my_tools

Release 0.0.1

Hadrien Nauroy

Jan 03, 2022

CONTENTS:

CHAPTER

ONE

MY_TOOLS

1.1 my_tools package

1.1.1 Subpackages

my_tools.cryptools package

Submodules

my_tools.cryptools.mock module

A script used to create mocks of binance API

exception my_tools.cryptools.mock.**SizeError**(*message*)
Bases: Exception

class my_tools.cryptools.mock.**kline_mock**(*data*)
Bases: Iterator

This class is aimed to retreive candle of binance API from a kline call one by one

input :

- data : raw data from client.get_historical_klines

object :

- iterator that returns a candle in OHLC format

usage :

To retreive open prices :

```
>>> for kline in kline_mock(data):
>>>     print(kline["Open"])
```

class my_tools.cryptools.mock.**klines_mock**(***kwargs*)
Bases: Iterator

This class is aimed to retreive candles of diffrent binance API from a kline calls one by one

input :

- **kwargs : raw data from client.get_historical_klines

object :

- iterator that returns all candles in OHLC format in a dict

usage :

To retreive all open prices :

```
>>> for klines in klines_mock(btc=data_btc, eth=data_btc):
>>>     print(klines["btc"]["Open"], klines["eth"]["Open"])
```

class my_tools.cryptools.mock.**tickers_mock**(*data*)

Bases: Iterator

This class is aimed to mock the get_all_tickers() functin of binance given a set data from previous call. It arrange data in a dict

input :

- data : list data from succesive call of client.get_all_tickers()

object :

- iterator that returns all all tickers in a dict one by one

usage :

To retreive BTCUSDT price :

```
>>> for tick in tickers_mock(data):
>>>     print(tick["BTCUSDT"])
```

my_tools.cryptools.visualisation module

The srcipt that contains visualisation tools

my_tools.cryptools.visualisation.**candle**(*data*, ***kwargs*)

This function is aimed to to plot candle with data from binance klines

parameters :

- data : raw data from "client.get_historical_klines(*args), type list
- **Kwargs : all sort of argument that mpl.plot can accept

More information here : https://github.com/matplotlib/mplfinance

my_tools.cryptools.visualisation.**organnise_data**(*data*)

this function handle the re-organisation of data in a dataframe

parameters :

- data : raw data from "client.get_historical_klines(*args), type list

outputs :

- data_frame : a panda DataFrame with Open, High, Low and Close columns with a DateTimeIndex

Module contents

my_tools.monitoring package

Submodules

my_tools.monitoring.bot module

This script is aimed too define all useful function for monitoring another bot with messenger

exception my_tools.monitoring.bot.**MessageError**(*type*, *world*)
Bases: Exception

class my_tools.monitoring.bot.**Monitor_Bot**(*config*)
Bases: object

A type of bot that is able to send messages and retreive instruction from user

Inputs :

- config : a monitor_config object with messenger password and username

Accepted command :

- START : start the other bot for undefined time
- START for <duration> : start the other bot for <duration>
- STOP : stop the other bot for undefined time
- STOP for <duration> : stop the other bot for <duration>

Note that "Stop" or "stop" will also work

Accepted duration :

- "x s" / "x sec" / "x second" / "x seconds" / "x secs"
- "x m" / "x min" / "x mins" / "x minute" / "x minutes"
- "x h" / "x hour" / "x hours
- "x d" / "x day" / "x day"

Note that a space is required between x and the time frame

Usage exemple:

- in messenger : "STOP for 1 hour"
- in messenger : "START

Any other command will Raise an Error wich will be sended to user on messenger

monitor()
The main function that returns True if the bot we are monitoring should trade and False otherwise.

This is the function one should call in an other script to monitor something.

class my_tools.monitoring.bot.**State**(*value*)
Bases: enum.Enum

The diffrent possible state of Monitor_Bot:

- trading

- stand_by

stand_by = 2

trading = 1

my_tools.monitoring.message module

A script aimed to send and retreive messages via messenger

my_tools.monitoring.message.**connect**(*config*)
Set up of the bot

This function launch a selenium driver and connect to messenger

class my_tools.monitoring.message.**monitoring_config**
Bases: object

A class with username and password for messenger.

It includes :

- USR : messenger username
- PWD : messenger password

my_tools.monitoring.message.**retreive_messages**(*browser*)
Retreive the last message send by user

my_tools.monitoring.message.**send_message**(*browser*, *message*)
Send the message in "message" string

Module contents

1.1.2 Module contents

CHAPTER

TWO

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

m

INDEX