babao Documentation

Release 0.2

JeanMax

Oct 05, 2018

Contents:

CHAPTER 1

babao package

1.1 Subpackages

1.1.1 babao.inputs package

1.1.1.1 Subpackages

babao.inputs.ledger package

Submodules

babao.inputs.ledger.fakeLedgerInput module

Handle logging in database all our fake transactions (dry-run)

class babao.inputs.ledger.fakeLedgerInput.**ABCFakeLedgerInput**(*log_to_file=True*, *temp=False*)

Bases: *babao.inputs.ledger.ledgerInputBase.ABCLedgerInput*

Base class for any fake ledger

asset
Overide this method with the desired CryptoEnum / QuoteEnum ex: self.asset = CryptoEnum.XBT

buy(*ledger*, *volume_spent*, *price*, *timestamp=None*)
Buy with the current ledger asset the asset of the given ´ledger´

(If the current ledger is a quote, this is a buy) ´volume_spent´ quantity spent (including fees)

deposit(*ledger*, *volume*, *timestamp=None*)
Deposit from the current ledger to the given ´ledger´

fetch()
Return a time-serie DataFrame fetched from the internets

This data will be stored on database for later use (and eventual resampling). Data can be continuous. Index must be nanosecond timestamps.

logTransaction(*typ*, *volume*, *refid*, *fee=0*, *product=0*, *timestamp=None*)
Log transaction in database if ´timestamp´ is not given, the current time will be used

This should'nt be used outside of this class

sell(*ledger*, *volume_spent*, *price*, *timestamp=None*)
Buy with the asset of the given ´ledger´ the current ledger asset

(If the current ledger is a quote, this is a sell) ´volume_spent´ quantity spent (including fees)

withdraw(*ledger*, *volume*, *timestamp=None*)
Withdraw from the current ledger to the given ´ledger´

babao.inputs.ledger.krakenLedgerInput module

Handle logging in database all our real transactions on kraken api (wet-run) Also fetch from kraken api the history of your past transactions

class babao.inputs.ledger.krakenLedgerInput.**ABCKrakenLedgerInput**(*log_to_file=True*, *unused_temp=False*)
Bases: *babao.inputs.ledger.ledgerInputBase.ABCLedgerInput*, *babao.inputs.krakenInputBase.ABCKrakenInput*

Base class for any kraken ledger

buy(*ledger*, *volume_spent*, *price*, *timestamp=None*)
Buy with the current ledger asset the asset of the given ´ledger´

(If the current ledger is a quote, this is a buy) ´volume_spent´ quantity spent (including fees)

deposit(*ledger*, *volume*, *timestamp=None*)
Deposit from the current ledger to the given ´ledger´

fetch()
Return a time-serie DataFrame fetched from the internets

This data will be stored on database for later use (and eventual resampling). Data can be continuous. Index must be nanosecond timestamps.

sell(*ledger*, *volume_spent*, *price*, *timestamp=None*)
Buy with the asset of the given ´ledger´ the current ledger asset

(If the current ledger is a quote, this is a sell) ´volume_spent´ quantity spent (including fees)

withdraw(*ledger*, *volume*, *timestamp=None*)
Withdraw from the current ledger to the given ´ledger´

babao.inputs.ledger.ledgerInputBase module

Handle logging in database all our transactions

TODO: It is really not obvious how you're gonna link the transaction across various ledgers... kraken doesn't give you anything else than an "order id", so you still have to iterate over all entries in all ledgers to find the matching one :/ The good new is, I don't know when we'll need that! Anyway, I'll leave an empty column "product", which reference another ledger; this could be used for later indexing?

class babao.inputs.ledger.ledgerInputBase.**ABCLedgerInput**
Bases: *babao.inputs.inputBase.ABCInput*

Base class for any ledger

asset
Overide this method with the desired CryptoEnum / QuoteEnum ex: self.asset = CryptoEnum.XBT

buy(*ledger*, *volume_spent*, *price*, *timestamp=None*)
Buy with the current ledger asset the asset of the given ´ledger´

(If the current ledger is a quote, this is a buy) ´volume_spent´ quantity spent (including fees)

deposit(*ledger*, *volume*, *timestamp=None*)
Deposit from the current ledger to the given ´ledger´

fillMissing(*resampled_data*)
Fill missing values (np.nan/np.inf) in ´resampled_data´

raw_columns = ['volume', 'balance', 'fee', 'refid', 'type', 'product']

resampled_columns = ['balance']

sell(*ledger*, *volume_spent*, *price*, *timestamp=None*)
Buy with the asset of the given ´ledger´ the current ledger asset

