Инструкция по запуску кода Торговой площадки

Подготовила команда: Алимова И., Самарина А., Сухих В., Тарасов А., Ханиев А.

Торговая площадка - платформа, которая позволяет торговать участникам рынка (users). Нами были созданы книга заявок OrderBook, книга сделок TradeBook, стакан заявок, а также включены 4 участника рынка, такие как бот моментум по рыночным заявкам, бот моментум-реверс по рыночным заявкам. На основе стакана заявок формируем таблицу лучших цен спроса и предложения. Платформа позволяет задать количество users.

Файлы Orderlog должны находить в одной папке с кодом, потому что рабочая директория та, где код. Функции юзеров необходимо вставить в текущий код.

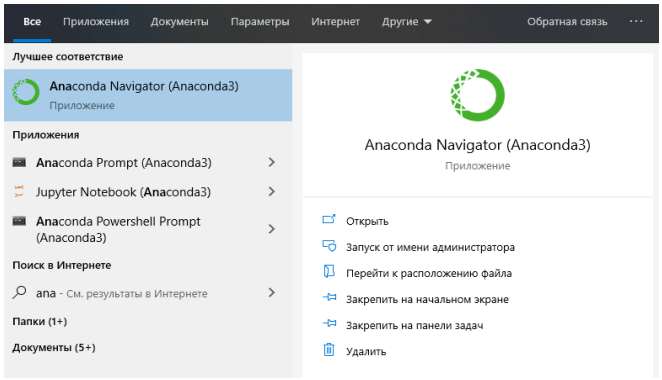
Код торговой площадки был написан на языке Python.

Для того чтобы прогнать код, необходимо скачать следующие программы:

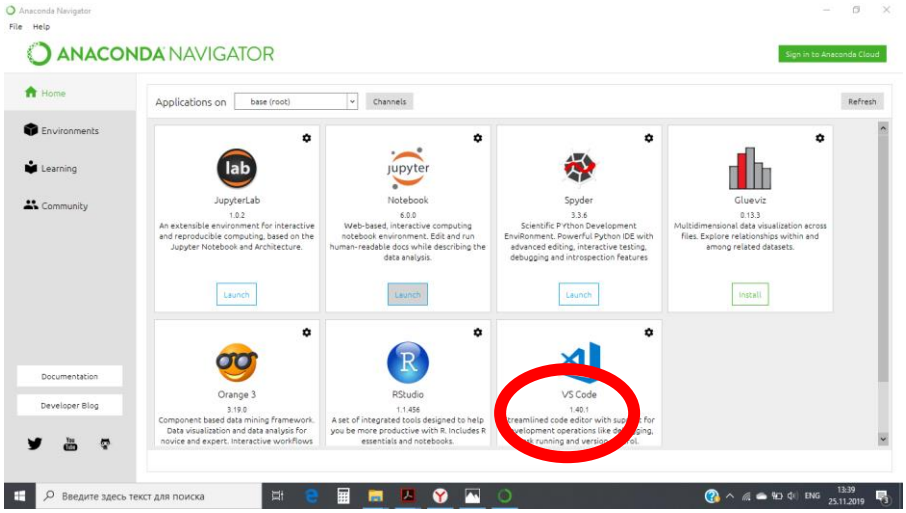
1) Скачиваем Python3 с официального сайта https://www.python.org/. Заходим во вкладку Downloads и выбираем нужную операционную систему. Скачиваем последний релиз. При установке не нужно ничего менять
2) Скачиваем Anaconda также с официального сайта https://www.anaconda.com/distribution/ Скачиваем последнюю версию для своей операционной системы. При установке также менять ничего не надо

Для того чтобы начать работу, необходимо выполнить следующие действия:

