B4 - Computer Numerical Analysis – Trade

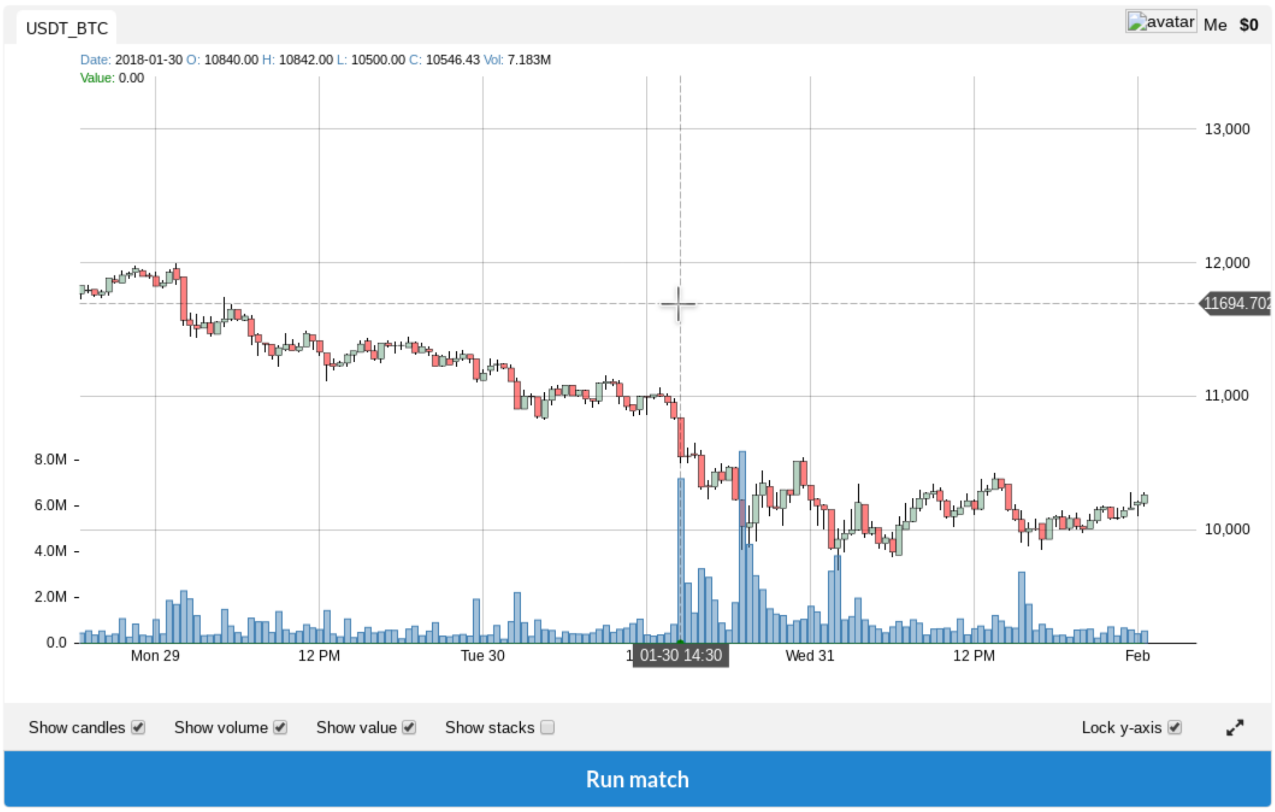
B-CNA-410

Trade

Bootstrap

3.2.2

Let's play around with a trading environment.



You are to craft a simple client bot, that:

- parse given information,
- make a simple forecast
- issue a sell/buy order accordingly.

