# B4 - Computer Numerical Analysis – Trade

B-CNA-410

# Trade

"Hard work beats talent every time."

4.0

# Trade

| | |
|---|---|
| **binary name:** | trade |
| **repository name:** | CNA_trade_`$ACADEMICYEAR` |
| **repository rights:** | ramassage-tek |
| **language:** | everything working on "the dump" |

- Your repository must contain the totality of your source files, but no useless files (binary, temp files, obj files,...).

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.

Since 2002, trading robots, AKA algorithmic trading, are considered to be one of the major breakthroughs on financial markets over the last decade. In concrete terms, these bots' mission is to evaluate a share's quotation, and to react by deciding to either buy or sell, through an automated process.

These 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, 70% of the transactions in the USA are processed by algorithmic trading, and almost 50% in Europe.

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

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

Check the indicators you built in 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 product is undervalued and may start increase soon; the Bollinger bands may help you distinguish noise from the real trend...
However, a really good algorithm will use much more sophisticated techniques, so we can only suggest you do a lot of research on that one!

## Evaluation

The aim of this project it to make as much money as possible.
A training dataset is provided alongside this subject. For the final evaluation, another dataset (with different values) will be used, your algorithm has to adapt!

## 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 = 'ETH' | '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):

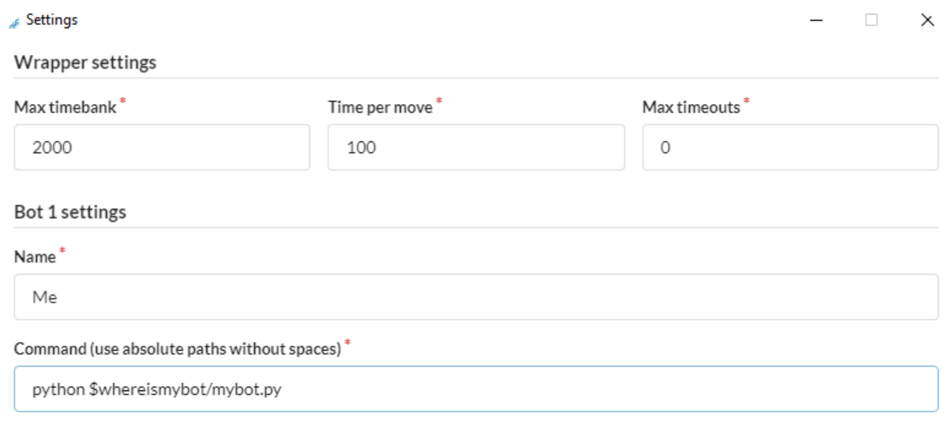
```
session = update_c eol update_s eol 'action order' integer
update_c = 'update game next_candles' rate ';' rate ';' rate
update_s = 'update game stacks' currency : decimal ',' currency : decimal ','
    currency : decimal
rate = currency '_' currency ',' decimal (',' decimal){5}
currency = 'ETH' |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

Your bot has to answer with a strict grammar:

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


Settings

Wrapper settings

|  |  |  |
|---|---|---|
| Max timebank * | Time per move * | Max timeouts * |
| 2000 | 100 | 0 |

Bot 1 settings

Name *
Me

Command (use absolute paths without spaces) *
python $whereismybot/mybot.py

For training purposes, you can download the client-server interface, tell it the location of your bot together with the command line instruction, and watch it make millions.