"""

Charlie Ang

CSC 4800 Python Applications Programming

Lab # 4

Dr. Tindall

January 31, 2017

This program 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.

"""

import urllib.request,re, string

def ProcessQuotes(strSyms):

"""

Retrieve text data line using urlopen() and readlines() method and

processes the text data line using regular expressions to

print out the text data line that is retrieved from Yahoo Finance.

:param strSyms: parameter to concatenate stock symbol to url

:return:

"""

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

strUrl = strUrl + strSyms

try:

f = urllib.request.urlopen(strUrl)

except:

# Something failed. Could not 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')

# symbol is not a valid stock symbol

symbol = strSyms.replace('&s=', '')

urlRetrieved = "\"" + symbol + "\"" + ",N/A,N/A,N/A,N/A,N/A,N/A,N/A,N/A,N/A,N/A"

if line == urlRetrieved:

print('Unknown stock symbol: match failed')

print('')

return

# sample line

# "MSFT","1/31/2017","4:00pm",64.65,64.66,64.75,"48.03 - 65.91","64.26 - 65.15",

# "-0.48 - -0.74%","4:00pm - <b>64.65</b>",25270549,7727529000

print("<stockquote>")

# qSymbol

qSymbol = re.search("([a-zA-Z]+)", line).group(1)

if qSymbol is not None:

print("\t<qSymbol>" + qSymbol + "</qSymbol>")

# qDate

if re.search("\d+/\d+/\d+", line) is None:

pass

else:

qDate = re.search("\d+/\d+/\d+", line).group()

print("\t<qDate>" + qDate + "</qDate>")

# qTime

if re.search("\d+:\d\d[a-z]{2}", line) is None:

pass

else:

qTime = re.search("(\d+:\d\d[a-z]{2})", line).group(1)

print("\t<qTime>" + qTime + "</qTime>")

# qLastSalePrice

if re.search("(?:.\*?),(\d+\.\d+),", line) is None:

pass

else:

qLastSalePrice = re.search("(?:.\*?),(\d+\.\d+),", line).group(1)

if qLastSalePrice is not None:

print("\t<qLastSalePrice>" + qLastSalePrice + "</qLastSalePrice>")

# qBidPrice

if re.match("(?:.\*?),(?:\d+\.\d+),(\d+\.\d+)", line) is None:

pass

else:

qBidPrice = re.match("(?:.\*?),(?:\d+\.\d+),(\d+\.\d+)", line).group(1)

if qBidPrice is not None:

print("\t<qBidPrice>" + qBidPrice + "</qBidPrice>")

# qAskPrice

if re.search(",(\d+\.\d+),\"", line) is None:

pass

else:

qAskPrice = re.search(",(\d+\.\d+),\"", line).group(1)

if qAskPrice is not None:

print("\t<qAskPrice>" + qAskPrice + "</qAskPrice>")

#q52WeekLow

if re.search("(\d+\.\d+\s-\s\d+\.\d+)", line) is None:

pass

else:

q52Week = re.search("(\d+\.\d+)\s-\s(\d+\.\d+)", line)

q52WeekLow = q52Week.group(1)

q52WeekHigh = q52Week.group(2)

if q52WeekLow is not None:

print("\t<q52WeekLow>" + q52WeekLow + "</q52WeekLow>")

# q52WeekHigh

if q52WeekHigh is not None:

print("\t<q52WeekHigh>" + q52WeekHigh + "</q52WeekHigh>")

# qTodaysLow

if re.search("(?:\d+\.\d+\s-\s\d+\.\d+)\",\"(\d+\.\d+)\s-", line) is None:

pass

else:

qTodaysLow = re.search("(?:\d+\.\d+\s-\s\d+\.\d+)\",\"(\d+\.\d+)\s-", line).group(1)

if qTodaysLow is not None:

print("\t<qTodaysLow>" + qTodaysLow + "</qTodaysLow>")

# qTodaysHigh

if re.search("(?:\d+\.\d+\s-\s\d+\.\d+)\",\"(?:\d+\.\d+)\s-\s(\d+\.\d+)\"", line) is None:

pass

else:

qTodaysHigh = re.search("(?:\d+\.\d+\s-\s\d+\.\d+)\",\"(?:\d+\.\d+)\s-\s(\d+\.\d+)\"", line).group(1)

if qTodaysHigh is not None:

print("\t<qTodaysHigh>" + qTodaysHigh + "</qTodaysHigh>")

# qNetChangePrice

if re.search("([\+|-]\d+\.\d+)", line) is None:

pass

else:

qNetChangePrice = re.search("([\+|-]\d+\.\d+)", line).group(1)

if qNetChangePrice is not None:

print("\t<qNetChangePrice>" + qNetChangePrice + "</qNetChangePrice>")

# qShareVolumeQty

if re.search("(?:\w+),(\d+),(\w+)", line) is None:

pass

else:

pattern = re.search("(?:\w+),(\d+),(\w+)", line)

qShareVolumeQty = pattern.group(1)

#qTotalOutstandingSharesQty = pattern.group(2)

if qShareVolumeQty is not None:

print("\t<qShareVolumeQty>" + qShareVolumeQty + "</qShareVolumeQty>")

# qTotalOutstandingSharesQty

if re.search("(?:\w+),(\w+),(\d+)", line) is None:

pass

else:

pattern = re.search("(?:\w+),(\w+),(\d+)", line)

qTotalOutstandingSharesQty = pattern.group(2)

# testing removing spaces and commas

#qTEST = " 7,775,350,000"

#qTEST = qTEST.replace(',', '').strip()

#print("\t<qTEST>" + qTEST + "</qTEST>")

# remove commas and whitespaces

qTotalOutstandingSharesQty = qTotalOutstandingSharesQty.replace(',', '').strip()

if qTotalOutstandingSharesQty is not None:

print("\t<qTotalOutstandingSharesQty>" + qTotalOutstandingSharesQty + "</qTotalOutstandingSharesQty>")

print("</stockquote>")

print('')

def main():

"""

Prompts user to input a stock symbol name and produces the corresponding

<XMl> output. Program terminates when user presses ENTER without entering a

symbol name.

:return:

"""

while True:

userInput = input("Enter a Stock Symbol: ") # prompts user for input

if(len(userInput) == 0): # terminates when user doesn't enter a symbol

break

strSyms = '&s=' + userInput # add '&s' to retrieve stock symbol

ProcessQuotes(strSyms) # calls function to process quotes

if \_\_name\_\_ == '\_\_main\_\_':

main()