(If the current ledger is a quote, this is a sell) ´volume_spent´ quantity spent (including fees)

withdraw(*ledger*, *volume*, *timestamp=None*)
Withdraw from the current ledger to the given ´ledger´

babao.inputs.ledger.ledgerManager module

Manage all the ledgers and give some utils functions to check balance or buy/sell

babao.inputs.ledger.ledgerManager.**buy**(*crypto_enum*, *volume*)
Buy the given ´volume´ of ´crypto_enum´

babao.inputs.ledger.ledgerManager.**buyOrSell**(*action_enum*, *crypto_enum*, *volume=None*)
Decide wether to buy or sell (or not) ´volume´ the ´crypto_enum´ based on the given ´action_enum´

It will consider the current ´ledger.BALANCE´, and evenutally update it.

babao.inputs.ledger.ledgerManager.**gameOver**()
Check if you're broke

babao.inputs.ledger.ledgerManager.**getBalanceInQuote**(*crypto_enum*)
Convert to quote the balance of the LEDGERS[crypto_enum]

babao.inputs.ledger.ledgerManager.**getGlobalBalanceInQuote**()
Sum and convert to quote the balance of all your LEDGERS

babao.inputs.ledger.ledgerManager.**getLastTx**()
Return the timestamp of the last transaction in all LEDGERS

babao.inputs.ledger.ledgerManager.**initLedgers**()
Instantiate all the ledgers and corresponding trade-inputs needed by conf.CRYPTOS / conf.QUOTE

babao.inputs.ledger.ledgerManager.**sell**(*crypto_enum*, *volume*)
Sell the given ´volume´ of ´crypto_enum´

Module contents

babao.inputs.trades package

Submodules

babao.inputs.trades.krakenTradesInput module

Concrete class for kraken trade inputs

We could have defined all these with the following commentend out snippet, but for explicitness reasons we'll keep them this way. This also allows linter to understand what's going on.

class babao.inputs.trades.krakenTradesInput.**KrakenTradesBCHEURInput**
Bases: *babao.inputs.trades.krakenTradesInputBase.ABCKrakenTradesInput*

Kraken trade input for CHE crypto vs R quote

crypto = -1

pair = 'BCHEUR'

quote = 2

class babao.inputs.trades.krakenTradesInput.**KrakenTradesBCHUSDInput**
Bases: *babao.inputs.trades.krakenTradesInputBase.ABCKrakenTradesInput*

Kraken trade input for CHU crypto vs D quote

crypto = -1

pair = 'BCHUSD'

quote = 5

class babao.inputs.trades.krakenTradesInput.**KrakenTradesDASHEURInput**
Bases: *babao.inputs.trades.krakenTradesInputBase.ABCKrakenTradesInput*

Kraken trade input for ASH crypto vs UR quote

crypto = -2

pair = 'DASHEUR'

quote = 2

class babao.inputs.trades.krakenTradesInput.**KrakenTradesDASHUSDInput**
Bases: *babao.inputs.trades.krakenTradesInputBase.ABCKrakenTradesInput*

Kraken trade input for ASH crypto vs SD quote

crypto = -2

pair = 'DASHUSD'

quote = 5

class babao.inputs.trades.krakenTradesInput.**KrakenTradesEOSEURInput**
Bases: *babao.inputs.trades.krakenTradesInputBase.ABCKrakenTradesInput*

Kraken trade input for OSE crypto vs R quote

crypto = -3

pair = 'EOSEUR'

quote = 2

class babao.inputs.trades.krakenTradesInput.**KrakenTradesEOSUSDInput**
Bases: *babao.inputs.trades.krakenTradesInputBase.ABCKrakenTradesInput*

Kraken trade input for OSU crypto vs D quote

crypto = -3

pair = 'EOSUSD'

quote = 5

class babao.inputs.trades.krakenTradesInput.**KrakenTradesGNOEURInput**
Bases: *babao.inputs.trades.krakenTradesInputBase.ABCKrakenTradesInput*

Kraken trade input for NOE crypto vs R quote

crypto = -4

pair = 'GNOEUR'

quote = 2

class babao.inputs.trades.krakenTradesInput.**KrakenTradesGNOUSDInput**
Bases: *babao.inputs.trades.krakenTradesInputBase.ABCKrakenTradesInput*

