B4 - Computer Numerical Analysis – Trade

Trade

"Hard work beats talent every time."

Trade

binary name: trade
language: everything working on "the dump"

⚛ • The totality of your source files, except all useless files (binary, temp files, obj files,...), must be included in your delivery.

⚛ We accept a lot of languages. However most scientific libraries, such as tensorflow or scipy are not available. In any case, you are expected to build yourself the tools you need.

Trading bots are able to compute thousands of orders per second (this is called High Frequency Trading) which in return, of course, modifies the very nature of markets.

They are not merely trading spots anymore, but have become fighting arenas for various devices of artificial intelligence.

🎧 Today, most of the transactions are processed by algorithmic trading.

Besides, this very promising field is still widely open.
This is why the best talents in the world are working on it.
You are one of them.

You are to create a trading bot that will bring about a revolution of the trade markets.

To get you acquainted with the basics, you can check some **indicators** (like the ones you built for a previous project) and you should try some online trading simulators (such as tradingview.com).

However, a really good algorithm will use many sophisticated techniques, so we can only suggest you do a lot of research!

EVALUATION

The aim is to **make as much money as possible AND to justify how your trading bot is (not) working.**

3 training datasets and a generator are provided alongside this subject. Some other datasets will be used for evaluation and your total amount of money will be compared with other groups.

Be careful with ovefitting, your algorithm MUST adapt.

In the past, some students tried to make millions of profits with deterministic algorithms. Nowadays, they'll succeed to get, at least, -42!

First, the server sends general information about the game:

```
info = 'settings' variable value (, value)*
variable = string
value = string | integer
```

Then comes first part of the data: no action is asked, this is just for training:

```
update_c = 'update game next_candles' rate ';' rate ';' rate
rate = currency '_' currency decimal (',' decimal){5}
currency = 'BTC' | 'USDT'
```

Finally the rest of the data comes online: your algorithm is asked what to do and has to make decision within seconds (otherwise the whole program collapses and you lose everything):

```
session = update_c eol update_s eol 'action order' integer
update_c = 'update game next_candles' rate ';' rate ';' rate
update_s = 'update game stacks' currency : decimal ',' currency : decimal ','
    currency : decimal
rate = currency '_' currency ',' decimal (',' decimal){5}
currency = 'BTC' | 'USDT'
```

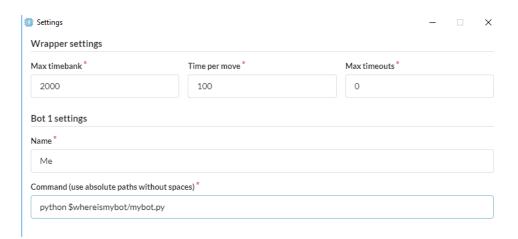Your bot has to answer with a strict grammar:

```
order = 'pass' | trade_order
trade_order = ('buy' | 'sell') currency '_' currency decimal
currency = 'BTC' | 'USDT'
```

Orders need to be valid all the time; typically any attempt to sell more than what you own will collapse the program. This partial grammar is only to help you start. You will have to reverse engineer the logs from the server.

Architecture

For training purposes, you should download the client-server interface (check the Bootstrap). Tell it the location of your bot together with the command line instruction, and watch it make millions.