CSC 4800 Python Applications Programming

HW/Lab #4 – RegEx: Stock Quote XML

Due: Wednesday, February 1, 2017

Design and implement a Python 3.x program "getQuotes.py" that accesses the Yahoo Finance website to obtain the latest stock price information for some stock symbols, and prints out the resulting information in an XML format.

Retrieving latest stock price data for a stock symbol:

The Yahoo Finance website provides a data feed that can be used to acquire stock quote information about one or more stock symbols. As an example, if you enter the following URL in your web browser:

http://finance.yahoo.com/d/quotes.csv?f=sd1t1l1bawmc1vj2&e=.csv&s=MSFT

it will respond with a line of text data similar to (with everything on one line):

```
"MSFT","11/15/2006","10:19am",29.32,29.31,29.32,"21.46 - 29.46",
          "29.11 - 29.33",+0.0915,11550079,  9,830,459,000
```

The exact data values returned are controlled by the 'f=. . .' setting in the URL:

f=sd1t1l1bawmc1vj2	**s**	**d1**	**t1**	**l1**	**b**	**a**	**w**	**m**	**c1**	**v**	**j2**
	ess	dee-one	tee-one	ell-one					cee-one		jay-two

and for this lab exercise your program must use the lower-case flags shown above.

In Python, you can import the urllib module and define a function that will retrieve the text data line using the urlopen( ) method and the readlines( ) method. For example, if you call

ProcessQuotes("&s=MSFT")

the following will simply print out the text data line that is retrieved from Yahoo Finance:

---

import urllib, re, string

```
def ProcessQuotes(strSyms):
      strUrl='http://finance.yahoo.com/d/quotes.csv?f=sd1t1l1bawmc1vj2&e=.csv'
      strUrl = strUrl + strSyms
      try:
            f = urllib.urlopen(strUrl)
      except:
            #something failed... couldn't open url?
            print "URL access failed:\n" + strUrl + "\n"
            return

      for line in f.readlines():
            line = line.decode().strip(); # convert byte array to string
            print line + '\n'
```

Although not needed in this lab, Yahoo Finance actually allows the retrieval of stock data for several symbols, using notations like

ProcessQuotes('&s=MSFT&s=GE&s=KKD')

or

ProcessQuotes('&s=MSFT+GE+KKD')

and the return data will have one line for each valid symbol in the list. Yahoo Finance permits up to 10 symbols to be specified in a single URL request.

Main program:

Your getQuotes.py program must include a main program that loops repeatedly, each time inputting from the user a stock symbol name and producing the corresponding <XML> output. The program can terminate when the user presses ENTER without entering a symbol name. For example:

```
while True:
      print("Enter a Stock Symbol: ")
      sym = sys.stdin.readline().strip()
      if(len(sym) == 0):
            break
      strSyms = '&s=' + sym
      ProcessQuotes(strSyms)
```

In ProcessQuotes( ), your program must retrieve and print out the raw quote data line, and then must analyze and parse the stock quote data line, and output text using an XML-style tagged format, e.g.,

---

```
"MSFT","1/24/2017","4:00pm",63.52,63.53,63.59,"48.03 - 64.10",
            "62.94 - 63.74",+0.56,24650457,  7,775,350,000

<stockquote>
      <qSymbol>MSFT</qSymbol>
      <qDate>1/24/2017</qDate>
      <qTime>4:00pm</qTime>
      <qLastSalePrice>63.52</qLastSalePrice>
      <qBidPrice>63.53</qBidPrice>
      <qAskPrice>63.59</qAskPrice>
      <q52WeekLow>48.03</q52WeekLow>
      <q52WeekHigh>64.10</q52WeekHigh>
      <qTodaysLow>62.94</qTodaysLow>
      <qTodaysHigh>63.74</qTodaysHigh>
      <qNetChangePrice>+0.56</qNetChangePrice>
      <qShareVolumeQty>24650457</qShareVolumeQty>
      <qTotalOutstandingSharesQty>7775350000</qTotalOutstandingSharesQty>
</stockQuote>
```

The various <tag-names> in your generated XML output must match the ones above. If you look carefully, you'll notice several details about the actual data values:

- the surrounding " . . . " quotes have been removed.
- the q52Week Low and High values (if any) and the qTodays Low and High values (if any) have been extracted from components of their embedded strings.
- the last value on the data line (qTotalOutstandingSharesQty) is a potentially large numeric value that may (or may not) initially have whitespace and embedded commas; the <XML> field value has any spaces and commas removed.
- Some stock symbols do not have data values for all of the fields, with missing fields (e.g. N/A).
  e.g., "kkd","7/27/2016","3:00pm",21.00,N/A,N/A,"12.90 - 21.75",N/A,+0.00,0,N/A
  Do not include these 'N/A' or missing fields in the generated XML output.

For this lab, you must use a RegEx regular expression pattern match text-analysis implementation to do most of the work of extracting out the various data fields. Individual string operations are OK if needed for additional processing of some of the more complex data fields.

Turn in printed assignment in class.

1) Print out of source code for all your modules for this assignment.
2) Printout of the output of a sample execution showing the processing for 2 stock symbols.