Kraken trade input for NOU crypto vs D quote

crypto = -4

pair = 'GNOUSD'

quote = 5

class babao.inputs.trades.krakenTradesInput.**KrakenTradesXETCZCADInput**
Bases: *babao.inputs.trades.krakenTradesInputBase.ABCKrakenTradesInput*

Kraken trade input for ETC crypto vs CAD quote

crypto = -5

pair = 'XETCZCAD'

quote = 1

class babao.inputs.trades.krakenTradesInput.**KrakenTradesXETCZEURInput**
Bases: *babao.inputs.trades.krakenTradesInputBase.ABCKrakenTradesInput*

Kraken trade input for ETC crypto vs EUR quote

crypto = -5

pair = 'XETCZEUR'

quote = 2

class babao.inputs.trades.krakenTradesInput.**KrakenTradesXETCZGBPInput**
Bases: *babao.inputs.trades.krakenTradesInputBase.ABCKrakenTradesInput*

Kraken trade input for ETC crypto vs GBP quote

crypto = -5

pair = 'XETCZGBP'

quote = 3

class babao.inputs.trades.krakenTradesInput.**KrakenTradesXETCZJPYInput**
Bases: *babao.inputs.trades.krakenTradesInputBase.ABCKrakenTradesInput*

Kraken trade input for ETC crypto vs JPY quote

crypto = -5

pair = 'XETCZJPY'

quote = 4

class babao.inputs.trades.krakenTradesInput.**KrakenTradesXETCZUSDInput**
Bases: *babao.inputs.trades.krakenTradesInputBase.ABCKrakenTradesInput*

Kraken trade input for ETC crypto vs USD quote

crypto = -5

pair = 'XETCZUSD'

quote = 5

class babao.inputs.trades.krakenTradesInput.**KrakenTradesXETHZCADInput**
Bases: *babao.inputs.trades.krakenTradesInputBase.ABCKrakenTradesInput*

Kraken trade input for ETH crypto vs CAD quote

crypto = -6

pair = 'XETHZCAD'

quote = 1

class babao.inputs.trades.krakenTradesInput.**KrakenTradesXETHZEURInput**
Bases: *babao.inputs.trades.krakenTradesInputBase.ABCKrakenTradesInput*

Kraken trade input for ETH crypto vs EUR quote

crypto = -6

pair = 'XETHZEUR'

quote = 2

class babao.inputs.trades.krakenTradesInput.**KrakenTradesXETHZGBPInput**
Bases: *babao.inputs.trades.krakenTradesInputBase.ABCKrakenTradesInput*

Kraken trade input for ETH crypto vs GBP quote

crypto = -6

pair = 'XETHZGBP'

quote = 3

class babao.inputs.trades.krakenTradesInput.**KrakenTradesXETHZJPYInput**
Bases: *babao.inputs.trades.krakenTradesInputBase.ABCKrakenTradesInput*

Kraken trade input for ETH crypto vs JPY quote

crypto = -6

pair = 'XETHZJPY'

quote = 4

class babao.inputs.trades.krakenTradesInput.**KrakenTradesXETHZUSDInput**
Bases: *babao.inputs.trades.krakenTradesInputBase.ABCKrakenTradesInput*

Kraken trade input for ETH crypto vs USD quote

crypto = -6

pair = 'XETHZUSD'

quote = 5

class babao.inputs.trades.krakenTradesInput.**KrakenTradesXLTCZCADInput**
Bases: *babao.inputs.trades.krakenTradesInputBase.ABCKrakenTradesInput*

Kraken trade input for LTC crypto vs CAD quote

crypto = -7

pair = 'XLTCZCAD'

quote = 1

class babao.inputs.trades.krakenTradesInput.**KrakenTradesXLTCZEURInput**
Bases: *babao.inputs.trades.krakenTradesInputBase.ABCKrakenTradesInput*

Kraken trade input for LTC crypto vs EUR quote

crypto = -7

pair = 'XLTCZEUR'

quote = 2

class babao.inputs.trades.krakenTradesInput.**KrakenTradesXLTCZGBPInput**
Bases: *babao.inputs.trades.krakenTradesInputBase.ABCKrakenTradesInput*

Kraken trade input for LTC crypto vs GBP quote

crypto = -7

pair = 'XLTCZGBP'

quote = 3

class babao.inputs.trades.krakenTradesInput.**KrakenTradesXLTCZJPYInput**
Bases: *babao.inputs.trades.krakenTradesInputBase.ABCKrakenTradesInput*

Kraken trade input for LTC crypto vs JPY quote

crypto = -7

pair = 'XLTCZJPY'

quote = 4

class babao.inputs.trades.krakenTradesInput.**KrakenTradesXLTCZUSDInput**
Bases: *babao.inputs.trades.krakenTradesInputBase.ABCKrakenTradesInput*