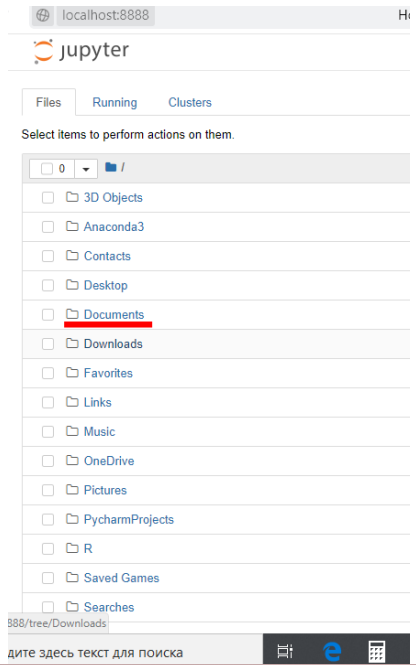
1) Запускаем Anaconda. Для этого в Пуске ищем Anaconda Navigator и запускаем его



2) В открывшемся окне кликаем Launch под Jupyter Notebook



3) Теперь открываем файл со скриптом. Для этого, если вы не перемещали скаченный файл, то заходим в Downloads, находим нужный файл и открываем его. Если же вы переместили файл со скриптом в другую папку, то нажмите Upload и найдите файл на компьютере.


localhost:8888
jupyter
Files
Running
Clusters
Select items to perform actions on them.
/
3D Objects
Anaconda3
Contacts
Desktop
Documents
Downloads
Favorites
Links
Music
OneDrive
Pictures
PycharmProjects
R
Saved Games
Searches
888/tree/Downloads
дите здесь текст для поиска


jupyter
Quit
Logout
Files
Running
Clusters
Select items to perform actions on them.
Upload
New
Click to browse for a file to upload.
/ Downloads
Name
..
несколько секунд назад
hft
месяц назад
БД
12 дней назад
ВКР
2 месяца назад
Логика
8 месяцев назад
Магистратура
2 месяца назад
Метрика
2 месяца назад
ОмГУ_лекции
4 месяца назад
Оплата общаги
25 дней назад
Репликация
4 дня назад
Теория Финансов
2 месяца назад
Установка
9 месяцев назад
chains_identification.ipynb
25 дней назад
23.5 kB
chains_identification_1.ipynb
25 дней назад
20.3 kB

Реализация кода

Начинаем с формирования базы данных.

Формируем базу данных

```
In [*]: # Сколько надо Юзеров
        # Определяем количество элементов в USERS
        try:
            a = int(input("Сколько участников торгов? \n"))
            USERS = ["User_{}".format(i) for i in range(1, a + 1)]
        except:
            print("Введите число")
```

```
Сколько участников торгов?
4
```

В первой ячейке задаём количество юзеров. В нашем коде необходимо задать 4, потому что далее представлено 4 стратегии юзеров.

Запускаем вторую ячейку. В результате получаем файл hist_price, который собирает в себе торги за несколько дней, данных нам, по тикеру SBER (пользователь может выбрать любой другой тикер), со стороны покупки и только завершённые сделки. В данной части кода можно задать большее количество дней, для этого следует скопировать 5-7 строки и заменить df1 на dfn, где n – день торгов.

