python-tradingview-ta Documentation

deathlyface (Brian)

Apr 29, 2023

Contents

TradingView_TA is an unofficial Python API wrapper to retrieve technical analysis from TradingView.

This documentation will help you to understand and use TradingView-TA.

CHAPTER 1

Getting Started

This guide will help you understand the basics of TradingView_TA package.

1.1 Requirements

- Python 3.6 or newer
- Internet access

1.2 Installation

TradingView_TA is available on PyPI.

```
pip install tradingview-ta
```

1.3 Quick Start

```
from tradingview_ta import TA_Handler, Interval, Exchange

tesla = TA_Handler(
    symbol="TSLA",
    screener="america",
    exchange="NASDAQ",
    interval=Interval.INTERVAL_1_DAY
)
print(tesla.get_analysis().summary)
# Example output: {"RECOMMENDATION": "BUY", "BUY": 8, "NEUTRAL": 6, "SELL": 3}
```

CHAPTER 2

Usage

Note: Please install or update TradingView_TA to the latest version. Please read the getting started guide before continuing.

Warning: TradingView_TA older than v3.2.0 is no longer supported. Please update using `pip install tradingview_ta --upgrade`.

2.1 Importing TradingView_TA

```
from tradingview_ta import TA_Handler, Interval, Exchange
import tradingview_ta
```

2.2 Checking the version

Starting from version 3.1.3, you can retrieve the version of TradingView_TA through the `__version__` attribute.

```
print(tradingview_ta.__version__)
# Example output: 3.1.3
```

2.3 Instantiating TA_Handler

```
handler = TA_Handler(
    symbol="",
```

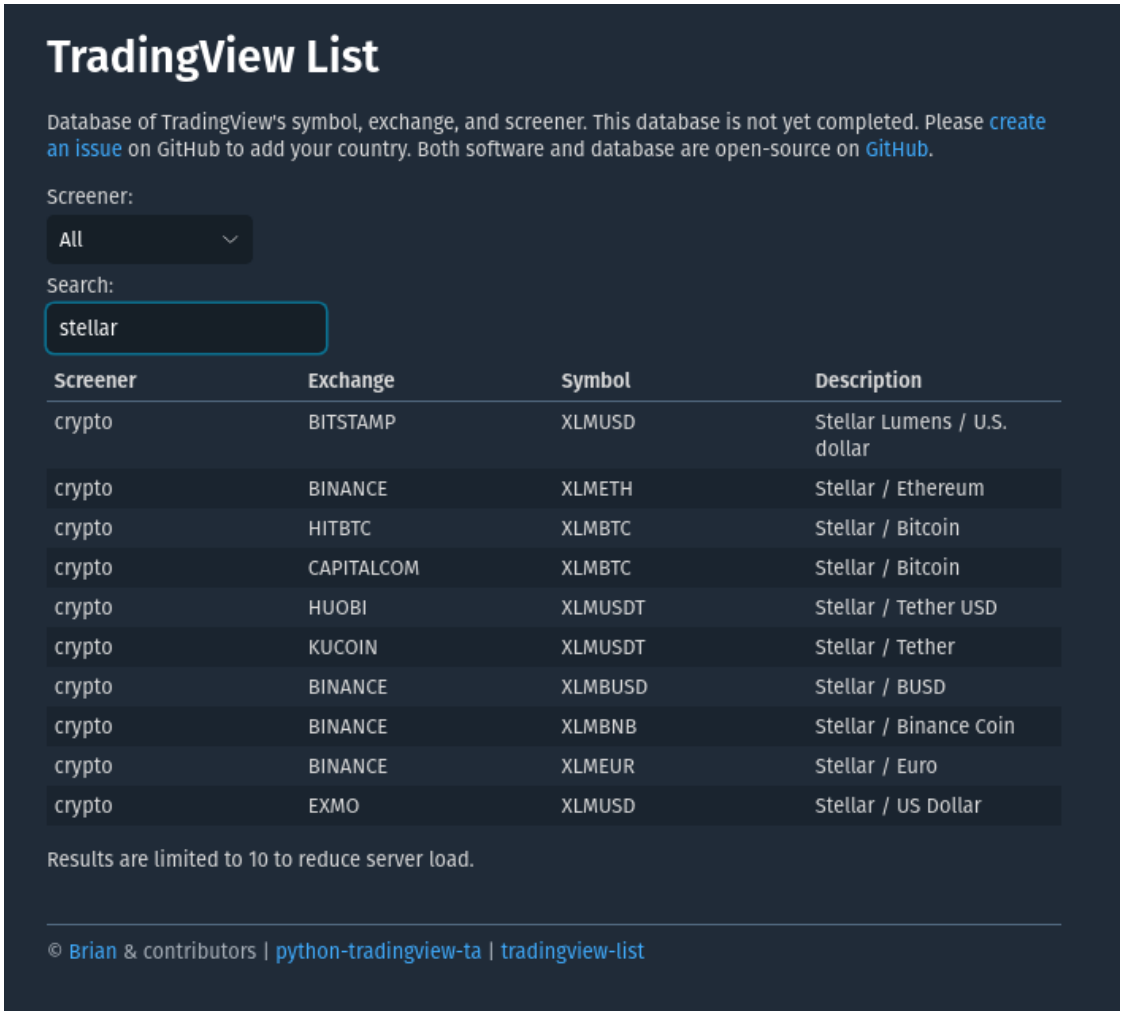
(continues on next page)