Kraken trade input for LTC crypto vs USD quote

crypto = -7

pair = 'XLTCZUSD'

quote = 5

class babao.inputs.trades.krakenTradesInput.**KrakenTradesXREPZCADInput**
Bases: *babao.inputs.trades.krakenTradesInputBase.ABCKrakenTradesInput*

Kraken trade input for REP crypto vs CAD quote

crypto = -8

pair = 'XREPZCAD'

quote = 1

class babao.inputs.trades.krakenTradesInput.**KrakenTradesXREPZEURInput**
Bases: *babao.inputs.trades.krakenTradesInputBase.ABCKrakenTradesInput*

Kraken trade input for REP crypto vs EUR quote

crypto = -8

pair = 'XREPZEUR'

quote = 2

class babao.inputs.trades.krakenTradesInput.**KrakenTradesXREPZGBPInput**
Bases: *babao.inputs.trades.krakenTradesInputBase.ABCKrakenTradesInput*

Kraken trade input for REP crypto vs GBP quote

crypto = -8

pair = 'XREPZGBP'

quote = 3

class babao.inputs.trades.krakenTradesInput.**KrakenTradesXREPZJPYInput**
Bases: *babao.inputs.trades.krakenTradesInputBase.ABCKrakenTradesInput*

Kraken trade input for REP crypto vs JPY quote

crypto = -8

pair = 'XREPZJPY'

quote = 4

class babao.inputs.trades.krakenTradesInput.**KrakenTradesXREPZUSDInput**
Bases: *babao.inputs.trades.krakenTradesInputBase.ABCKrakenTradesInput*

Kraken trade input for REP crypto vs USD quote

crypto = -8

pair = 'XREPZUSD'

quote = 5

class babao.inputs.trades.krakenTradesInput.**KrakenTradesXXBTZCADInput**
Bases: *babao.inputs.trades.krakenTradesInputBase.ABCKrakenTradesInput*

Kraken trade input for XBT crypto vs CAD quote

crypto = -9

pair = 'XXBTZCAD'

quote = 1

class babao.inputs.trades.krakenTradesInput.**KrakenTradesXXBTZEURInput**
Bases: *babao.inputs.trades.krakenTradesInputBase.ABCKrakenTradesInput*

Kraken trade input for XBT crypto vs EUR quote

crypto = -9

pair = 'XXBTZEUR'

quote = 2

class babao.inputs.trades.krakenTradesInput.**KrakenTradesXXBTZGBPInput**
Bases: *babao.inputs.trades.krakenTradesInputBase.ABCKrakenTradesInput*

Kraken trade input for XBT crypto vs GBP quote

crypto = -9

pair = 'XXBTZGBP'

quote = 3

class babao.inputs.trades.krakenTradesInput.**KrakenTradesXXBTZJPYInput**
Bases: *babao.inputs.trades.krakenTradesInputBase.ABCKrakenTradesInput*

Kraken trade input for XBT crypto vs JPY quote

crypto = -9

pair = 'XXBTZJPY'

quote = 4

class babao.inputs.trades.krakenTradesInput.**KrakenTradesXXBTZUSDInput**
Bases: *babao.inputs.trades.krakenTradesInputBase.ABCKrakenTradesInput*

Kraken trade input for XBT crypto vs USD quote

crypto = -9

pair = 'XXBTZUSD'

quote = 5

class babao.inputs.trades.krakenTradesInput.**KrakenTradesXXLMZCADInput**
Bases: *babao.inputs.trades.krakenTradesInputBase.ABCKrakenTradesInput*

Kraken trade input for XLM crypto vs CAD quote

crypto = -10

pair = 'XXLMZCAD'

quote = 1

class babao.inputs.trades.krakenTradesInput.**KrakenTradesXXLMZEURInput**
Bases: *babao.inputs.trades.krakenTradesInputBase.ABCKrakenTradesInput*

Kraken trade input for XLM crypto vs EUR quote

crypto = -10

pair = 'XXLMZEUR'

quote = 2

class babao.inputs.trades.krakenTradesInput.**KrakenTradesXXLMZGBPInput**
Bases: *babao.inputs.trades.krakenTradesInputBase.ABCKrakenTradesInput*

Kraken trade input for XLM crypto vs GBP quote

crypto = -10

pair = 'XXLMZGBP'

quote = 3

class babao.inputs.trades.krakenTradesInput.**KrakenTradesXXLMZJPYInput**
Bases: *babao.inputs.trades.krakenTradesInputBase.ABCKrakenTradesInput*

