PyPtax

Release 0.1.0

Bruno Cardoso

Feb 03, 2020

CONTENTS:

PyPtax is a Python library to retrieve information on Ptax Rates.

CHAPTER

ONE

INSTALLATION

Install the latest released version from PyPI:

```
$ pip install pyptax
```

Install the development version from GitHub:

```
$ pip install git+https://github.com/brunobcardoso/pyptax
```

Installing from source:

PyPtax is actively developed on GitHub, where the code is available under the MIT license.

```
$ git clone https://github.com/brunobcardoso/pyptax.git
$ cd pyptax
$ pip install -e ".[dev]"
```

CHAPTER

TWO

QUICKSTART

2.1 Closing Ptax rates for a requested date

As a simple example, we'll request closing Ptax rates on January 20, 2020. We have to inform the date in the format YYYY-MM-DD.

```
>>> from pyptax import ptax
>>> bulletin = ptax.close('2020-01-20')
>>> bulletin
Bulletin(
    datetime="2020-01-20 13:09:02.871",
    bid=4.1823,
    ask=4.1829,
    bulletin_type="close"
)
```

The ptax.close returns an instance of Bulletin:

```
>>> bulletin.datetime
'2020-01-20 13:09:02.871'
>>> bulletin.bid
4.1823
>>> bulletin.ask
4.1829
```

It's also possible to process the information as a dictionary:

```
>>> bulletin.as_dict
{
    "datetime": "2020-01-20 13:09:02.871",
    "bid": 4.1823,
    "ask": 4.1829,
    "bulletin_type": "close"
}
```

Or as a fixed width table for pretty printing:

```
>>> print(bulletin.display())
+---------------+-------------------------+
| datetime      | 2020-01-20 13:09:02.871 |
| bid           | 4.1823                  |
| ask           | 4.1829                  |
| bulletin_type | close                   |
+---------------+-------------------------+
```

2.2 Historical Ptax rates for a requested period

```
>>> from pyptax import ptax
>>> historical_bulletin = ptax.historical('2020-01-01', '2020-01-05')
>>> historical_bulletin
HistoricalBulletin(
    start_date="2020-01-01",
    end_date="2020-01-03",
    bulletins=[
        Bulletin("2020-01-02 13:11:10.762", 4.0207, 4.0213, "close"),
        Bulletin("2020-01-03 13:06:22.606", 4.0516, 4.0522, "close"),
    ],
)
```

The ptax.historical returns an instance of HistoricalBulletin:

```
>>> historical_bulletin.start_date
"2020-01-01"
>>> historical_bulletin.end_date
"2020-01-03"
>>> historical_bulletin.bulletins
[
    Bulletin("2020-01-02 13:11:10.762", 4.0207, 4.0213, "close"),
    Bulletin("2020-01-03 13:06:22.606", 4.0516, 4.0522, "close"),
]
```

As a dictionary:

```
>>> historical_bulletin.as_dict
{
    "start_date": "2020-01-01",
    "end_date": "2020-01-03",
    "bulletins": [
        {
            "datetime": "2020-01-02 13:11:10.762",
            "bid": 4.0207,
            "ask": 4.0213,
            "bulletin_type": "close"
        },
        {
            "datetime": "2020-01-03 13:06:22.606",
            "bid": 4.0516,
            "ask": 4.0522,
            "bulletin_type": "close"
        },
    ],
}
```

Or a as a fixed width table:

```
>>> print(historical_bulletin.display())
+-------------------------+--------+--------+-----------------+
| datetime                |    bid |    ask | bulletin_type   |
|-------------------------+--------+--------+-----------------|
| 2020-01-02 13:11:10.762 | 4.0207 | 4.0213 | close           |
| 2020-01-03 13:06:22.606 | 4.0516 | 4.0522 | close           |
+-------------------------+--------+--------+-----------------+
```

2.3 Intermediary Ptax rates for a requested date

```
>>> from pyptax import ptax
>>> intermediary = ptax.intermediary('2020-01-02')
>>> intermediary
IntermediaryBulletin(
    date='2020-01-02',
    bulletins=[
        Bulletin(
            datetime='2020-01-02 10:08:18.114',
            bid=4.0101,
            ask=4.0107,
            bulletin_type='open'
        ),
        Bulletin(
            datetime='2020-01-02 11:03:40.704',
            bid=4.0118,
            ask=4.0124,
            bulletin_type='intermediary'
        ),
        Bulletin(
            datetime='2020-01-02 12:10:55.168',
            bid=4.0302,
            ask=4.0308,
            bulletin_type='intermediary'
        ),
        Bulletin(
            datetime='2020-01-02 13:11:10.756',
            bid=4.0305,
            ask=4.0311,
            bulletin_type='intermediary'
        ),
        Bulletin(
            datetime='2020-01-02 13:11:10.762',
            bid=4.0207,
            ask=4.0213,
            bulletin_type='close'
        )
    ]
)
```