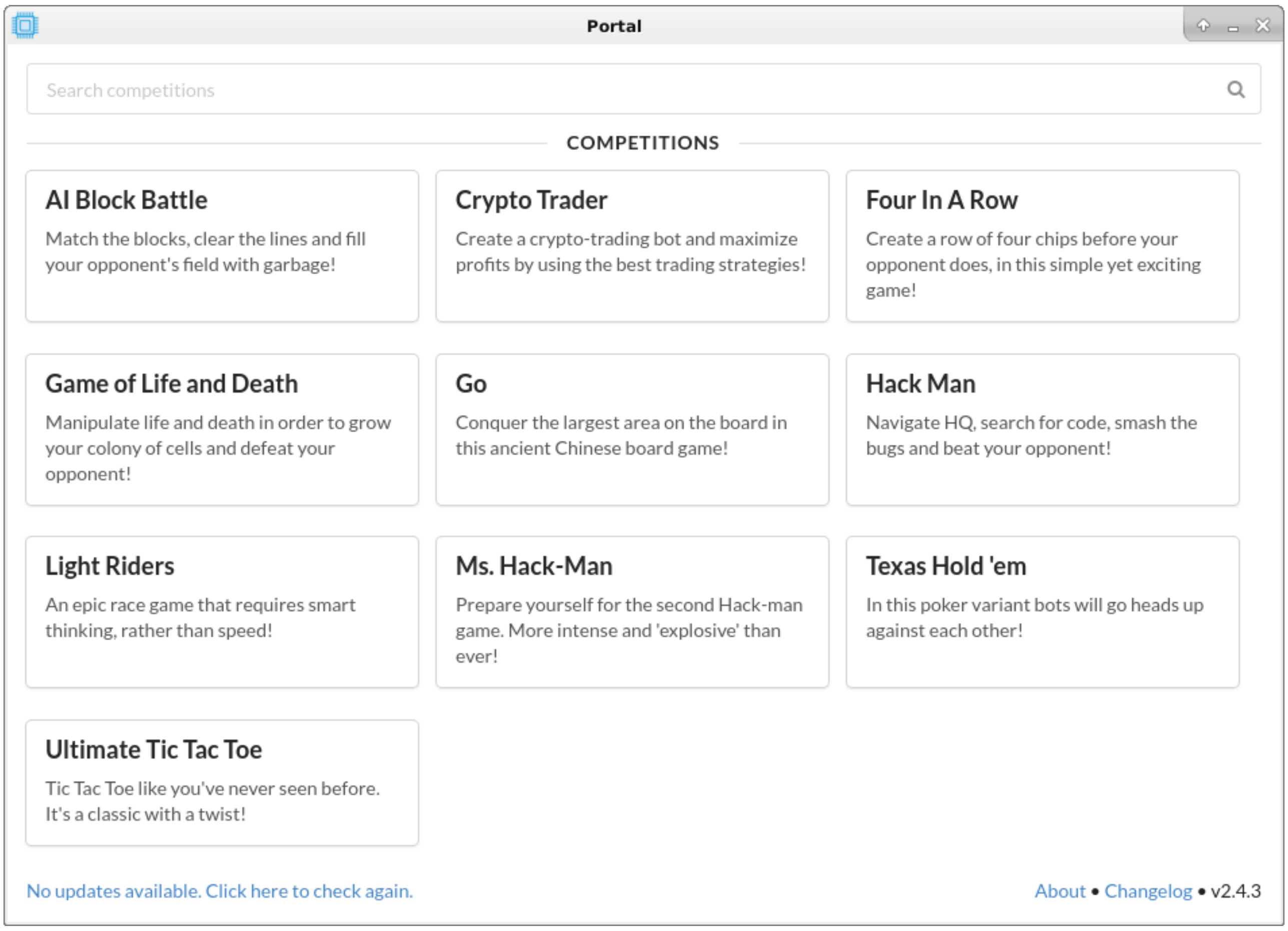
Step 0 - Let's gear up

The program `ai-bot-workspace` is the tool used for testing and evaluating this project.
Here's a short guide to get you started.

Download it from the Github source OR along with this subject.