Kraken trade input for XLM crypto vs JPY quote

crypto = -10

pair = 'XXLMZJPY'

quote = 4

class babao.inputs.trades.krakenTradesInput.**KrakenTradesXXLMZUSDInput**
Bases: *babao.inputs.trades.krakenTradesInputBase.ABCKrakenTradesInput*

Kraken trade input for XLM crypto vs USD quote

crypto = -10

pair = 'XXLMZUSD'

quote = 5

class babao.inputs.trades.krakenTradesInput.**KrakenTradesXXMRZCADInput**
Bases: *babao.inputs.trades.krakenTradesInputBase.ABCKrakenTradesInput*

Kraken trade input for XMR crypto vs CAD quote

crypto = -11

pair = 'XXMRZCAD'

quote = 1

class babao.inputs.trades.krakenTradesInput.**KrakenTradesXXMRZEURInput**
Bases: *babao.inputs.trades.krakenTradesInputBase.ABCKrakenTradesInput*

Kraken trade input for XMR crypto vs EUR quote

crypto = -11

pair = 'XXMRZEUR'

quote = 2

class babao.inputs.trades.krakenTradesInput.**KrakenTradesXXMRZGBPInput**
Bases: *babao.inputs.trades.krakenTradesInputBase.ABCKrakenTradesInput*

Kraken trade input for XMR crypto vs GBP quote

crypto = -11

pair = 'XXMRZGBP'

quote = 3

class babao.inputs.trades.krakenTradesInput.**KrakenTradesXXMRZJPYInput**
Bases: *babao.inputs.trades.krakenTradesInputBase.ABCKrakenTradesInput*

Kraken trade input for XMR crypto vs JPY quote

crypto = -11

pair = 'XXMRZJPY'

quote = 4

class babao.inputs.trades.krakenTradesInput.**KrakenTradesXXMRZUSDInput**
Bases: *babao.inputs.trades.krakenTradesInputBase.ABCKrakenTradesInput*

Kraken trade input for XMR crypto vs USD quote

crypto = -11

pair = 'XXMRZUSD'

quote = 5

class babao.inputs.trades.krakenTradesInput.**KrakenTradesXXRPZCADInput**
Bases: *babao.inputs.trades.krakenTradesInputBase.ABCKrakenTradesInput*

Kraken trade input for XRP crypto vs CAD quote

crypto = -12

pair = 'XXRPZCAD'

quote = 1

class babao.inputs.trades.krakenTradesInput.**KrakenTradesXXRPZEURInput**
Bases: *babao.inputs.trades.krakenTradesInputBase.ABCKrakenTradesInput*

Kraken trade input for XRP crypto vs EUR quote

crypto = -12

pair = 'XXRPZEUR'

quote = 2

class babao.inputs.trades.krakenTradesInput.**KrakenTradesXXRPZGBPInput**
Bases: *babao.inputs.trades.krakenTradesInputBase.ABCKrakenTradesInput*

Kraken trade input for XRP crypto vs GBP quote

crypto = -12

pair = 'XXRPZGBP'

quote = 3

class babao.inputs.trades.krakenTradesInput.**KrakenTradesXXRPZJPYInput**
Bases: *babao.inputs.trades.krakenTradesInputBase.ABCKrakenTradesInput*

Kraken trade input for XRP crypto vs JPY quote

crypto = -12

pair = 'XXRPZJPY'

quote = 4

class babao.inputs.trades.krakenTradesInput.**KrakenTradesXXRPZUSDInput**
Bases: *babao.inputs.trades.krakenTradesInputBase.ABCKrakenTradesInput*

Kraken trade input for XRP crypto vs USD quote

crypto = -12

pair = 'XXRPZUSD'

quote = 5

class babao.inputs.trades.krakenTradesInput.**KrakenTradesXZECZCADInput**
Bases: *babao.inputs.trades.krakenTradesInputBase.ABCKrakenTradesInput*

Kraken trade input for ZEC crypto vs CAD quote

crypto = -13

pair = 'XZECZCAD'

quote = 1

class babao.inputs.trades.krakenTradesInput.**KrakenTradesXZECZEURInput**
Bases: *babao.inputs.trades.krakenTradesInputBase.ABCKrakenTradesInput*