(continued from previous page)

```
    exchange="",
    screener="",
    interval="",
    timeout=None
)
```

Parameters:

Tip: You can search on https://tvdb.brianthe.dev to see which symbol, exchange, and screener to use.



- symbol (`str`) – Ticker symbol (e.g., `"AAPL"`, `"TLKM"`, `"USDEUR"`, `"BTCUSDT"`).
- exchange (`str`) – Exchange (e.g., `"nasdaq"`, `"idx"`, `Exchange.FOREX`, `"binance"`).
- screener (`str`) – Screener (e.g., `"america"`, `"indonesia"`, `"forex"`, `"crypto"`).

Note:

- If you're looking for stocks, enter the exchange's country as the screener.
- If you're looking for cryptocurrency, enter `"crypto"` as the screener.

– If you're looking for forex, enter `"forex"` as the screener.

- interval (`str`) – Time frame

Note: Please see the Interval class for available intervals.

```
class Interval:
    INTERVAL_1_MINUTE = "1m"
    INTERVAL_5_MINUTES = "5m"
    INTERVAL_15_MINUTES = "15m"
    INTERVAL_30_MINUTES = "30m"
    INTERVAL_1_HOUR = "1h"
    INTERVAL_2_HOURS = "2h"
    INTERVAL_4_HOURS = "4h"
    INTERVAL_1_DAY = "1d"
    INTERVAL_1_WEEK = "1W"
    INTERVAL_1_MONTH = "1M"
```

- timeout (`float`, optional) – How long to wait (in seconds) for the server to return a response.

2.4 Retrieving the analysis

```
analysis = handler.get_analysis()
```

Note: `analysis` is an instance of Analysis class. It contains information such as the exchange, symbol, screener, interval, local time (datetime.datetime), etc.

Attributes:

- symbol (`str`) – The symbol set earlier.
- exchange (`str`) – The exchange set earlier.
- screener (`str`) – The screener set earlier.
- interval (`str`) – The interval set earlier.
- time (`datetime.datetime`) – The time when the data is retrieved.
- summary (`dict`) – Technical analysis (based on both oscillators and moving averages).

```
# Example
{'RECOMMENDATION': 'BUY', 'BUY': 12, 'SELL': 7, 'NEUTRAL': 9}
```

- oscillators (`dict`) – Technical analysis (based on oscillators).

```
# Example
{'RECOMMENDATION': 'BUY', 'BUY': 2, 'SELL': 1, 'NEUTRAL': 8, 'COMPUTE': {
↪'RSI': 'NEUTRAL', 'STOCH.K': 'NEUTRAL', 'CCI': 'NEUTRAL', 'ADX':
↪'NEUTRAL', 'AO': 'NEUTRAL', 'Mom': 'BUY', 'MACD': 'SELL', 'Stoch.RSI':
↪'NEUTRAL', 'W%R': 'NEUTRAL', 'BBP': 'BUY', 'UO': 'NEUTRAL'}}
```

- moving_averages (`dict`) – Technical analysis (based on moving averages).

```
# Example
{'RECOMMENDATION': 'BUY', 'BUY': 9, 'SELL': 5, 'NEUTRAL': 1, 'COMPUTE': {
↪'EMA10': 'SELL', 'SMA10': 'SELL', 'EMA20': 'SELL', 'SMA20': 'SELL',
↪'EMA30': 'BUY', 'SMA30': 'BUY', 'EMA50': 'BUY', 'SMA50': 'BUY', 'EMA100
↪': 'BUY', 'SMA100': 'BUY', 'EMA200': 'BUY', 'SMA200': 'BUY', 'Ichimoku
↪': 'NEUTRAL', 'VWMA': 'SELL', 'HullMA': 'BUY'}}
```