The ptax.intermediary returns an instance of IntermediaryBulletin:

```
>>> intermediary.date
"2020-01-02"
>>> intermediary.bulletins
[
    Bulletin("2020-01-02 10:08:18.114", 4.0101, 4.0107, "open"),
    Bulletin("2020-01-02 11:03:40.704", 4.0118, 4.0124, "intermediary"),
    Bulletin("2020-01-02 12:10:55.168", 4.0302, 4.0308, "intermediary"),
    Bulletin("2020-01-02 13:11:10.756", 4.0305, 4.0311, "intermediary"),
    Bulletin("2020-01-02 13:11:10.762", 4.0207, 4.0213, "close"),
]
```

As a dictionary:

```
>>> intermediary.as_dict
{
    'date': '2020-01-02',
    'bulletins': [
        {
            'datetime': '2020-01-02 10:08:18.114',
            'bid': 4.0101,
            'ask': 4.0107,
            'bulletin_type': 'open'
        },
        {
            'datetime': '2020-01-02 11:03:40.704',
            'bid': 4.0118,
            'ask': 4.0124,
            'bulletin_type': 'intermediary'
        },
        {
            'datetime': '2020-01-02 12:10:55.168',
            'bid': 4.0302,
            'ask': 4.0308,
            'bulletin_type': 'intermediary'
        },
        {
            'datetime': '2020-01-02 13:11:10.756',
            'bid': 4.0305,
            'ask': 4.0311,
            'bulletin_type': 'intermediary'
        },
        {
            'datetime': '2020-01-02 13:11:10.762',
            'bid': 4.0207,
            'ask': 4.0213,
            'bulletin_type': 'close'
        }
    ]
}
```

Or a as a fixed width table:

```
>>> print(intermediary.display())
+-------------------------+--------+--------+-----------------+
| datetime                |    bid |    ask | bulletin_type   |
|-------------------------+--------+--------+-----------------|
| 2020-01-02 10:08:18.114 | 4.0101 | 4.0107 | open            |
| 2020-01-02 11:03:40.704 | 4.0118 | 4.0124 | intermediary    |
| 2020-01-02 12:10:55.168 | 4.0302 | 4.0308 | intermediary    |
| 2020-01-02 13:11:10.756 | 4.0305 | 4.0311 | intermediary    |
| 2020-01-02 13:11:10.762 | 4.0207 | 4.0213 | close           |
+-------------------------+--------+--------+-----------------+
```

CHAPTER

THREE

COMMAND LINE INTERFACE

After installing the package, the `pyptax` command becomes available:

```
$ pyptax

 /$$$$$$$            /$$$$$$$    /$$
| $$__  $$          | $$__  $$  | $$
| $$  \ $$ /$$   /$$| $$  \ $$ /$$$$$$    /$$$$$$  /$$   /$$
| $$$$$$$/| $$  | $$| $$$$$$$/|_  $$_/   |____  $$|  $$ /$$/
| $$____/ | $$  | $$| $$____/   | $$      /$$$$$$$ \  $$$$/
| $$      | $$  | $$| $$        | $$ /$$ /$$__  $$  >$$  $$
| $$      |  $$$$$$$| $$        |  $$$$/|  $$$$$$$ /$$/\  $$
|__/       \____  $$|__/         \___/   \_______/|__/  \__/
           /$$  | $$
          |  $$$$$$/
           \_____/

Usage: pyptax [OPTIONS] COMMAND [ARGS]...

  PyPtax Command Line Interface

  PyPtax is a Python library to retrieve information on Ptax rates.

Options:
  -v, --version
  --help         Show this message and exit.

Commands:
  close  Provides bid and ask rates for the requested date.
  historical  Provides bid and ask rates for the requested time period.
```

`pyptax.cli.`**`close`**(**args*, ***kwargs*)

Provide bid and ask rates for the requested date.

Options:

-d, --date YYYY-MM-DD [required]

--help Show this message and exit.

Examples:

```
$ pyptax close --date 2020-01-02
+----------+-------------------------+
| datetime | 2020-01-02 13:11:10.762 |
| bid      | 4.0207                  |
```

(continues on next page)

(continued from previous page)

```
| ask      | 4.0213                  |
+----------+-------------------------+
```

`pyptax.cli.`**`historical`**(**args*, ***kwargs*)

Provide bid and ask rates for the requested time period.

