RANDOM TRADING ROBOT

The purpose of this program is to enable trading of stocks by multiple trader bots generating orders according to certain random distributions.

# Before you start

## Import numpy dependency:

python -m pip install numpy

## Run the CDA instance:

python3 run\_exchange\_server.py --host 0.0.0.0 --port 9001 --debug --mechanism cda

# Set up variables:

## Quantity:

Go to build\_message() function (line 154). There you will be able to edit the average arrival rate of stocks per pre-set unit of time shares\_rate\_avg (line 160). It will be used to create an order with the amount of shares drawn from the Poisson distribution with λ=shares\_rate\_avg. The frequency of arrival of orders to the exchange can be changed by changing sleep time in line 264.

## Price:

Price will be drawn from the normal distribution N(μ,σ2). You can adjust the parameters by changing the constants mean\_price= μ and sd=σ (lines 162-163).

## Buy/sell probability:

By default, traders are set so that the quantities of ‘buy’ and ‘sell’ orders are equal. You can change that by changing the probability of a ‘buy’ order prob\_buy in line 156.

# Running the clients:

python random\_station.py <number\_of\_traders>

where number\_of\_traders is the desired number of traders between 1 and 9999.

# Interpreting the results:

## .csv filenames

The program makes a separate .csv file for each trader during each trading session. The convention for naming of the file is:

year\_month\_day\_hour-minute\_UUUU.csv

where UUUU is the user ID assigned by the iterator.

## File content:

The file contains the data for every message sent and received from the exchange. Each line of date is arranged in the following order:

'order\_ID', 'status', 'direction', 'time\_in\_force', 'timestamp','stock\_price', 'stock\_quantity', 'trader\_cash', 'current\_stock'

* order\_ID: consists of 14 digits, 4 of which designate the ID of trader, the following 10 is the ordinal for the order number for the current trader.
* status: ‘O’ means order was sent on exchange, but has not been confirmed, ‘A’ means order was accepted on the exchange, ‘E’ means that at least one share from this order was traded. Kept OUCH abbreviations for consistency.
* direction: ‘B’ for buy and ‘S’ for sell.
* time\_in\_force: only for ‘O’ messages, because this is the only place we are able to get that variable.
* trader\_cash: change of cash possessed since the beginning of the session (can be negative).
* current\_stock: change of stock quantity possessed since the beginnng of the session (can be negative).

## Example

# Automatized version (automated\_random\_station.py)

## Description

In this program the user is able to set up a number of classes of traders, according to the formula:

tradeGroup = Trader\_Group(#traders\_in\_class, prob\_of\_buying, mean\_price, sd\_price, rate\_of\_arrival)

Those classes can be added and amended in the final paragraph of the program, which is the instruction for execution of the main function:

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

As we want the orders to arrive in the frequency determined by the exponential distribution on some constant rate of arrival, by default the quantity of shares traded in each order is 1.