- indicators (`dict`) – Technical indicators.

```
# Example
{'Recommend.Other': 0, 'Recommend.All': 0.26666667, 'Recommend.MA': 0.
↪53333333, 'RSI': 60.28037412, 'RSI[1]': 58.58364778, 'Stoch.K': 73.
↪80404453, 'Stoch.D': 79.64297643, 'Stoch.K[1]': 78.88160227, 'Stoch.D[1]
↪': 85.97647064, 'CCI20': 46.58442886, 'CCI20[1]': 34.57058796, 'ADX':␣
↪35.78754863, 'ADX+DI': 23.16948389, 'ADX-DI': 13.82449817, 'ADX+DI[1]':␣
↪24.15991909, 'ADX-DI[1]': 13.87125505, 'AO': 6675.72158824, 'AO[1]':␣
↪7283.92420588, 'Mom': 1532.6, 'Mom[1]': 108.29, 'MACD.macd': 2444.
↪73734978, 'MACD.signal': 2606.00138275, 'Rec.Stoch.RSI': 0, 'Stoch.RSI.K
↪': 18.53740187, 'Rec.WR': 0, 'W.R': -26.05634845, 'Rec.BBPower': 0,
↪'BBPower': 295.52055898, 'Rec.UO': 0, 'UO': 55.68311917, 'close': 45326.
↪97, 'EMA5': 45600.06414333, 'SMA5': 45995.592, 'EMA10': 45223.22433151,
↪'SMA10': 45952.635, 'EMA20': 43451.52018338, 'SMA20': 43609.214, 'EMA30
↪': 41908.5944052, 'SMA30': 40880.391, 'EMA50': 40352.10222373, 'SMA50':␣
↪37819.3566, 'EMA100': 40356.09177879, 'SMA100': 38009.7808, 'EMA200':␣
↪39466.50411569, 'SMA200': 45551.36135, 'Rec.Ichimoku': 0, 'Ichimoku.
↪BLine': 40772.57, 'Rec.VWMA': 1, 'VWMA': 43471.81729377, 'Rec.HullMA9':␣
↪-1, 'HullMA9': 45470.37107407, 'Pivot.M.Classic.S3': 11389.27666667,
↪'Pivot.M.Classic.S2': 24559.27666667, 'Pivot.M.Classic.S1': 33010.
↪55333333, 'Pivot.M.Classic.Middle': 37729.27666667, 'Pivot.M.Classic.R1
↪': 46180.55333333, 'Pivot.M.Classic.R2': 50899.27666667, 'Pivot.M.
↪Classic.R3': 64069.27666667, 'Pivot.M.Fibonacci.S3': 24559.27666667,
↪'Pivot.M.Fibonacci.S2': 29590.21666667, 'Pivot.M.Fibonacci.S1': 32698.
↪33666667, 'Pivot.M.Fibonacci.Middle': 37729.27666667, 'Pivot.M.
↪Fibonacci.R1': 42760.21666667, 'Pivot.M.Fibonacci.R2': 45868.33666667,
↪'Pivot.M.Fibonacci.R3': 50899.27666667, 'Pivot.M.Camarilla.S3': 37840.
↪08, 'Pivot.M.Camarilla.S2': 39047.33, 'Pivot.M.Camarilla.S1': 40254.58,
↪'Pivot.M.Camarilla.Middle': 37729.27666667, 'Pivot.M.Camarilla.R1':␣
↪42669.08, 'Pivot.M.Camarilla.R2': 43876.33, 'Pivot.M.Camarilla.R3':␣
↪45083.58, 'Pivot.M.Woodie.S3': 21706.84, 'Pivot.M.Woodie.S2': 25492.42,
↪'Pivot.M.Woodie.S1': 34876.84, 'Pivot.M.Woodie.Middle': 38662.42,
↪'Pivot.M.Woodie.R1': 48046.84, 'Pivot.M.Woodie.R2': 51832.42, 'Pivot.M.
↪Woodie.R3': 61216.84, 'Pivot.M.Demark.S1': 35369.915, 'Pivot.M.Demark.
↪Middle': 38908.9575, 'Pivot.M.Demark.R1': 48539.915, 'open': 44695.95,
↪'P.SAR': 48068.64, 'BB.lower': 37961.23510877, 'BB.upper': 49257.
↪19289123, 'AO[2]': 7524.31223529, 'volume': 32744.424503, 'change': 1.
↪44612354, 'low': 44203.28, 'high': 45560}
```

Tip: Useful indicators:

- Opening price: `analysis.indicators["open"]`
- Closing price: `analysis.indicators["close"]`
- Momentum: `analysis.indicators["Mom"]`
- RSI: `analysis.indicators["RSI"]`
- MACD: `analysis.indicators["MACD.macd"]`

2.5 Retrieving multiple analysis

```
from tradingview_ta import *
analysis = get_multiple_analysis(screener="america", interval=Interval.INTERVAL_1_
↪HOUR, symbols=["nasdaq:tsla", "nyse:docn", "nasdaq:aapl"])
```

Note: You can't mix different screener and interval.