Options:

-sd, --start_date YYYY-MM-DD [required]

-ed, --end_date YYYY-MM-DD [required]

--help Show this message and exit.

Examples:

```
$ pyptax historical --start_date 2020-01-02 --end_date 2020-01-10
+-------------------------+--------+--------+-----------------+
| datetime                |    bid |    ask | bulletin_type   |
|-------------------------+--------+--------+-----------------|
| 2020-01-02 13:11:10.762 | 4.0207 | 4.0213 | close           |
| 2020-01-03 13:06:22.606 | 4.0516 | 4.0522 | close           |
| 2020-01-06 13:03:22.271 | 4.0548 | 4.0554 | close           |
| 2020-01-07 13:06:14.601 | 4.0835 | 4.0841 | close           |
| 2020-01-08 13:03:56.075 | 4.0666 | 4.0672 | close           |
| 2020-01-09 13:03:52.663 | 4.0738 | 4.0744 | close           |
| 2020-01-10 13:10:19.927 | 4.0739 | 4.0745 | close           |
+-------------------------+--------+--------+-----------------+
```

`pyptax.cli.`**`intermediary`**(**args*, ***kwargs*)

Provide intermediary bulletins of ptax rates for the requested date.

Options:

-d, --date YYYY-MM-DD [required]

--help Show this message and exit.

Examples:

```
$ pyptax intermediary --date 2020-01-02
+-------------------------+--------+--------+-----------------+
| datetime                |    bid |    ask | bulletin_type   |
|-------------------------+--------+--------+-----------------|
| 2020-01-02 10:08:18.114 | 4.0101 | 4.0107 | open            |
| 2020-01-02 11:03:40.704 | 4.0118 | 4.0124 | intermediary    |
| 2020-01-02 12:10:55.168 | 4.0302 | 4.0308 | intermediary    |
| 2020-01-02 13:11:10.756 | 4.0305 | 4.0311 | intermediary    |
| 2020-01-02 13:11:10.762 | 4.0207 | 4.0213 | close           |
+-------------------------+--------+--------+-----------------+
```

CHAPTER

FOUR

PYPTAX PACKAGE: API DOCUMENTATION

This part of the documentation covers the interfaces of PyPtax.

`pyptax.ptax.close`(*date: str*) → pyptax.models.Bulletin

Retrieve closing Ptax rates on a certain date.

Parameters **date** – Year, month and day of the date to be searched. Format - "YYYY-MM-DD"

Returns A Bulletin object with datetime, bid and ask attributes

Return type Bulletin

Raises **DateFormatError** – If fails to parse the informed date

Examples

```
>>> bulletin = close("2020-01-20")
>>> bulletin
Bulletin(
    datetime="2020-01-20 13:09:02.871",
    bid=4.1823,
    sk=4.1829,
    bulletin_type="close"
)
>>> bulletin.bid
4.1823
>>> bulletin.ask
4.1829
>>> bulletin.as_dict
{
    "datetime": "2020-01-20 13:09:02.871",
    "bid": 4.1823,
    "ask": 4.1829,
    "bulletin_type": "close"
}
```

`pyptax.ptax.historical`(*start_date: str*, *end_date: str*) → pyptax.models.HistoricalBulletin

Retrieve historical closing Ptax rates for the requested time period.

Parameters

- **start_date** – Beginning of the time period to be searched. Format - 'YYYY-MM-DD'
- **end_date** – End of the time period to be searched. Format - 'YYYY-MM-DD'

Returns A HistoricalBulletin object with start_date, end_date and bulletins with a list of Bulletins.

Return type HistoricalBulletin

Raises

- **DateFormatError** – If fails to parse the informed date
- **ClientError** – If Ptax Service response returns an error
- **UnavailableDataError** – If receives an empty list from Ptax Service

Examples

```
>>> historical_bulletin = historical("2020-01-02", "2020-01-03")
>>> historical_bulletin
HistoricalBulletin(
    start_date="2020-01-02",
    end_date="2020-01-04",
    bulletins=[
        Bulletin(
            datetime="2020-01-02 13:11:10.762",
            bid=4.0207,
            ask=4.0213,
            bulletin_type="close"
        ),
        Bulletin(
            datetime="2020-01-03 13:06:22.606",
            bid=4.0516,
            ask=4.0522,
            bulletin_type="close"
        ),
    ],
)
>>> historical_bulletin.as_dict
{
    "start_date": "2020-01-02",
    "end_date": "2020-01-04",
    "bulletins": [
        {
            "datetime": "2020-01-02 13:11:10.762",
            "bid": 4.0207,
            "ask": 4.0213,
            "bulletin_type": "close"
        },
        {
            "datetime": "2020-01-03 13:06:22.606",
            "bid": 4.0516,
            "ask": 4.0522,
            "bulletin_type": "close"
        },
    ],
}
```

`pyptax.ptax.`**`intermediary`**(*date: str*)

Retrieve intermediary bulletins of Ptax rates for the requested date.