Kraken trade input for ZEC crypto vs EUR quote

crypto = -13

pair = 'XZECZEUR'

quote = 2

class babao.inputs.trades.krakenTradesInput.**KrakenTradesXZECZGBPInput**
Bases: *babao.inputs.trades.krakenTradesInputBase.ABCKrakenTradesInput*

Kraken trade input for ZEC crypto vs GBP quote

crypto = -13

pair = 'XZECZGBP'

quote = 3

class babao.inputs.trades.krakenTradesInput.**KrakenTradesXZECZJPYInput**
Bases: *babao.inputs.trades.krakenTradesInputBase.ABCKrakenTradesInput*

Kraken trade input for ZEC crypto vs JPY quote

crypto = -13

pair = 'XZECZJPY'

quote = 4

class babao.inputs.trades.krakenTradesInput.**KrakenTradesXZECZUSDInput**
Bases: *babao.inputs.trades.krakenTradesInputBase.ABCKrakenTradesInput*

Kraken trade input for ZEC crypto vs USD quote

crypto = -13

pair = 'XZECZUSD'

quote = 5

babao.inputs.trades.krakenTradesInputBase module

Module containing base class for any kraken trades input

class babao.inputs.trades.krakenTradesInputBase.**ABCKrakenTradesInput**
Bases: *babao.inputs.trades.tradesInputBase.ABCTradesInput*, *babao.inputs.krakenInputBase.ABCKrakenInput*

Base class for any kraken trades input

fetch()
Return a time-serie DataFrame fetched from the internets

This data will be stored on database for later use (and eventual resampling). Data can be continuous. Index must be nanosecond timestamps.

pair
Overide this method with the desired asset pair as string ex: self.pair = "XXBTZEUR"

babao.inputs.trades.tradesInputBase module

Module containing base class for any trades input

class babao.inputs.trades.tradesInputBase.**ABCTradesInput**
Bases: *babao.inputs.inputBase.ABCInput*

Base class for any trades input

crypto
Overide this method with the desired CryptoEnum ex: self.crypto = CryptoEnum.XBT

fetch()
Return a time-serie DataFrame fetched from the internets

This data will be stored on database for later use (and eventual resampling). Data can be continuous. Index must be nanosecond timestamps.

fillMissing(*resampled_data*)
Fill missing values (np.nan/np.inf) in ´resampled_data´

quote
Overide this method with the desired QuoteEnum ex: self.quote = QuoteEnum.EUR

raw_columns = ['price', 'volume']

resampled_columns = ['open', 'high', 'low', 'close', 'vwap', 'volume', 'count']

Module contents

1.1.1.2 Submodules

1.1.1.3 babao.inputs.inputBase module

Base class for any input

class babao.inputs.inputBase.**ABCInput**
Bases: abc.ABC

Base class for any input

Your subclass should at least implement: * fetch : self -> DataFrame * raw_columns : List[str]

And eventually: (if you want self.resample to works) * _resample : self -> DataFrame -> DataFrame * fillMissing : self -> DataFrame -> DataFrame * resampled_columns : List[str]

(cf. specific method doc-string in this class)

cache(*fresh_data=None*, *since=None*, *till=None*)
Save some data to cache

If ´fresh_data´ is given, append it to cache, otherwise read in database from ´since´ to ´till´ and cache it

fetch()
Return a time-serie DataFrame fetched from the internets

This data will be stored on database for later use (and eventual resampling). Data can be continuous. Index must be nanosecond timestamps.

fillMissing(*resampled_data*)
Fill missing values (np.nan/np.inf) in ´resampled_data´

raw_columns
The columns names of your raw data (as fetched and stored in database)

read(*since=None*, *till=None*)
Read data in database or cache from ´since´ to ´till´

refreshCache()
Make sure the cache is up to date

resample(*raw_data*)
Return the DataFrame ´raw_data´ as a continuous time-serie

This is a wrapper around _resample and fillMissing

resampled_columns
The columns names of your resampled data (from raw data)

updateCurrentRow(*current_row=None*, *timestamp=None*)
Update the property self.current_row, useful for time travel

write(*raw_data*)
Write the given raw_data to the database, and cache it if needed

babao.inputs.inputBase.**resampleSerie**(*s*)
Call Serie.resample on s with preset parameters (the serie's index must be datetime)

1.1.1.4 babao.inputs.inputManager module

Common interface to inputs to call methods on all of them at once

babao.inputs.inputManager.**fetchInputs**()
Fetch all the INPUTS in a pool thread

The raw data resulting is then wrote to database.

babao.inputs.inputManager.**readInputs**(*input_list: Optional[List[babao.inputs.inputBase.ABCInput]] = None*, *since=None*)
Read all INPUTS from ´since´ and resample them with matching time base

