.gov/Archives/edgar/data/789019/000156459021039151/R2.
↪htm' # Find the specific URL that I need to scrape


# Get data from webpage
page = requests.get(URL, headers=heads)
```

3.3 Use ”Beautiful Soup” Library to implement an ”html/xml parser”

Here is the link to the documentation: https://www.crummy.com/software/BeautifulSoup/bs4/doc/

In this example we will download the Income Statement for Microsoft (MSFT) from the sec website.

[11]:
```
from bs4 import BeautifulSoup

soup = BeautifulSoup(page.content, 'lxml')
```

[ ]:
```
# The Prettify function formats that html/xml text to make it easier to read.
# You would inspect the html code to find marker tags that identifies the␣
 ↪specific section of text you wish to extract.
print(soup.prettify())
```

3.4 The Next Step is to Find the HTML Tags

- The tag ***<td class="nump">*** precedes any reported numbers Microsoft's financial statements

[13]:
```
# Find the tag that identify numbers from Microsoft's Income Statement
# Extract numbers from Microsoft's financial statements using HTML marker: <td␣
 ↪class="nump">
Numbers = soup.find_all ('td', class_='nump')
```

3.5 Print out the scraped Numbers that had the class tag "nump" from Microsoft's Income Statement

[14]:
```
print(Numbers)
```

```
[<td class="nump">$ 168,088<span></span>
</td>, <td class="nump">$ 143,015<span></span>
</td>, <td class="nump">$ 125,843<span></span>
</td>, <td class="nump">52,232<span></span>
</td>, <td class="nump">46,078<span></span>
</td>, <td class="nump">42,910<span></span>
</td>, <td class="nump">115,856<span></span>
</td>, <td class="nump">96,937<span></span>
</td>, <td class="nump">82,933<span></span>
</td>, <td class="nump">20,716<span></span>
</td>, <td class="nump">19,269<span></span>
</td>, <td class="nump">16,876<span></span>
</td>, <td class="nump">20,117<span></span>
</td>, <td class="nump">19,598<span></span>
</td>, <td class="nump">18,213<span></span>
</td>, <td class="nump">5,107<span></span>
</td>, <td class="nump">5,111<span></span>
</td>, <td class="nump">4,885<span></span>
</td>, <td class="nump">69,916<span></span>
</td>, <td class="nump">52,959<span></span>
</td>, <td class="nump">42,959<span></span>
</td>, <td class="nump">1,186<span></span>
</td>, <td class="nump">77<span></span>
</td>, <td class="nump">729<span></span>
</td>, <td class="nump">71,102<span></span>
</td>, <td class="nump">53,036<span></span>
</td>, <td class="nump">43,688<span></span>
</td>, <td class="nump">9,831<span></span>
</td>, <td class="nump">8,755<span></span>
</td>, <td class="nump">4,448<span></span>
</td>, <td class="nump">$ 61,271<span></span>
</td>, <td class="nump">$ 44,281<span></span>
</td>, <td class="nump">$ 39,240<span></span>
</td>, <td class="nump">$ 8.12<span></span>
</td>, <td class="nump">$ 5.82<span></span>
</td>, <td class="nump">$ 5.11<span></span>
</td>, <td class="nump">$ 8.05<span></span>
</td>, <td class="nump">$ 5.76<span></span>
</td>, <td class="nump">$ 5.06<span></span>
</td>, <td class="nump">7,547<span></span>
</td>, <td class="nump">7,610<span></span>
</td>, <td class="nump">7,673<span></span>
</td>, <td class="nump">7,608<span></span>
</td>, <td class="nump">7,683<span></span>
</td>, <td class="nump">7,753<span></span>
</td>, <td class="nump">$ 71,074<span></span>
</td>, <td class="nump">$ 68,041<span></span>
</td>, <td class="nump">$ 66,069<span></span>
</td>, <td class="nump">18,219<span></span>
</td>, <td class="nump">16,017<span></span>
</td>, <td class="nump">16,273<span></span>
</td>, <td class="nump">97,014<span></span>
</td>, <td class="nump">74,974<span></span>
</td>, <td class="nump">59,774<span></span>
</td>, <td class="nump">$ 34,013<span></span>
</td>, <td class="nump">$ 30,061<span></span>
</td>, <td class="nump">$ 26,637<span></span>
</td>]
```

3.6 Clean & Merge Data and Return the Income Statement DataFrame

[15]:
```python
# import regex library to extract only digit numbers from "the scraped Numbers"
import re
l = []
for i in Numbers[:]:
    x = re.findall("([0-9]+.[0-9]+)", str(i)) # return the string format e.g.␣
↪"7.8" or '887,99'
    if x == []: # nothing matched in the first two clauses
      x = re.findall("([0-9]+)", str(i)) # return the string format e.g. "788"

    try:
      x = int(x[0].replace(',','')) # for x, replace the ',' with '' and␣
↪convert 'string' into 'int'
    except:
      x = float(x[0]) # otherwise, for x, convert 'string' to 'floating point'
    l.append(x) # store each number into a list 'l'