Parameters **date** – Date to be searched. Format - 'YYYY-MM-DD'

Returns A IntermediaryBulletin object with date and bulletins with a list of Bulletins.

Return type IntermediaryBulletin

Raises

- **DateFormatError** – If fails to parse the informed date
- **ClientError** – If Ptax Service response returns an error
- **UnavailableDataError** – If receives an empty list from Ptax Service

Examples

```
>>> intermediary("2020-01-02")
IntermediaryBulletin(
    date='2020-01-02',
    bulletins=[
        Bulletin(
            datetime='2020-01-02 10:08:18.114',
            bid=4.0101,
            ask=4.0107,
            bulletin_type='open'
        ),
        Bulletin(
            datetime='2020-01-02 11:03:40.704',
            bid=4.0118,
            ask=4.0124,
            bulletin_type='intermediary'
        ),
        Bulletin(
            datetime='2020-01-02 12:10:55.168',
            bid=4.0302,
            ask=4.0308,
            bulletin_type='intermediary'
        ),
        Bulletin(
            datetime='2020-01-02 13:11:10.756',
            bid=4.0305,
            ask=4.0311,
            bulletin_type='intermediary'
        ),
        Bulletin(
            datetime='2020-01-02 13:11:10.762',
            bid=4.0207,
            ask=4.0213,
            bulletin_type='close'
        )
    ]
)
```

CHAPTER

FIVE

CONTRIBUTING

Contributions are always welcome and appreciated. You can contribute with pull requests, bug reports, features requests, and feedback.

5.1 Development installation

- Fork PyPtax to your GitHub account
- Clone your fork and create a branch for the code you want to add
- Create a new virtualenv and install the package in development mode

```
$ git clone git@github.com:your_user_name/pyptax.git
$ cd pyptax
$ git checkout -b your_contribution_branch
$ python -m venv .venv
$ source .venv/bin/activate
(.venv) $ python -m pip install -U pip setuptools
(.venv) $ pip install -U -e .[dev]
```

5.2 Testing

PyPtax uses tox for testing and development. With tox installed, just execute:

```
$ tox
```

It'll run the tests in all supported Python versions and pre-commit checks.

You can run against a specific available version, by executing:

```
$ tox -e py38
```

CHAPTER

SIX

RELEASE NOTES

6.1 Version 0.5.0

- Add resource to provide historical bid and ask rates for a requested time period.
- Add resource to provide intermediary rates for a requested date.
- Rename *CloseReport* to *Bulletin* and define its type on the internal attribute *bulletin_type*.
- Remove string conversion on bid and ask rates retrieved.
- Fix development installation docs.

6.2 Version 0.4.0

- Add Command line interface.
- Improve exceptions handling.
  - Add `ClientException` to deal with response error.
  - Add `UnavailableDataError` to handle empty data returned.
- Add `Close.display` method to show data as a fixed width table for pretty printing.

6.3 Version 0.3.0

- `ptax.close` requires date in the format YYYY-MM-DD
- Requires Python 3.7 or later
- Add documentation to readthedocs https://pyptax.readthedocs.io/.

6.4 Version 0.2.0

- `ptax.close` returns an instance of the `CloseReport` model instead of a dict.

Before:

```
>>> from pyptax import ptax
>>> ptax.close('01-20-2020')
>>> {'datetime': '2020-01-20 13:09:02.871', 'bid': '4.1823', 'ask': '4.1829'}
```

After:

```
>>> from pyptax import ptax
>>> ptax.close('01-20-2020')
>>> CloseReport(datetime='2020-01-20 13:09:02.871', bid='4.1823', ask='4.1829')
>>> ptax.close('01-20-2020').as_dict
>>> {'datetime': '2020-01-20 13:09:02.871', 'bid': '4.1823', 'ask': '4.1829'}
```

- Create a client to deal with service request to add support for new features.
- Add PyPI publish commands.
- Fix typo in documentation

6.5 Version 0.1.0

- First public release.

CHAPTER

SEVEN

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

p

INDEX

C

H

I

P