1. Options data recording.
   Read the Pandas documentation on how to retrieve options price data from Yahoo Finance using Pandas method and make the following code to work.
   In particular, you need to get the line after # NOTE: to work properly.

   ###*** beginning of python code ****

```
"""
Homework problem 1.
"""
import datetime as dt
import pandas as pd
import numpy as np
from pandas.io.data import Options
import h5py as h5

# ticker_list.csv contains a list of tickers on nyse and nasdaq stock exchanges
tickers = pd.Series.from_csv('ticker_list.csv')
num_ticks = tickers.size  # used to print status

months = {1: 'Jan', 2: 'Feb', 3: 'Mar', 4: 'Apr', 5: 'May', 6: 'Jun',
        7: 'Jul', 8: 'Aug', 9: 'Sep', 10: 'Oct', 11: 'Nov', 12: 'Dec'}

now = dt.datetime.now()  # Get current time
c_month = months[now.month]  # Get current month
c_day = str(now.day)  # Get current day
c_year = str(now.year)  # Get current year

f = h5.File('./options_db.h5')  # open database file
year = f.require_group(c_year)  # Make hdf5 group for year
month = year.require_group(c_month)  # Make hdf5 group for month
day = month.require_group(c_day)  # Make hdf5 group for day

num = 0
for i in tickers:

   option = Options(i,'yahoo')
   # NOTE: the following line needs to be corrected  to retrieve the options price data
   raw_puts, raw_calls = option.get_options_data(expiry=PANDAS_DATE)

   #raw_calls = option.get_forward_data(months=3, call=1, put=0,
   #                              near=1, above_below=6)
```

```
#raw_puts = option.get_forward_data(months=3, call=0, put=1,
#                                   near=1, above_below=6)

if raw_calls.values.any():  # Check if any calls were returned
   try:  # Try to add item to file.
      # This block (and below for puts) does the following:
      #   - Get unique expiry dates
      #   - make hdf5 group for ticker
      #   - Get options data for each expiry
      #   - Put each set of expiry data in unique hdf5 dataset

      expiries = raw_calls.Expiry.unique().astype(str)
      tick = day.require_group(i)

      for ex in expiries:
         data = raw_calls[raw_calls.Expiry == ex]
         i_calls = data[['Strike', 'Last', 'Vol']]
         i_calls.Vol = i_calls.Vol.str.replace(',', '')

         ex_m_y = ex[:2] + ex[-3:]
         call_ds = tick.require_dataset('C' + i + ex_m_y,
                                        i_calls.shape, float)
         call_ds[...] = i_calls.astype(np.float32)
   except:  # If it doesn't work just pass
      pass

if raw_puts.values.any():  # Check if any puts were returned
   try:
      expiries = raw_puts.Expiry.unique().astype(str)
      tick = day.require_group(i)

      for ex in expiries:
         data = raw_puts[raw_puts.Expiry == ex]
         i_puts = data[['Strike', 'Last', 'Vol']]
         i_puts.Vol = i_puts.Vol.str.replace(',', '')
         ex_m_y = ex[:2] + ex[-3:]
         put_ds = tick.require_dataset('P' + i + ex_m_y,
                                       i_puts.shape, float)
         put_ds[...] = i_puts.astype(np.float32)
   except:
      pass
```

```
    # status update
    num += 1
    if num % 500 == 0:
        print "just finished %s of %s" % (str(num), str(num_ticks))

f.close()  # Close file
###***** end of python code ****
```

Content of “ticker_list.csv”

```
aapl,
ibm,
spy
```

2. Real-time market price monitoring
   a. The following code monitors the pre-market trading prices of a single ticker AAPL starting at 9am but it fails after 9.30am EST (NYC time). Modify the code (or write a new function) that:
      a. It records the pre-market trading price every 60 seconds between 9am and 9.30am EDT
      b. It continues to record the regular trading prices every 60 seconds from 9:30am to 16:00
      c. The code records the after-hours trading prices every 60 seconds from 16:00 to 16:30.
   b. It is possible to get N=10 queries for N=10 stocks by calling, for example: NASDAQ:AAPL,NYSE:JNJ,… in line #7 of the code. Modify the program to fetch pre-market time-series, xi(t) (i=1,…,N), for N-asset portfolio. Given that, compute a fractional root-mean-square volatility, $\sigma xi(t)/\langle xi(t)\rangle$, i.e. standard deviation divided by the mean, between 6am and 9.30am EDT for each asset and check can you use it as an indicator for stock price movement after 9.30am? Tip: the higher market capitalization the firms the more trading are expected in first 15 min of a new session at Wall Street.

```
##*****  beginning of python code *****

import urllib2  # works fine with Python 2.7.9 (not 3.4.+)
import json
import time

import os, re, csv
```

```
        def  fetchPreMarket(symbol, exchange):
            link = "http://finance.google.com/finance/info?client=ig&q="
            url = link+"%s:%s" % (exchange, symbol)
            u = urllib2.urlopen(url)
            content = u.read()
            data = json.loads(content[3:])
            info = data[0]
            t = str(info["elt"])    # time stamp
            l = float(info["l"])    # close price (previous trading day)
            p = float(info["el"])   # stock price in pre-market (after-hours)
            return (t,l,p)

def fetchGF(googleticker):

    url="http://www.google.com/finance?&q="

    txt=urllib.urlopen(url+googleticker).read()

    k=re.search('id="ref_(.*?)">(.*?)<',txt)

    if k:

        tmp=k.group(2)

        q=tmp.replace(',','')

    else:

        q="Nothing found for: "+googleticker

    return q


def combine(ticker):

    quote=fetchGF(ticker) # use the core-engine function

    t=time.localtime()    # grasp the moment of time

    output=[t.tm_year,t.tm_mon,t.tm_mday,t.tm_hour,  # build a list

          t.tm_min,t.tm_sec,ticker,quote]

    return output
```

```
def getRTtickerQuote(ticker, t, fname, freq):
    with open(fname,'a') as f:
        writer=csv.writer(f,dialect="excel") #,delimiter=" ")
        while(t.tm_hour<=16):
          if(t.tm_hour==16):
            while(t.tm_min<01):
              data=combine(ticker)
              print(data)
              writer.writerow(data) # save data in the file
              time.sleep(freq)
            else:
              break
          else:
            for ticker in tickers:
              data=combine(ticker)
              print(data)
              writer.writerow(data) # save data in the file
              time.sleep(freq)
    f.close()


def getRTportQuote(ticker_list, t, fname, freq):
    with open(fname,'a') as f:
        writer = csv.writer(f,dialect="excel") #,delimiter=" ")
        print t.tm_hour
        while(t.tm_hour <= 9):
```

```
        if(t.tm_hour == 9):
          while(t.tm_min < 31):
             data = combine(ticker)
             print(data)
             writer.writerow(data) # save data in the file
             time.sleep(freq)
          else:break
        else:
          for ticker in ticker_list:
             data=combine(ticker)
             print(data)
             writer.writerow(data) # save data in the file
             time.sleep(freq)
    f.close()

if __name__ == "__main__":
   # display time corresponding to your location
   print(time.ctime())
  print
 # Set local time zone to NYC
 os.environ['TZ']='America/New_York'
t=time.localtime() # string
print(time.ctime())
print
p0 = 0
```

```
while True:
    t, l, p = fetchPreMarket("AAPL","NASDAQ")
     if(p!=p0):
        p0 = p
        print("%s\t%.2f\t%.2f\t%+.2f\t%+.2f%%" % (t, l, p, p-l,(p/l-1)*100.))
        time.sleep(60)
```