- If you're on Linux, you should use `ai-bot-workspace-2.4.3-x86_64.AppImage`
- If you're on Windows, you sould use `ai-bot-workspace-setup-2.4.3.exe`

The executables launch ai-bot-workspace's portal. There, you must select `Crypto Trader`



Portal

Search competitions

COMPETITIONS

AI Block Battle
Match the blocks, clear the lines and fill your opponent's field with garbage!

Crypto Trader
Create a crypto-trading bot and maximize profits by using the best trading strategies!

Four In A Row
Create a row of four chips before your opponent does, in this simple yet exciting game!

Game of Life and Death
Manipulate life and death in order to grow your colony of cells and defeat your opponent!

Go
Conquer the largest area on the board in this ancient Chinese board game!

Hack Man
Navigate HQ, search for code, smash the bugs and beat your opponent!

Light Riders
An epic race game that requires smart thinking, rather than speed!

Ms. Hack-Man
Prepare yourself for the second Hack-man game. More intense and 'explosive' than ever!

Texas Hold 'em
In this poker variant bots will go heads up against each other!

Ultimate Tic Tac Toe
Tic Tac Toe like you've never seen before. It's a classic with a twist!

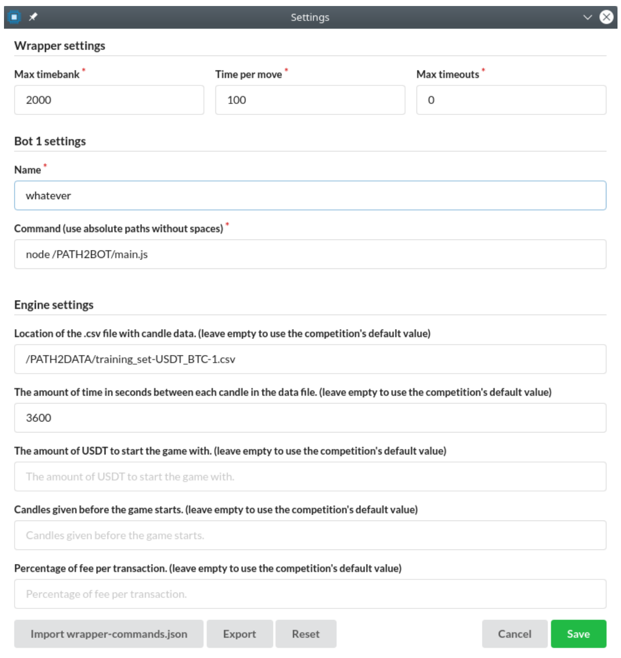
No updates available. Click here to check again.

About • Changelog • v2.4.3

Before the first launch, you have to configure some **settings**:

- the command to run your executable (interpreter + path)
- the path to a .csv file with candlestick chart data
- the amount of time in seconds between each candle

Below are 2 screenshots showing how it should be configured once you're done here:



Wrapper settings

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

Bot 1 settings

Name *: whatever

Command (use absolute paths without spaces) *: node /PATH2BOT/main.js

Engine settings

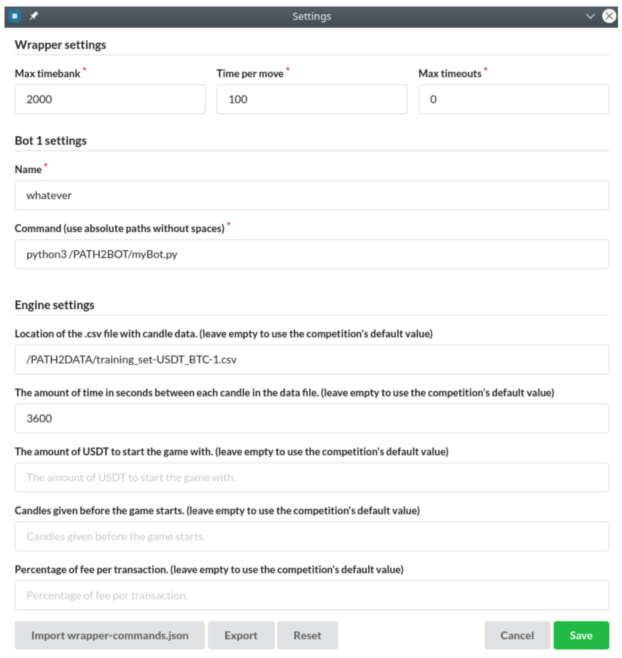
Location of the .csv file with candle data. (leave empty to use the competition's default value): /PATH2DATA/training_set-USDT_BTC-1.csv