The return is one dataframe containing all concatened columns (so they will be renamed with the input name as prefix

babao.inputs.inputManager.**refreshInputs**(*input_list: Optional[List[babao.inputs.inputBase.ABCInput]] = None*)
Make sure the cache is up to date on the given inputs (or all the INPUTS)

babao.inputs.inputManager.**timeTravel**(*timestamp*)
Travel to the specified timestamp, for simulation purposes

1.1.1.5 babao.inputs.krakenInputBase module

This module define the base class and methods for kraken inputs

class babao.inputs.krakenInputBase.**ABCKrakenInput**
Bases: *babao.inputs.inputBase.ABCInput*

Base class for any kraken input

fetch()
Return a time-serie DataFrame fetched from the internets

This data will be stored on database for later use (and eventual resampling). Data can be continuous. Index must be nanosecond timestamps.

1.1.1.6 Module contents

1.1.2 babao.models package

1.1.2.1 Subpackages

babao.models.tree package

Submodules

babao.models.tree.extremaModel module

babao.models.tree.macdModel module

babao.models.tree.tendencyModel module

Module contents

1.1.2.2 Submodules

1.1.2.3 babao.models.modelBase module

1.1.2.4 babao.models.modelManager module

1.1.2.5 babao.models.rootModel module

1.1.2.6 Module contents

1.1.3 babao.utils package

1.1.3.1 Submodules

1.1.3.2 babao.utils.date module

Some utils functions for date handling and time traveling

class babao.utils.date.**TimeTraveler**
Bases: object

Class handling time travel tricks

getTime(*force=False*)
Return the current time in nanoseconds

Used for time traveling purpose, so this might be a date in the past matching the current simulation state, unless ´force´ is set to True.

nowMinus(*years=0*, *weeks=0*, *days=0*, *hours=0*, *minutes=0*)
Return the current timestamp (nanoseconds) minus the given parameters

This will take into account time traveling tricks.

setTime(*now*)
Set time to the given ´now´ nanoseconds

Used for time traveling purpose

babao.utils.date.**nanoToSec**(*nano*)
Convert nanoseconds to seconds

babao.utils.date.**secToNano**(*sec*)
Convert seconds to nanoseconds Just trying to avoid float rounding...

babao.utils.date.**toDatetime**(*df*)
Convert the index of the given dataframe to datetime

Also works directly on a dataframe index.

babao.utils.date.**toStr**(*t*)
Return the string representation of timestamp

´t´ can be a nanoseconds timestamp, or a panda datetime object.

babao.utils.date.**toTimestamp**(*df*)
Convert the index of the given dataframe to nanoseconds

Also works directly on a dataframe index.

1.1.3.3 babao.utils.enum module

Enums describing positions and assets.

class babao.utils.enum.**ActionEnum**
Bases: enum.Enum

Enum describing a transaction action

BUY = 1

DEPOSIT = 2

FEE = -3

HODL = 0

SELL = -1

WITHDRAW = -2

class babao.utils.enum.**CryptoEnum**
Bases: enum.Enum

Enum describing a crypto asset

BCH = -1

DASH = -2

EOS = -3

ETC = -5

ETH = -6

GNO = -4

LTC = -7

REP = -8

XBT = -9

XLM = -10

XMR = -11

XRP = -12

ZEC = -13

class babao.utils.enum.**QuoteEnum**
Bases: enum.Enum

Enum describing a quote asset

CAD = 1

EUR = 2

GBP = 3

JPY = 4

USD = 5

class babao.utils.enum.**TradeEnum**
Bases: enum.Enum

Enum describing a transaction on a given asset

BUY_BCH = 1

BUY_DASH = 2

BUY_EOS = 3

BUY_ETC = 5

BUY_ETH = 6

BUY_GNO = 4

BUY_LTC = 7

BUY_REP = 8

BUY_XBT = 9

BUY_XLM = 10

BUY_XMR = 11

BUY_XRP = 12

BUY_ZEC = 13

HODL = 0

SELL_BCH = -1

SELL_DASH = -2

SELL_EOS = -3

SELL_ETC = -5

SELL_ETH = -6

SELL_GNO = -4

SELL_LTC = -7

SELL_REP = -8

SELL_XBT = -9

SELL_XLM = -10

SELL_XMR = -11

SELL_XRP = -12

SELL_ZEC = -13

babao.utils.enum.**cryptoAndActionTotrade**(*crypto_enum_val*, *action_enum_val*)
Convert a crypto enum and an action enum to the matching trade enum

babao.utils.enum.**floatToTrade**(*f*)
Round a float value to the nearest trade enum

babao.utils.enum.**tradeToAction**(*trade_enum_val*)
Extract an action enum from a trade enum

babao.utils.enum.**tradeToCrypto**(*trade_enum_val*)
Extract a crypto enum from a trade enum

