Reference Solutions to Assignment 3

Question 1 – 8.

Add the following functions in getstock.py:

```
def decorateAx(ax, xs, ys):
    def x_fmt_func(x, pos=None):
        idx = np.clip(int(x + 0.5), 0, len(xs) - 1)
        return xs[idx]
    idx_pxy = np.arange(len(xs))
    ax.plot(idx_pxy, ys, linewidth=1, linestyle='-')
    ax.xaxis.set_major_formatter(mtk.FuncFormatter(x_fmt_func))
    plt.xticks(rotation=45)


def plot_pov_simulation(n, fig, minute_data, trades):
    ax = fig.add_subplot(3, 3, n + 1)
    decorateAx(ax, minute_data.index, minute_data['close'])
    decorateAx(ax, trades.index, trades['price'])


def decorateAx_for_barchart(ax, xs, ys):
    def x_fmt_func(x, pos=None):
        idx = np.clip(int(x + 0.5), 0, len(xs) - 1)
        return xs[idx]
    idx_pxy = np.arange(len(xs))
    ax.bar(idx_pxy, ys)
    plt.xlim(idx_pxy[0], idx_pxy[-1])
    ax.xaxis.set_major_formatter(mtk.FuncFormatter(x_fmt_func))
    plt.xticks(rotation=45)


def plot_bar_chart(n, fig, trades):
    ax = fig.add_subplot(3, 3, n + 1)
    decorateAx_for_barchart(ax, trades.index, trades['shares'])
```

Then run the main function as follows:

```
%matplotlib inline

import matplotlib
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import io
import requests
from datetime import *
import matplotlib.ticker as mtk

%run getstock.py
```

```
minute_data = getMinuteStockPrices('LYFT')
minute_data.head()
```

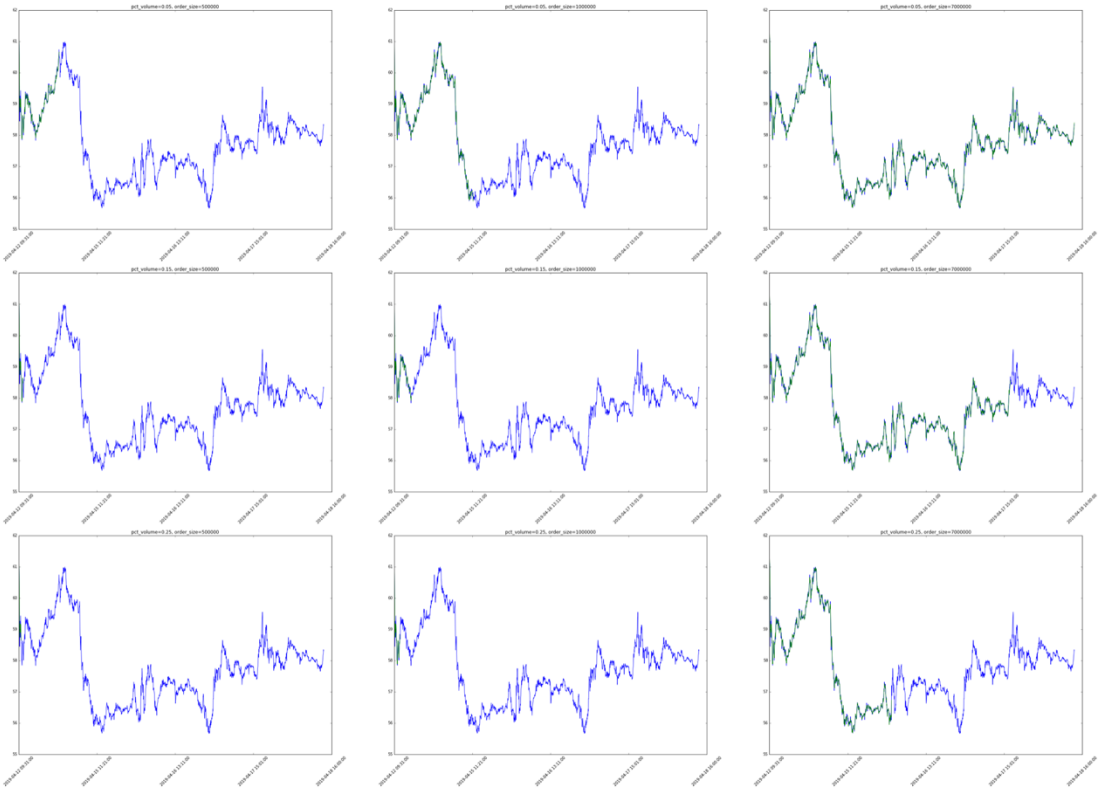
timestamp	open	high	low	close	volume
2019-04-12 09:31:00	61.38	61.4899	61.15	61.160	184054
2019-04-12 09:32:00	61.15	61.2300	61.10	61.100	28226
2019-04-12 09:33:00	61.13	61.1350	60.77	60.879	85347
2019-04-12 09:34:00	60.83	60.8538	60.16	60.310	94621
2019-04-12 09:35:00	60.36	60.4200	59.25	59.380	182748

```
def pov_strategy(ticker, order_size, pct_volume):
    quantity_remaining = order_size
    trades = pd.DataFrame(columns=['price', 'shares'], index=ticker.index)
    for index, row in ticker.iterrows():
        px = (row['high'] + row['low']) / 2
        volume = row['volume']
        new_trade = min(pct_volume * volume, quantity_remaining)
        trades.loc[index] = [px, new_trade]
        quantity_remaining -= new_trade
        if (quantity_remaining <= 0):
            break
    return trades
```

