B4 - Computer Numerical Analysis – Trade

B-CNA-410

Trade

"Hard work beats talent every time."

4.2.2

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.

Check some **indicators**, like the ones you built for the Groundhog project.... It may help you get an idea of what is happening. For instance, the combination of a high g and a negative r may suggest the value is undervalued and may start increase soon. Some tool such as the **Bollinger bands** may also help you...

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

Evaluation

The aim of this project it to make as much money as possible.
3 training datasets and a generator are provided alongside this subject.
For the final evaluation, other datasets will be used, your algorithm MUST adapt!

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

Updates and answers

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):

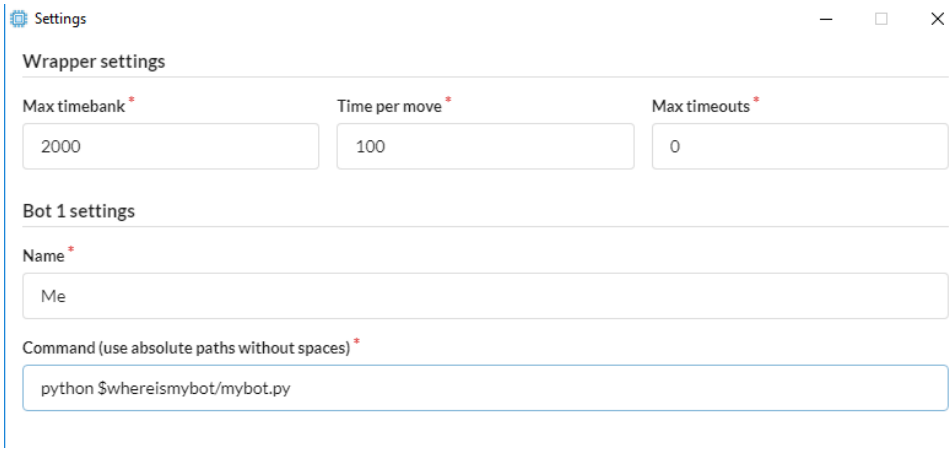
```
session = update_c eol update_s eol 'action order' integer
update_c = 'update game next_candles' rate ';' rate ';' rate
update_s = 'update game stacks' currency : decimal ',' currency : decimal ','
    currency : decimal
rate = currency '_' currency ',' decimal (',' decimal){5}
currency = 'BTC' |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