The amount of time in seconds between each candle in the data file. (leave empty to use the competition's default value): 3600

The amount of USDT to start the game with. (leave empty to use the competition's default value): The amount of USDT to start the game with.

Candles given before the game starts. (leave empty to use the competition's default value): Candles given before the game starts.

Percentage of fee per transaction. (leave empty to use the competition's default value): Percentage of fee per transaction.



Wrapper settings

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

Bot 1 settings

Name *: whatever

Command (use absolute paths without spaces) *: python3 /PATH2BOT/myBot.py

Engine settings

Location of the .csv file with candle data. (leave empty to use the competition's default value): /PATH2DATA/training_set-USDT_BTC-1.csv

The amount of time in seconds between each candle in the data file. (leave empty to use the competition's default value): 3600

The amount of USDT to start the game with. (leave empty to use the competition's default value): The amount of USDT to start the game with.

Candles given before the game starts. (leave empty to use the competition's default value): Candles given before the game starts.

Percentage of fee per transaction. (leave empty to use the competition's default value): Percentage of fee per transaction.

You are encouraged to play with all the values to test the adaptability of your bot.
The default dataset within the ai-bot-workspace contain 30-minutes candles for multiple pairs.

All provided datasets, as the ones used for evaluation, only contain 60-minutes candles for USDT_BTC trading pair.
The 3 last settings will be kept as default (1000 USDT, ~337 given candles before start, 0.2% transaction fee)

Step 1 - Get some data

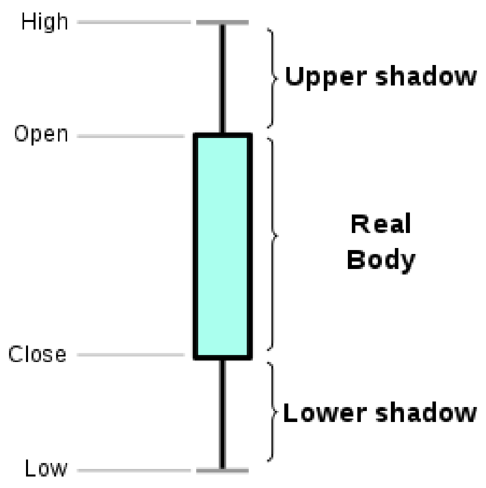
Close to this file, you could get your hands on 3 training datasets.

Each (training and evaluation) set contain **60-minutes candles data for USDT_BTC trading pair**

You're also provided with a tool to generate "fake" candles data, in case you need some more.

You are encouraged to test with other datasets. Many specialized websites provide some historical OHLCV price data that can be downloaded as csv files.

Taxonomy



- pair : The chart to which this candle belongs
- date : Unix timestamp representing a datetime
- high : The highest price traded in this candle
- low : The lowest price traded in this candle
- open : The opening price of this candle
- close : The closing price of this candle
- volume : The total volume that has been traded

STEP 2 - A FIRST DUMB BOT

STARTER-BOTS

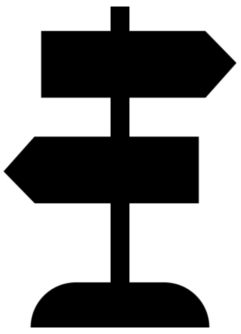
To quickly get you started and only focused on the fun part, you're provided with some starter bots. You can download some of them from Github OR all of them from your intranet.
Feel free to use one of them or build your own bot from scratch if you prefer.