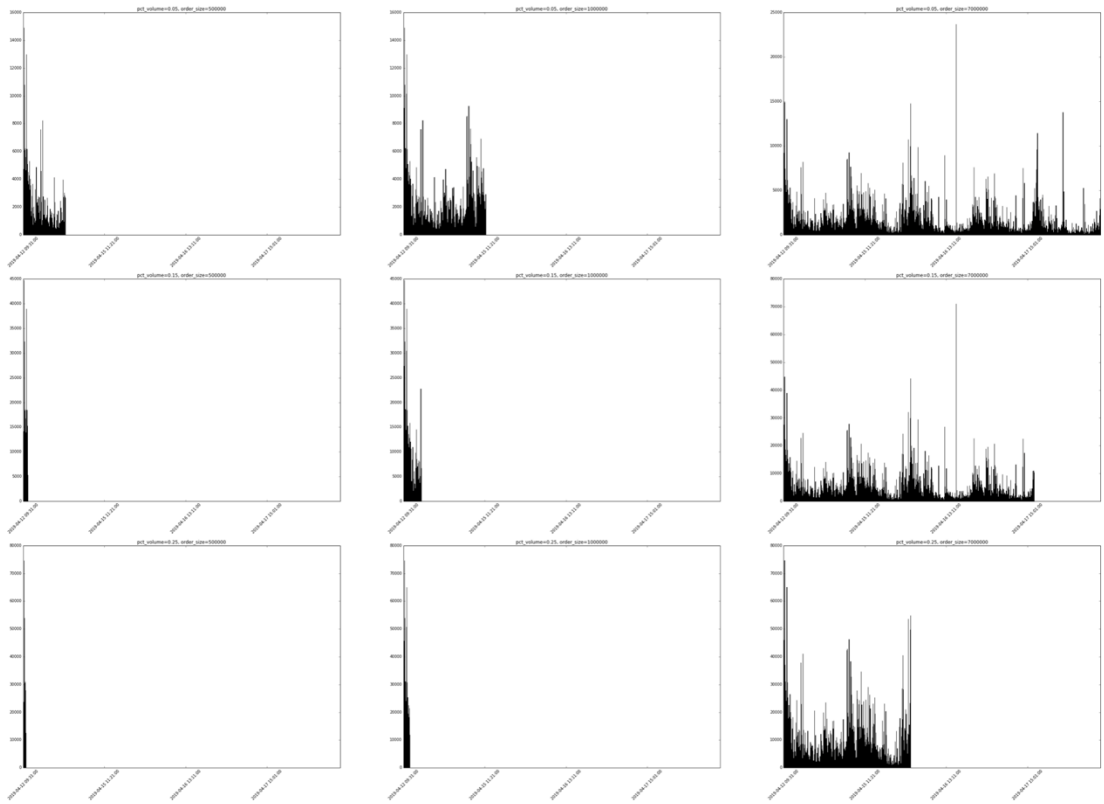
```
pct_volume = [0.05, 0.15, 0.25]
order_size = [500000, 1000000, 7000000]
```

```
trades_grid = []
titles = []
unexecuted_size = []
for i in pct_volume:
    for j in order_size:
        trades = pov_strategy(minute_data, j, i)
        trades_grid.append(trades)
        unexecuted_size.append(j - trades["shares"].sum())
        titles.append("pct_volume=" + str(i) + ", order_size=" + str(j))
```

```
price_fig = plt.figure(figsize=(50, 35))
for n in range(len(trades_grid)):
    plot_pov_simulation(n, price_fig, minute_data, trades_grid[n])
    plt.title(titles[n])
```



```
volume_fig = plt.figure(figsize=(50, 35))
for n in range(len(trades_grid)):
    plot_bar_chart(n, volume_fig, trades_grid[n])
    plt.title(titles[n])
```



```
dates = ['2019-04-12', '2019-04-15', '2019-04-16', '2019-04-17', '2019-04-18']
vwaps = [ (minute_data[date]['close'] * minute_data[date]['volume']).sum() / minute_data[date]['volume'].sum()
          for date in dates ]
pavgs = [ (trades_grid[i]['price'] * trades_grid[i]['shares']).sum() / trades_grid[i]['shares'].sum()
          for i in range(5) ]
slippages = [ (vwaps[i] - pavgs[i]) / vwaps[i] * 10000 for i in range(5) ]
stats_dict = {'VWAPs': vwaps, 'PAVGs': pavgs, 'Slippages': slippages}
stats = pd.DataFrame(stats_dict, index=dates)
display(stats)
print('5-day Average Slippage: ' + str(np.mean(slippages)))
```

