# Reference Solutions to Assignment 2

## Question 1.

Add the following functions in getstock.py:

```
def random_walks(annual_volatility, drift):
    n_minutes = 252 * 390
    start_price = 100.00
    volatility = annual_volatility / np.sqrt(252 * 390)
    moves = np.random.normal(0, volatility, n_minutes)
    ts = pd.Series(moves)
    ts = ts.add(1.0)
    ts = ts.add(drift)
    ts = ts.cumprod()
    ts = ts.multiply(start_price)
    ts_0 = pd.Series([start_price])
    ts = ts_0.append(ts, ignore_index=True)
    return ts


def plot_random_walks(n, ts, annual_volatility, drift):
    plt.subplot(2, 2, n)
    plt.ylim(75, 150)
    plt.title('volatility:'+str(annual_volatility)+', Drift: '+str(drift))
    plt.plot(ts, 'b-')


def cum_return(ts):
    cum_return = ts.diff().cumsum() / ts.loc[0]
    cum_return = cum_return.tail(1)
    return cum_return
```

Then run the main function as follows:

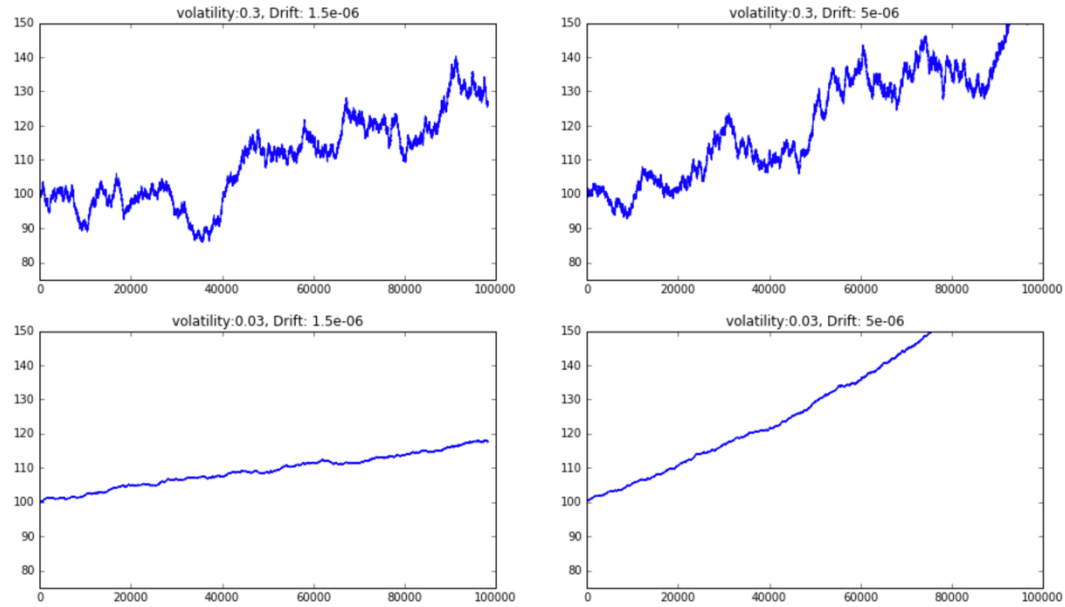
```
import numpy as np
import pandas as pd
import matplotlib
import matplotlib.pyplot as plt

%run getstock.py
```

```
<matplotlib.figure.Figure at 0x11cdcf780>
```

```
ts1 = random_walks(0.30, 1.5e-6)
ts2 = random_walks(0.30, 5e-6)
ts3 = random_walks(0.03, 1.5e-6)
ts4 = random_walks(0.03, 5e-6)
```

```
plt.figure(figsize=(16, 9))
plot_random_walks(1, ts1, 0.30, 1.5e-6)
plot_random_walks(2, ts2, 0.30, 5e-6)
plot_random_walks(3, ts3, 0.03, 1.5e-6)
plot_random_walks(4, ts4, 0.03, 5e-6)
```