Each provided starter-bot performs a stupid, but valid, operation. All provided bots may not perform the same operation, you should have a closer look.

In the language of your choice, write OR customize an elementary bot that only performs one single operation ten times, then passes until the end. Please refer to the subject and/or the last section below to get you acquainted with the **bot orders** grammar.

Make sure you respect carefully the syntax so the server understands your orders.
Damn it.

Once your bot is ready, just tell the server how to find it.



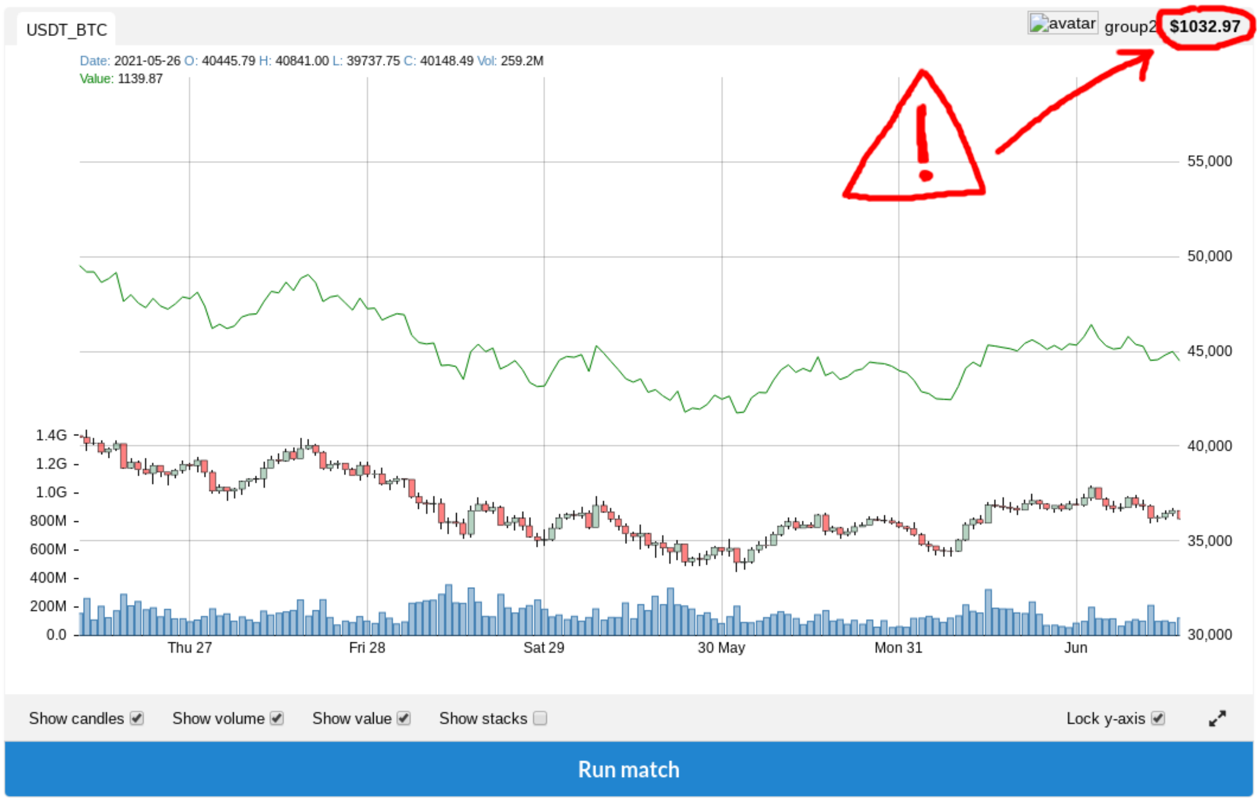
Step 3 - Know your stocks

You have your first bot, but no thrill so far...
Let's add some essential features:

