WTDpy

Release 0.0.3

Apr 18, 2020

Contents:

WTDpy is a python package for making basic calls to the GitHub World Trading Data API.

Welcome to WTDpy documentation! Please check the contents below for information on installation, getting started and example code. If you want to dive straight into the code you can check out the GitHub page or the working examples presented in Jupyter Notebooks.

CHAPTER 1

Installation

1.1 Stable release

Installing WTDpy via pip is the preferred method, as it will always install the most recent stable release. If you do not have pip installed, this Python installation guide can guide you through the process.

To install WTDpy, run this command in your terminal:

```
# Use pip to install wtdpy
pip install wtdpy
```

1.2 From sources

The sources for WTDpy can be downloaded from the Github repo.

You can either clone the public repository:

```
# Use git to clone WTDpy
git clone https://github.com/uijl/wtdpy/
```

Or download the tarball:

```
# Use curl to obtain the tarball
curl  -OL https://github.com/uijl/wtdpy/tarball/master
```

Once you have a copy of the source, you can install it with:

```
# Use python to install
python setup.py install
```

CHAPTER 2

Examples

This small example guide will cover the basic start-up and the functions now available in WTDpy:

- Import package and set-up a class.
- Make a call to the World Trading Data API.

2.1 Start-Up

The first part is importing the package and setting up the class with your own API key.

```
# Import wtdpy
from wtdpy import WTDpy
```

If you do not yet have your own API key of the World Trading Data API you can create a free account. With this free account you can request up to 250 historical time series per day. As soon as you log on to your dashboard it shows large on the screen "Your API Token". Copy this random string and assign it to a variable.

```
# Set your own api_key
api_key = "your_unique_api_key"

# Create the class
wtd = WTDpy(api_token = api_key)
```

2.2 Make a call to the API

Once you have initialised your WTDpy class you can make a call to the World Trading data API. If you are not sure about the symbol you want to look up you can check if the symbol you want is returned by the API with the following code.

```
# Check for correct response
# Returns True if the returned symbol is equal to the request
symbol = ["^AEX"]
wtd.search_available_data(symbol = symbol)
```

If you have selected the correct symbol, or list of symbols, you can use the code below to request all the available historical data.

```
# Request historical time series of two indices
symbol = ["^AEX", "^DAX"]
wtd.get_historical_data(symbol = symbol)
```

CHAPTER 3

WTDpy

This page lists all functions and classes available of WTDpy.

3.1 Submodules

There is only one submodule with one class: WTDpy. Please check the information below on how the code reads.

3.2 wtdpy.wtdpy module

WTDpy Packge.

class wtdpy.wtdpy.**WTDpy**(*api_token: str*)

Python wrapper for the World Trading Data API.

Parameters

api_token [str] Your personal API token from https://www.worldtradingdata.com/.

Attributes

api_token [str] Your personal API token from https://www.worldtradingdata.com/.

url [str] The base url for requesting the API.

Notes

Create an account on https://www.worldtradingdata.com/ to obtain your API key. It will show on the top of your personal dashboard. You can make 250 API calls per day with the free tier.

Currently supporting:

- Historical data;
- Checking if a symbol excists.

get_historical_data(*symbol: Union[str, List[str]], date_from: Union[datetime.datetime, str, None] = None, date_to: Union[datetime.datetime, str, None] = None, sort: str = 'asc', output: str = 'dict'*) → Union[dict, pandas.core.frame.DataFrame]

Get the historical data for the given symbol.

Parameters

symbol [str or list of str] A string or list of strings such as "^AEX", or ["^AEX", "^DAX"]. The symbols should match the format as shown on the World Trading Data site. You can check https://www.worldtradingdata.com/search to see what data is available at World Trading Data.

date_from [str or datetime] A string representing a date in isoformat (yyyy-mm-dd), a datetime object or None. If no data is provided the earliest date will be requested.

date_to: str or datetime A string representing a date in isoformat (yyyy-mm-dd), a datetime object or None. If no data is provided the latest date will be requested.

sort: str A string representing the sorting of the requested data. Either ascending "asc" or descending "desc".

output: str A string representing the desired output. Either *dict* for a python dict or "df" for a pandas DataFrame.

Raises

ValueError

- If provided symbol is not available.

TypeError

- If *date_from* is not a correct string or datetime object
- If *date_to* is not a correct string or datetime object
- If *sort* is not a correct string
- If *output* is not a correct string

Returns

to_return [dict or pd.DataFrame] A dictionary or a pandas DataFrame. If *symbol* is a list then a dictionary of pandas DataFrames will be returned.

search(*query: Union[str, List[str]], number_of_hits: int = 5*) → Dict[str, List[dict]]

Search the query in the World Trading Data database.

Parameters

query [str or list of str] A string or list of strings such as "Apple computers" or ["Apple computers", "Microsoft"]. This is similar to manually checking https://www.worldtradingdata.com/search to see what data is available at World Trading Data.

number_of_hits [int] The number of hits that will be returned.

Returns

returned_hits [dict] A dict, with the listed hits based on your search query.

search_available_data(*symbol: Union[str, List[str]], list_alternatives: bool = False, number_of_alternatives: int = 1*) → Union[bool, List[dict]]

Check if the requested data is available.

Parameters

symbol [str or list of str] A string or list of strings such as "^AEX", or ["^AEX", "^DAX"]. The symbols should match the format as shown on the World Trading Data site. You can check https://www.worldtradingdata.com/search to see what data is available at World Trading Data.

list_alternatives [bool] If *True* a list of likely matches is presented

number_of_alternatives [int] The number of alternatives, besides the top hit, that are checked whether or not they match the provided symbol.

Returns

matching_symbols or alternatives [bool or list] A boolean to check whether data is avaible. If *list_alternatives* is *True* and a symbol is not found a list with possible alternatives will be returned.

CHAPTER 4

History

4.1 0.3.0 (2020-03-28)

- Structured package
- Type hints for all functions

4.2 v0.2.0 (2020-03-23)

- Added search

4.3 v0.1.0 (2020-03-21)

- First release to PyPI

CHAPTER 5

Indices and tables

- genindex
- modindex
- search

Python Module Index

W

Index

G

S

W