Parameters:

- symbols (`list`) – List of exchange and ticker symbol separated by a colon. Example: ["NASDAQ:TSLA", "NYSE:DOCN"] or ["BINANCE:BTCUSDT", "BITSTAMP:ETHUSD"].
- screener (`str`) – Screener (e.g., `"america"`, `"indonesia"`, `"forex"`, `"crypto"`).
- timeout (`float`, optional) – How long to wait (in seconds) for the server to return a response.
- additional_indicators (`list`, optional) – List of additional indicators to retrieve. Example: `["RSI", "Mom"]`.
- interval (`str`) – Time frame

Note: Please see the Interval class for available intervals.

```
class Interval:
    INTERVAL_1_MINUTE = "1m"
    INTERVAL_5_MINUTES = "5m"
    INTERVAL_15_MINUTES = "15m"
    INTERVAL_30_MINUTES = "30m"
    INTERVAL_1_HOUR = "1h"
    INTERVAL_2_HOURS = "2h"
    INTERVAL_4_HOURS = "4h"
    INTERVAL_1_DAY = "1d"
    INTERVAL_1_WEEK = "1W"
    INTERVAL_1_MONTH = "1M"
```

Note: `get_multiple_analysis()` returns a dictionary with a format of {"EXCHANGE:SYMBOL": Analysis}.

```
# Example
{'NYSE:DOCN': <tradingview_ta.main.Analysis object at 0x7f3a5ba49be0>, 'NASDAQ:TSLA':
↪<tradingview_ta.main.Analysis object at 0x7f3a5ba65040>, 'NASDAQ:AAPL':
↪<tradingview_ta.main.Analysis object at 0x7f3a5ba801c0>}
```

Please use UPPERCASE letters when accessing the dictionary.

If there is no analysis for a certain symbol, `Analysis` will be replaced with a `None`. For example, `BINANCE:DEXEUSDT` does not have an analysis, but `BINANCE:BTCUSDT` has:

```
# Example
{'BINANCE:DEXEUSDT': None, 'BINANCE:BTCUSDT': <tradingview_ta.main.Analysis object at␣
↪0x7f3561cdeb20>}
```

2.6 Symbol search

New in version 3.3.0.

Search for symbols using the TradingView symbol search API. Returns a list of symbols, exchanges, types, descriptions, and logo URLs matching the search query.

```
from tradingview_ta import TradingView
print(TradingView.search("tesla", "america"))
# Output: [{'symbol': 'TSLA', 'exchange': 'NASDAQ', 'type': 'stock', 'description':
↪'Tesla, Inc.', 'logo': 'https://s3-symbol-logo.tradingview.com/tesla.svg'}, ...]
```

Note: While symbols listed on https://tvdb.brianthe.dev are guaranteed to work with the "get analysis()" function, symbols returned by this function may not.

Parameters:

- text (`str`) – Query string.
- type (`str`, optional) – Type of asset (stock, crypto, futures, index). Defaults to None (all).

2.7 Proxy

Simply add the `proxies` parameter if you wish to utilize a proxy. Works with both `TA_Handler()` and `get_multiple_analysis()`. It's worth noting that a bad proxy could result in TradingView rejecting your request.

```
from tradingview_ta import TA_Handler, Interval, Exchange
tesla = TA_Handler(
    symbol="TSLA",
    screener="america",
    exchange="NASDAQ",
    interval=Interval.INTERVAL_1_DAY,
    proxies={'http': 'http://0.0.0.0:8080', 'https': 'https://0.0.0.0:443'}
)
print(tesla.get_analysis().summary)
# Example output: {"RECOMMENDATION": "BUY", "BUY": 8, "NEUTRAL": 6, "SELL":␣
↪3}
```

CHAPTER 3

Changelog

3.1 3.3.0

New:

- Symbol search

3.2 3.2.10

New:

- Support for proxy
- Additional indicators support for `get_multiple_analysis()`

3.3 3.2.9

New:

- Interval: 30 minutes and 2 hours

3.4 3.2.8

New:

- Indicators: change, low, high

3.5 3.2.7

Bug fix:

- `get_multiple_analysis()` will now return `None` if there is no analysis for a certain symbol. See #55 for more details.