- store the amount of currency you own in the beginning, according to the messages from the server
- when issuing a sell/buy order, fix the amount of currency you sell to X% of your current stockpile
- naturally, update the stacks according to your operations

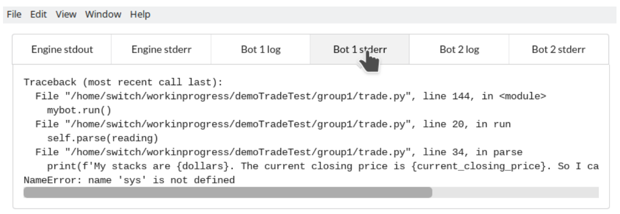
Don't forget the transaction fees when updating!

Now, your bot should be able to hold firm and never crash... but may lose a bunch of money overall.

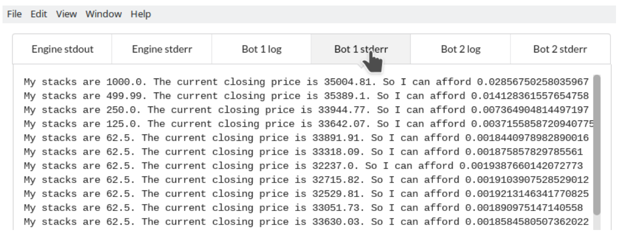


The first thing you want to check is probably your **final result on the top right corner of the chart.**

The `Bot 1 stderr` window will warn you in case something is going wrong. You should use this specific output to debug your bot or check what it's doing.