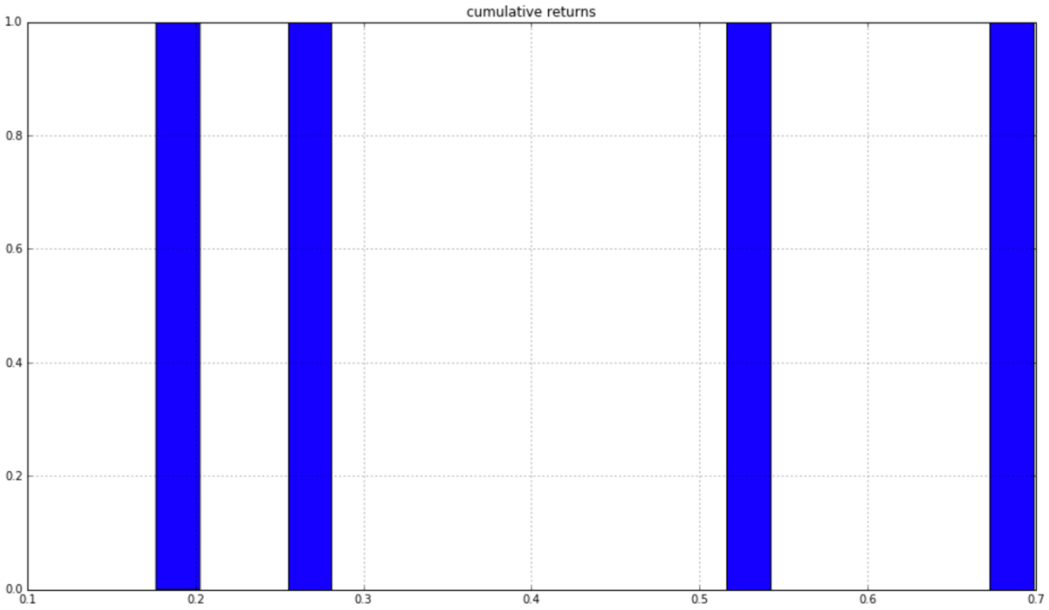



```
pre=[cum_return(ts1).iloc[0],cum_return(ts2).iloc[0],cum_return(ts3).iloc[0],cum_return(ts4).iloc[0]]
tot_returns=pd.DataFrame(pre,columns=['cumulative returns'])
tot_returns
```

| | cumulative returns |
|---|---|
| **0** | 0.270255 |
| **1** | 0.528454 |
| **2** | 0.176572 |
| **3** | 0.698919 |

```
tot_returns.hist(bins=20,figsize=(16,9))
```

```
array([[<matplotlib.axes._subplots.AxesSubplot object at 0x122918cc0>]],
      dtype=object)
```



Question 2.

(a)(b)(c):

1. Add the following functions in getstock.py:

```
def cal_macd(stock_data):
    ema_26 = stock_data['close'].ewm(span=26).mean()
    ema_12 = stock_data['close'].ewm(span=12).mean()
    macd = ema_12 - ema_26
    macd_9_ema = macd.ewm(span=9).mean()
    macd_diff = macd - macd_9_ema
    return ema_26, ema_12, macd, macd_9_ema, macd_diff
```