Формируем файл исторических цен за 5 дней

```
import os
import pandas as pd
import numpy as np

df1 = pd.read_csv("OrderLog20151207.txt")
df1 = df1[(df1['SECCODE'] == "SBER") & (df1['ACTION'] == 1) & (df1['BUYSELL'] == "B")]
df1['DAY'] = 1
df2 = pd.read_csv("OrderLog20151208.txt")
df2 = df2[(df2['SECCODE'] == "SBER") & (df1['ACTION'] == 1) & (df1['BUYSELL'] == "B")]
df2['DAY'] = 2
df3 = pd.read_csv("OrderLog20151209.txt")
df3 = df3[(df3['SECCODE'] == "SBER") & (df1['ACTION'] == 1) & (df1['BUYSELL'] == "B")]
df3['DAY'] = 3
df4 = pd.read_csv("OrderLog20151210.txt")
df4 = df4[(df4['SECCODE'] == "SBER") & (df1['ACTION'] == 1) & (df1['BUYSELL'] == "B")]
df4['DAY'] = 4
df5 = pd.read_csv("OrderLog20151211.txt")
df5 = df5[(df5['SECCODE'] == "SBER") & (df1['ACTION'] == 1) & (df1['BUYSELL'] == "B")]
df5['DAY'] = 5

df_w = pd.concat([df1, df2, df3, df4, df5], keys = [1, 2, 3, 4, 5])

df_w['CURRENT_MINUTE'] = (df_w['TIME']) % (100 * 1000)
df_w = df_w.pivot_table(index = ['SECCODE', 'CURRENT_MINUTE'], values = ['PRICE'],
                        aggfunc={'PRICE': ['first', 'min', 'max', 'last']})
df_w = pd.DataFrame({'OPEN': df_w['PRICE']['first'],'LOW': df_w['PRICE']['min'],
                     'HIGH': df_w['PRICE']['max'], 'CLOSE': df_w['PRICE']['last'] })
df_w['TIME'] = range(1, len(df_w.index) + 1)
```

Файл hist_price.csv содержит исторические цены сделок актива в формате свечей частотой 1 минута.

TIME	OPEN	LOW	HIGH	CLOSE
1	103.45	103.08	103.53	103.2
2	103.19	103.15	103.29	103.29
3	103.29	103.25	103.49	103.48
4	103.48	103.4	103.54	103.44
5	103.43	103.3	103.45	103.33

В следующей ячейке конвертируем время в привычное нам время (часы + минуты + секунды). Формируем базу данных для пользовательских роботов, включающую список текущих действий роботов и временных данных. Создаем книгу заявок, создаем книгу сделок, создаем стакан заявок, создаём таблицы, в которые будут добавляться данные по итогам торгов.

```
In [65]: OrderBook
```

Out[65]:

	no	time	trade_sec	ID	action	buysell	orderno	bot_orderno	price	volume
0	0	0	0	bot	drop	0	0	0	0	0

```
In [66]: TradeBook
```

Out[66]:

	tradeno	time	trade_sec	buy_ID	sell_ID	buy_orderno	sell_orderno	price	volume
0	0	0	0	bot	bot	0	0	0	0

```
In [67]: DOM
```

```
Out[67]: {'supply':    orderno   ID  price  volume  bot_orderno
          0         0  bot      0      0       0,
          'demand':    orderno   ID  price  volume  bot_orderno
          0         0  bot      0      0       0}
```

```
In [68]: bid_ask
```

Out[68]:

	time	bid	ask
0	0	0	0

Далее идут поля, в которые необходимо добавить стратегии юзеров. В нашем коде представлено 4 функции, содержащие:

бота со стратегией моментум по рыночным торгам;

бота со стратегией моментум по рыночным торгам с иной скользящей средней;

бота, работающего на реверс;

бота, работающего на реверс с иной скользящей средней.

В следующих трёх ячейках прописаны функции, которые нужны непосредственно для работы торговой площадки:

queueing – выставляет заявки юзеров по порядку.

Непосредственно торговая площадка

```
def group_mutate_ungroup(df, column_of_values, id_name):
    """
    Это промежуточная функция, нужна для того, чтобы заменить несколько строк в R
    В дальнейшем используется внутри queueing
    """
    df_ = df.sort_values(by = [column_of_values])
    nums = []
    for val in np.unique(df_[column_of_values]):
        nums += range(1, len(df_[df_[column_of_values] == val]) + 1)
    df_[id_name] = nums
    return df_

def queueing(hist_price, bid_ask, accounts_info, temp_data):

    # Создаем список с функциями юзеров
    robots = [strategy_M,strategy_M2,strategy_R,strategy_R2]
```

Здесь же формируется список функций роботов-юзеров, для этого в выделенную на рисунке область необходимо добавить название функций, далее о скрипт их переименут в User_1, User_2 и т.д.

user_trading – реализует заявки от юзеров по рыночной цене, лимитные и заявка на снятие ранее поданной лимитной;

bot_trading – выполняет функцию маркет-мейкера, на основе исторических данных выставляет заявки с большими объемами, для поддержания исторического уровня цен.