3.6 3.2.6

Bug fix:

- Add indicators
- Get multiple analysis

3.7 3.2.5

New:

- Retrieve indicators `TA_Handler.get_indicators()`
- Add custom indicators `TA_Handler.add_indicators()`
- Indicators: volume

Bug fix:

- Update RSI algorithms

3.8 3.2.4

New:

- Retrieve multiple analysis

Bug fix:

- Update compute algorithms

3.9 3.2.3

New:

- Timeout
- Indicators: BBUpper and BBLower
- Test script (test.py)

3.10 3.2.2

New:

- Indicators: open and P.SAR

3.11 3.2.1

New:

- Removed EMA5 and SMA5 from analysis

Bug fix:

- Switched buy/sell on momentum

3.12 3.2.0

New:

- Instantiate `TA_Handler` using parameters

3.13 3.1.6

New:

- Set interval to `"1d"` if invalid

3.14 3.1.5

New:

- Move indicators to `Analysis` class

3.15 3.1.4

Bug fix:

- Pull request #19

3.16 3.1.3

New:

- Added user agent
- Added `__version__` attribute

3.17 3.1.1

Bug fix:

- Pull request #7

3.18 3.1.0

New:

- Set symbol/exchange/screener/interval using functions

3.19 3.0.0

New:

- Use scanner (https://scanner.tradingview.com/america/scan) instead of selenium
- Indicators

3.20 2.5.0

New:

- Support for Heroku

3.21 2.2.0

New:

- Rename `pair` to `symbol`
- Support for Python 3.4
- Added warnings

3.22 2.1.0

Bug fix:

- Requirements

3.23 2.0.0

New:

- Use class
- Use headless selenium webdriver

CHAPTER 4

FAQ

4.1 Is the data delayed?

Yes and no. Quoted from TradingView:

> We provide real-time data for free whenever we're allowed. However, some data is delayed due to specific exchange regulations. Because of this, real-time data must be purchased separately using the page below. US stock market data is real-time and provided by CBOE BZX.

Note: TradingView_TA does not support paid real-time data at the moment.

Note: Please refer to TradingView's website to see whether the data is delayed or not.

4.2 How do I get past data?

Retrieving past data is currently not supported.

4.3 How do I create a trading bot?

Warning: Trading (especially using bots) is very risky. I won't be responsible for any financial loss. You have been warned.

The pseudocode below should help you get started in creating your own trading bot.

```
# Import packages.
from tradingview_ta import TA_Handler, Interval, Exchange
import time

# Store the last order.
last_order = "sell"

# Instantiate TA_Handler.
handler = TA_Handler(
    symbol="SYMBOL",
    exchange="EXCHANGE",
    screener="SCREENER",
    interval="INTERVAL",
)

# Repeat forever.
while True:
    # Retrieve recommendation.
    rec = handler.get_analysis()["RECOMMENDATION"]

    # Create a buy order if the recommendation is "BUY" or "STRONG_BUY" and the last␣
↪order is "sell".
    # Create a sell order if the recommendation is "SELL" or "STRONG_SELL" and the␣
↪last order is "buy".
    if "BUY" in rec and last_order == "sell":
        # REPLACE COMMENT: Create a buy order using your exchange's API.

        last_order = "buy"
    elif "SELL" in rec and last_order == "buy":
        # REPLACE COMMENT: Create a sell order using your exchange's API.

        last_order = "sell"

    # Wait for x seconds before retrieving new analysis.
    # The time should be the same as the interval.
    time.sleep(x)
```

Warning: `last_order` won't be saved when the program exit. When the bot restarts, it will always create a new buy order.

Tip: Always paper trade before risking your money.

4.4 How can I get involved?

If you found a bug, please create an issue on the GitHub repository.

You can contribute (new features, bug fix, typo, etc) through the GitHub repository. Please follow the guidelines and don't send spammy pull requests.

4.5 How does TradingView_TA works?

A simple network inspection on TradingView's website revealed that the data is retrieved through an undocumented API.

TradingView_TA works by calculating similar data using algorithms reverse-engineered from their JavaScript code.

4.6 Why do I get 4XX error?

400 error indicates that the request is invalid. Usually, this is caused when the indicators does not exist. See https://pastebin.com/1DjWv2Hd for valid indicators.

404 error indicates that the webpage does not exist. Usually, this is caused when the screener does not exist. Check if the screener, symbol, and exchange are correct using this tool: https://tvdb.brianthe.dev.