	VWAPs	PAVGs	Slippages
2019-04-12	59.251697	58.937346	53.053634
2019-04-15	56.701569	58.591581	-333.326225
2019-04-16	57.055678	57.773063	-125.734256
2019-04-17	57.633019	58.963026	-230.771846
2019-04-18	58.248124	58.797739	-94.357490

```
5-day Average Slippage: -146.22723643836122
```

```
for n in range(9):
    bools = pd.isnull(trades_grid[n]['shares'])
    for ind in range(len(bools)):
        if bools[ind] == True:
            end = ind - 1
            break
    return_over_period = minute_data['close'][end] - minute_data['close'][0]
    opportunity_cost = unexecuted_size[n] * return_over_period
    if opportunity_cost==-0.0:
        opportunity_cost=0.0
    print('Opportunity cost under ' + titles[n] + ': ' + str(opportunity_cost))
```

```
Opportunity cost under pct_volume=0.05, order_size=500000: 1.0477378964424034e-10
Opportunity cost under pct_volume=0.05, order_size=1000000: 0.0
Opportunity cost under pct_volume=0.05, order_size=7000000: -22197609.56954998
Opportunity cost under pct_volume=0.15, order_size=500000: 0.0
Opportunity cost under pct_volume=0.15, order_size=1000000: 1.056469045579433e-09
Opportunity cost under pct_volume=0.15, order_size=7000000: 0.0
Opportunity cost under pct_volume=0.25, order_size=500000: 0.0
Opportunity cost under pct_volume=0.25, order_size=1000000: 0.0
Opportunity cost under pct_volume=0.25, order_size=7000000: 0.0
```