# import numpy
import numpy as np
Num = np.round(np.array(l).reshape(int(len(l)/3),3),2) # convert the list 'l'␣
↪into numpy array 'Num', reshape the array 'Num' into the dimension: 23 X 3,␣
↪round all numbers to 2 decimal places
Num = np.insert(Num, [11,13,15,17], np.nan, axis=0 ) # insert 4 rows of Nan␣
↪values before the row index 11,13,15,17


# Extract the income statement items using the HTML Tag <a class="a">
coln = []
for i in range(len(soup.find_all ('a', class_='a'))):
  name = str(soup.find_all ('a', class_='a')[i])
  name = re.findall('>.+?<',name)[0] # return the string format e.g. ">ABc<"
  name = re.sub('[<,>]','', name) # replace the '<' , ',' , '>' with  ''
  name = re.sub('strong','', name) # replace the 'strong' with  ''
  coln.append(name) # store each name into a list 'coln'



coln = np.array(coln).reshape(23,1) # reshape the coln name into the dimension:␣
↪23 X 1
coln_Num = np.concatenate((coln, Num), axis=1) # concat 'coln' and 'Num'


# Extract the income statement headers using the HTML Tag   <th class="th">
header = []
for i in range(len(soup.find_all('th', class_='th'))):
  name = str(soup.find_all('th', class_='th')[i])
  name = re.findall('>.+?<',name)[0] # return the string format e.g. ">ABc<"
  name = re.sub('[<,>]','', name) # replace the '<' , ',' , '>' with  ''
  name = re.sub('div','',name) # replace the 'div' with  ''
  header.append(name) # store each name into a list 'header'
header.pop(0) # exclude the first element '12 Months Ended'
header.insert(0,'Income Statement Item for Microsoft') # insert the 'Income␣
↪Statement Item for Microsoft' at the index 0


import pandas as pd
df_micros = pd.DataFrame(coln_Num, columns = header) # Convert the income␣
↪statement into DataFrame
display(df_micros)
```

```
    Income Statement Item for Microsoft Jun. 30 2021 Jun. 30 2020  \
0                               Revenue     168088.0     143015.0
1                       Cost of revenue      52232.0      46078.0
2                          Gross margin     115856.0      96937.0
3              Research and development      20716.0      19269.0
4                   Sales and marketing      20117.0      19598.0
5            General and administrative       5107.0       5111.0
6                      Operating income      69916.0      52959.0
7                     Other income net        1186.0         77.0
8            Income before income taxes      71102.0      53036.0
9            Provision for income taxes       9831.0       8755.0
10                           Net income      61271.0      44281.0
11                  Earnings per share:          nan          nan
12                                Basic         8.12         5.82
13                              Diluted         8.05         5.76
14  Weighted average shares outstanding:         nan          nan
15                                Basic       7547.0       7610.0
16                              Diluted       7608.0       7683.0
17                              Product          nan          nan
18                              Revenue      71074.0      68041.0
19                      Cost of revenue      18219.0      16017.0
20                    Service and Other          nan          nan
21                              Revenue      97014.0      74974.0
22                      Cost of revenue      34013.0      30061.0

   Jun. 30 2019
0      125843.0
1       42910.0
2       82933.0
3       16876.0
4       18213.0
5        4885.0
6       42959.0
7         729.0
8       43688.0
9        4448.0
10      39240.0
11          nan
```

```
12          5.11
13          5.06
14           nan
15        7673.0
16        7753.0
17           nan
18       66069.0
19       16273.0
20           nan
21       59774.0
22       26637.0
```

[21]:
```
from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call
drive.mount("/content/drive", force_remount=True).
```

[ ]:
```
!sudo apt-get install texlive-xetex texlive-fonts-recommended␣
↪texlive-plain-generic
```

[24]:
```
!jupyter nbconvert --to pdf '/content/drive/MyDrive/BA_870/HW/1/
↪BA870_Assignment #1_Ji_Qi.ipynb'
```

```
[NbConvertApp] Converting notebook
/content/drive/MyDrive/BA_870/HW/1/BA870_Assignment #1_Ji_Qi.ipynb to pdf
[NbConvertApp] Writing 334828 bytes to ./notebook.tex
[NbConvertApp] Building PDF
[NbConvertApp] Running xelatex 3 times: ['xelatex', './notebook.tex', '-quiet']
[NbConvertApp] Running bibtex 1 time: ['bibtex', './notebook']
[NbConvertApp] WARNING | bibtex had problems, most likely because there were no
citations
[NbConvertApp] PDF successfully created
[NbConvertApp] Writing 228819 bytes to
/content/drive/MyDrive/BA_870/HW/1/BA870_Assignment #1_Ji_Qi.pdf
```

[17]:
```

```