1.1.3.4 babao.utils.file module

Some utils functions for hdf handling

babao.utils.file.**closeStore**()
Close the hdf database

babao.utils.file.**delete**(*frame*)
Remove the given ´frame´ entry (key) from the hdf database

Thread Safe!

babao.utils.file.**getLastRows**(*frame*, *nrows*)
Return ´nrows´ rows from a ´frame´ (key) in the hdf database as a DataFrame

babao.utils.file.**initStore**(*filename*)
Open the hdf database from ´filename´

babao.utils.file.**maintenance**()
Maintenance routine on the hdf database

Create table index for each table, and make sure everything is sorted.

babao.utils.file.**read**(*frame*, *where=None*)
Read a ´frame´ (key) from the hdf database

´where´ can be use to specify search criteria. Thread Safe!

babao.utils.file.**setLock**(*lock*)
Store the given ´lock´ object for later use in database processing

babao.utils.file.**write**(*frame*, *df*)
Append the given ´df´ dataframe to the ´frame´ entry (key) in the hdf database

Thread Safe!

1.1.3.5 babao.utils.indicators module

Various indicators which can be added to any serie

//www.quantinsti.com/blog/build-technical-indicators-in-python

babao.utils.indicators.**ewma**(*serie*, *look_back_delay*)
Exponentially-weighted Moving Average

babao.utils.indicators.**get**(*df*, *columns*)
Add indicators specified by columns to the given df

Expected ´columns´ format: ["sma_vwap_42", "ewma_volume_12"]

babao.utils.indicators.**macd**(*serie*, *fast_delay*, *slow_delay*, *signal_delay*, *full=False*)
Moving Average Convergence/Divergence Oscillator

babao.utils.indicators.**ppo**(*serie*, *fast_delay*, *slow_delay*, *signal_delay*, *full=False*)
Percentage Price Oscillator

Same as macd, but we do (a-b)/b instead of a-b, so the final value does not depend on input scale (it's a percentage!)

babao.utils.indicators.**sma**(*serie*, *look_back_delay*)
Simple Moving Average

1.1.3.6 babao.utils.lock module

Some utils functions for lock file handling

babao.utils.lock.**isLocked**(*lockfile*)
Check if the ´lockfile´ exists

babao.utils.lock.**tryLock**(*lockfile*)
Create the given ´lockfile´

Return false if it already exists

babao.utils.lock.**tryUnlock**(*lockfile*)
Remove the given ´lockfile´

Return false if it doesn't exist

1.1.3.7 babao.utils.log module

Some utils functions for logging

babao.utils.log.**debug**(**args*)
Log a debug (magenta)

babao.utils.log.**error**(**args*)
Log an error (red)

babao.utils.log.**info**(**args*)
Log a simple message (blue)

babao.utils.log.**initLogLevel**(*verbose*, *quiet*)
Initialize log level based on verbose flag

babao.utils.log.**setLock**(*lock*)
Store the given ´lock´ object for later use in logging

babao.utils.log.**warning**(**args*)
Log a warning (yellow)

1.1.3.8 babao.utils.scale module

Scaler

class babao.utils.scale.**Scaler**
Bases: object

Basic min/max scaler

fit(*arr*)
Init scaler

scale(*arr*)
Scale features before train/predict

scaleFit(*arr*)
Scale n Fit

unscale(*arr*)
Unscale features after train/predict

1.1.3.9 babao.utils.signal module

Signal handling

babao.utils.signal.**catchSignal**()
Catch signal INT/TERM, so we won't exit while playing with data files

babao.utils.signal.**restoreSignal**()
Restore the previous signal handler

1.1.3.10 Module contents

1.2 Submodules

1.3 babao.arg module

1.4 babao.babao module

1.5 babao.commands module

1.6 babao.config module

Here we'll handle the config file and the various file/dir paths

babao.config.**readConfigFile**(*cmd_name='dry-run'*)
Read config file and initialize global config variables

1.7 babao.graph module

1.8 Module contents

CHAPTER 2

Indices and tables

- genindex
- modindex
- search

Python Module Index

b

Index

L

M

N

P

R

S

T

U

W

X

Z