В последней ячейке запускается непосредственно торговый цикл

Комментарии также указаны в скрипте.

Требования к функции торгового робота

Торговый робот должен быть реализован в виде функции на языке python. В скрипте должно быть прописано, какие пакеты необходимо установить дополнительно. Постарайтесь заранее предвидеть возможные конфликты функций из разных пакетов, и явно это указать.

Input

- hist_price – датафрейм, содержащий исторические цены сделок в формате японских свечей частотой 1 минута. Данные о текущей свече обновляются каждую секунду;
- bid_ask – датафрейм из одной строки и трёх столбцов:
    - $time – номер секунды от начала торгов (int);
    - $bid – лучшая цена спроса на данную секунду (float);
    - $ask – лучшая цена предложения на данную секунду (float);
- account_info – список, содержащий информацию о счете и открытых позициях:
    - $cash – количество денежных единиц (float), находящиеся на счету (на старте 100 000, разрешается уйти в минус)
    - $assets – количество единиц актива (int), находящиеся на счету (на старте 0, разрешается уйти в минус)
    - $limit_orders – датафрейм, содержащий информацию об активных лимитных позициях, открытых роботом (по строчкам), со столбцами

- ✦ $orderno – порядковый номер (int) приказа в OrderBook
- ✦ $buysell – признак покупки продажи (chr): «B» – на покупку, «S» -- на продажу;
- ✦ $price – цена заявки (float);
- ✦ $volume – объем заявки (int);

- temp_data – переменная произвольной структуры, которую можно использовать для передачи информации между торговыми периодами.

Output

Список содержащий два элемента

- action – датафрейм, содержащий в строках описание действий, которые необходимо совершить с приказами на торговой площадке. Содержит столбцы:

  - $action – тип действия (int), которое надо совершить (см. таблицу ниже);
  - $orderno – порядковый номер (int) приказа в orderbook (только action = 2);
  - $buysell – признак покупки продажи (chr): «B» – на покупку, «S» -- на продажу;
  - $price – цена заявки (float);
  - $volume – объем заявки (int);

  Таблица учета информации в зависимости от типа действия

  (T – информация учитывается;

  F – информация игнорируется)

action type	$action	$orderno	$buysell	$price	$volume
set markert order	0	F	T	F	T
set limit order	1	F	T	T	T
drop limit order	2	T	F	F	F

- temp_data – переменная произвольной структуры, которую можно использовать для передачи информации между торговым периодами.

Описание алгоритма торговли

Обновление данных, исполнение приказов и реализация сделок происходит дискретно каждую игровую секунду. На торговой площадке есть участники-боты, поэтому на момент начала торгов стакан заявок не пустой.

Ежесекундный торговый цикл:

1. каждому торговому роботу передается информация об исторических ценах последних сделок (дискретность данных 1 минута, обновление каждую секунду), текущем бид-аск спреде и состоянии его счета, а также переменная temp_data (на первой секунде пустая);
2. торговые роботы производят расчеты, и возвращают системе список торговых действий;
4. Система проверяет списки действий на ошибки. Если информации для исполнения действия недостаточно, то действие игнорируется. Если есть избыточная информация, то действие совершается, но избыточная информация игнорируется.
5. Сначала исполняются рыночные приказы по очереди по одному от каждого робота (сначала исполняются все первые по порядку рыночные приказы роботов, потом вторые и т.д.);
6. Затем в аналогичном порядке выставляются лимитные приказы, если в стакане имеется подходящая встречная сделка, то она исполняется с разу же.
7. Затем производится снятие заявок
8. Система обновляет данные по историческим ценам, бид-аск спреду и информации о счете и переходит к п.1.