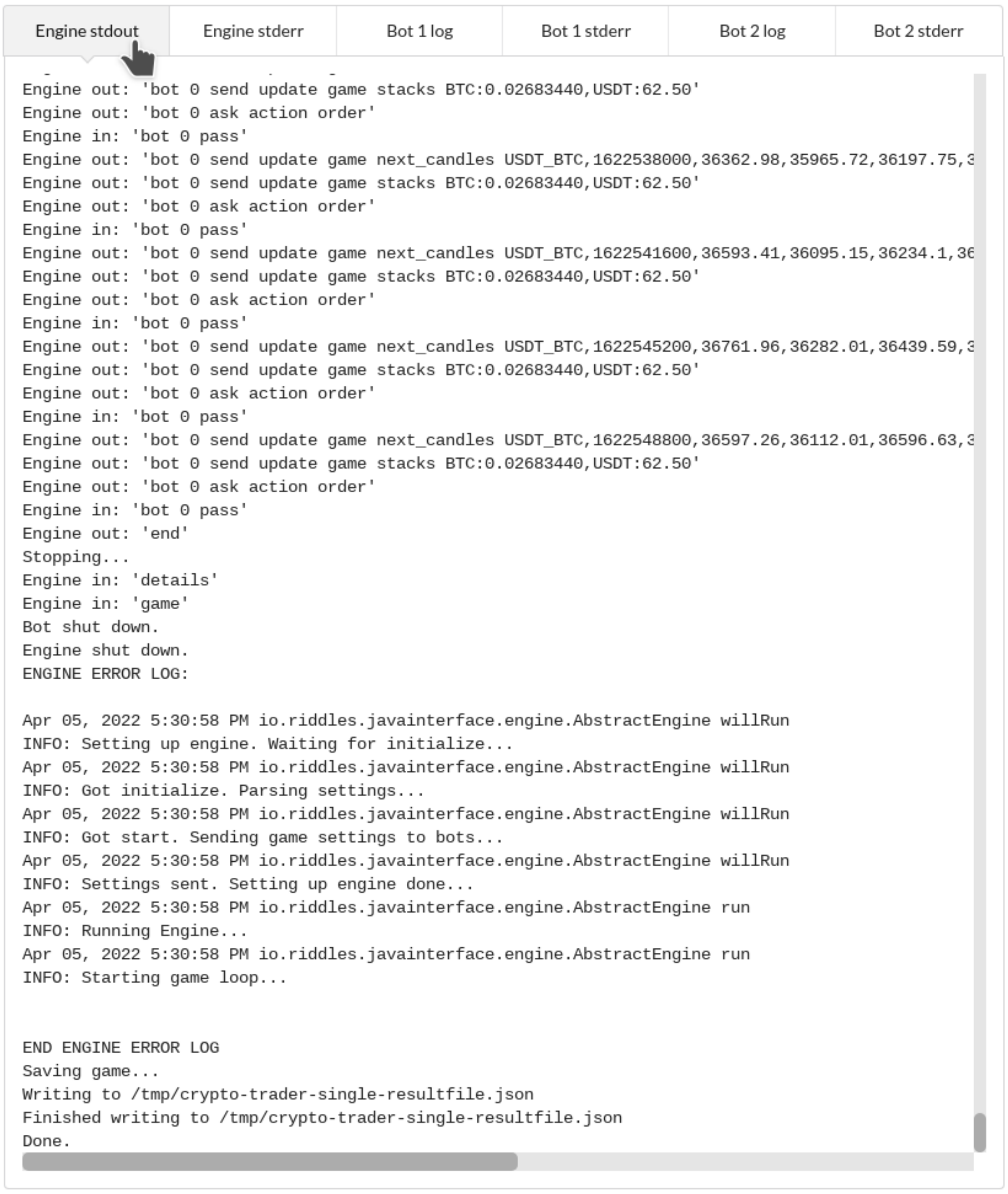


```
Traceback (most recent call last):
  File "/home/switch/workinprogress/demoTradeTest/group1/trade.py", line 144, in <module>
    mybot.run()
  File "/home/switch/workinprogress/demoTradeTest/group1/trade.py", line 20, in run
    self.parse(reading)
  File "/home/switch/workinprogress/demoTradeTest/group1/trade.py", line 34, in parse
    print(f'My stacks are {dollars}. The current closing price is {current_closing_price}. So I ca
NameError: name 'sys' is not defined
```



```
My stacks are 1000.0. The current closing price is 35004.81. So I can afford 0.02856750258035967
My stacks are 499.99. The current closing price is 35389.1. So I can afford 0.014128361557654758
My stacks are 250.0. The current closing price is 33944.77. So I can afford 0.007364904814497197
My stacks are 125.0. The current closing price is 33642.07. So I can afford 0.0037155858720940775
My stacks are 62.5. The current closing price is 33891.91. So I can afford 0.0018440978982890016
My stacks are 62.5. The current closing price is 33318.09. So I can afford 0.001875857829785561
My stacks are 62.5. The current closing price is 32237.0. So I can afford 0.001938766014207273
My stacks are 62.5. The current closing price is 32715.82. So I can afford 0.0019103907528529012
My stacks are 62.5. The current closing price is 32529.81. So I can afford 0.0019213146341770825
My stacks are 62.5. The current closing price is 33051.73. So I can afford 0.001890975147140558
My stacks are 62.5. The current closing price is 33630.03. So I can afford 0.0018584580507362022
```