2. Use the following codes to substitute some codes in L2.6_MACD.ipynb:

(new codes)

```
import pandas as pd
import matplotlib.pyplot as plt
import matplotlib.dates as dates

%run getstock.py

matplotlib.rcParams["figure.figsize"] = (16,9)
```

```
raw_data = getMinuteStockPrices('AAPL')
raw_data.head()
```

| timestamp | open | high | low | close | volume |
|---|---|---|---|---|---|
| **2019-04-02 09:31:00** | 191.0900 | 191.5075 | 191.0500 | 191.35 | 446561 |
| **2019-04-02 09:32:00** | 191.3778 | 191.8200 | 191.0600 | 191.54 | 175058 |
| **2019-04-02 09:33:00** | 191.5400 | 192.1061 | 191.5400 | 192.09 | 280368 |
| **2019-04-02 09:34:00** | 192.0700 | 192.0900 | 191.7627 | 191.96 | 118646 |
| **2019-04-02 09:35:00** | 191.9800 | 192.2833 | 191.8600 | 192.22 | 145214 |

```
stock_data = raw_data['2019-04-04']
```

```
ema_26, ema_12, macd, macd_9_ema, macd_diff = cal_macd(stock_data)
```

(the codes you need to replace in L2.6_MACD.ipynb)

```
import pandas as pd
import matplotlib.pyplot as plt
import matplotlib.dates as dates
%run getstock.py

# set figsize...
plt.rcParams["figure.figsize"] = [16,16]
```

```
raw_data = getDailyStockPrices('TSLA')
```

```
stock_data = raw_data['2018-10-01':'2019-02-28']
```

```
# calculate EMAs

# 26 day
ema_26 = stock_data['close'].ewm(span=26).mean()

# 12 day
ema_12 = stock_data['close'].ewm(span=12).mean()

# difference between fast and slow averages
macd = ema_12 - ema_26

# calculate the 9 period EMA of the macd
macd_9_ema = macd.ewm(span=9).mean()

# calculate the difference the macd and the "signal line"
macd_diff = macd - macd_9_ema
```

3. Run the codes in L2.6_MACD.ipynb as follows:

```
from pandas.plotting import register_matplotlib_converters
register_matplotlib_converters()

from mpl_finance import candlestick_ohlc
fig, (ax1, ax2) = plt.subplots(2, 1, sharex=True)

candlestick_ohlc(ax1, zip(dates.date2num(stock_data.index.to_pydatetime()),
                          stock_data['open'], stock_data['high'],
                          stock_data['low'], stock_data['close']),
                          width=0.5, colorup='g', colordown='r')

ax1.plot(ema_12, "b-")
ax1.plot(ema_26, "r-")
ax1.legend(['OHLC', 'EMA_12', 'EMA_26'], loc="lower right")

ax2.plot(macd, "b-")
ax2.plot(macd_9_ema, "r-")
ax2.bar(macd.index, macd_diff)
ax2.legend(['macd', 'macd signal line', 'macd - signal line'], loc="lower right")

ax2.xaxis.set_major_locator(dates.DayLocator(interval=5))
ax2.xaxis.set_major_formatter(dates.DateFormatter('%m-%d-%y'))

fig.tight_layout()
fig.autofmt_xdate()
fig.suptitle("MACD")
```

```
def calc_signal(x):
    if x > 0:
        return 1
    elif x < 0:
        return -1
    else:
        return 0
```

```
signal = macd_diff.apply(calc_signal)
```

```
position = 0
trade_size = 1
pnl = 0
previous_signal = 0
start_price = 0

for index, row in stock_data.iterrows():
    current_signal = signal.loc[index]
    if current_signal != previous_signal:
        if position != 0:
            current_pnl = position * (row['open'] - start_price)
            pnl += current_pnl
            msg = "{0} Closing. signal: {1:d} current_price: {2:.2f} start_price: {3:.2f} pnl: {4:.2f}".format(index, curre
            print(msg)

        position = trade_size * current_signal
        start_price = row['open']
        msg = "{0} Opening. signal: {1:d} current_price: {2:.2f}".format(index, current_signal, row['open'])
        print(msg)
        previous_signal = current_signal

print("Final P&L: {0:.2f}".format(pnl))
```

## Hints for (d):

1. The time interval between two crossing points of MACD;
2. The relative location between two crossing points of MACD;
3. The difference between DIFF and DEA.