The other windows, `Engine stdout` and `Bot 1 log`, will roughly provide you the same details.



```
Engine out: 'bot 0 send update game stacks BTC:0.02683440,USDT:62.50'
Engine out: 'bot 0 ask action order'
Engine in: 'bot 0 pass'
Engine out: 'bot 0 send update game next_candles USDT_BTC,1622538000,36362.98,35965.72,36197.75,3
Engine out: 'bot 0 send update game stacks BTC:0.02683440,USDT:62.50'
Engine out: 'bot 0 ask action order'
Engine in: 'bot 0 pass'
Engine out: 'bot 0 send update game next_candles USDT_BTC,1622541600,36593.41,36095.15,36234.1,36
Engine out: 'bot 0 send update game stacks BTC:0.02683440,USDT:62.50'
Engine out: 'bot 0 ask action order'
Engine in: 'bot 0 pass'
Engine out: 'bot 0 send update game next_candles USDT_BTC,1622545200,36761.96,36282.01,36439.59,3
Engine out: 'bot 0 send update game stacks BTC:0.02683440,USDT:62.50'
Engine out: 'bot 0 ask action order'
Engine in: 'bot 0 pass'
Engine out: 'bot 0 send update game next_candles USDT_BTC,1622548800,36597.26,36112.01,36596.63,3
Engine out: 'bot 0 send update game stacks BTC:0.02683440,USDT:62.50'
Engine out: 'bot 0 ask action order'
Engine in: 'bot 0 pass'
Engine out: 'end'
Stopping...
Engine in: 'details'
Engine in: 'game'
Bot shut down.
Engine shut down.
ENGINE ERROR LOG:

Apr 05, 2022 5:30:58 PM io.riddles.javainterface.engine.AbstractEngine willRun
INFO: Setting up engine. Waiting for initialize...
Apr 05, 2022 5:30:58 PM io.riddles.javainterface.engine.AbstractEngine willRun
INFO: Got initialize. Parsing settings...
Apr 05, 2022 5:30:58 PM io.riddles.javainterface.engine.AbstractEngine willRun
INFO: Got start. Sending game settings to bots...
Apr 05, 2022 5:30:58 PM io.riddles.javainterface.engine.AbstractEngine willRun
INFO: Settings sent. Setting up engine done...
Apr 05, 2022 5:30:58 PM io.riddles.javainterface.engine.AbstractEngine run
INFO: Running Engine...
Apr 05, 2022 5:30:58 PM io.riddles.javainterface.engine.AbstractEngine run
INFO: Starting game loop...


END ENGINE ERROR LOG
Saving game...
Writing to /tmp/crypto-trader-single-resultfile.json
Finished writing to /tmp/crypto-trader-single-resultfile.json
Done.
```

Step 4 - Parse and predict

To move ahead, we need to use market information to predict future outcomes.

Yes indeed, it does mean a bit of work.

Write a third bot (or upgrade your previous one), which performs the following actions:

- store the ‘candles’ as soon as they are provided,
- when asked to act, if bitcoins are on the rise (according to the last two values), buy as much as you can. Otherwise, sell.

Organize cleverly your code to clearly separate what handles interfacing with ai-bot-workspace and what addresses your AI.

Congratulations! Your environment is now fully set up.
You can now proceed to the interesting part of the project: artifical intelligence.

- Write an algorithm to place some smart orders
- Have a look at technical analysis
- Write another algorithm, with other **indicators**
- Get knowledge-rich or die tryin’

TECHNICALITIES

- Transaction fee
  - by default, the bot pays a 0.2% transaction fee for each order it places
- Timebank :
  - allow some additional thinking time when needed
  - start the game with 10 secs
  - each time an action is requested, the timebank is increased (100-500ms)
  - the time needed by the bot to request is deducted from the timebank
  - if the bot is too slow, the timebank will be exhausted

BOT ORDERS

- no_moves OR pass
  - you bot don't take any position
- buy CurrencyPaidWith_CurrencyReceived amount
  - amount is a float which specifies how much to buy of the CurrencyReceived (2nd symbol)
  - buy USDT_BTC 1 means the bot wants to buy 1 BTC in USDT
  - the cost of the operation would be 1 * [current USDT_BTC closing price]
  - when buying 1 BTC, the bot will actually receive 0.998 BTC
- sell CurrencyReceived_CurrencySold amount
  - amount is a float which specifies how much to sell of the CurrencySold (2nd symbol)
  - sell USDT_BTC 1 means the bot wants to sell 1 BTC in USDT
  - the bot will received 1 * [current USDT_BTC closing price] - [fee]
  - when selling some currency for 1 USDT, the bot will actually receive 0.998 USDT