pykitopen Documentation

Release 0.1.0

Jonas Teufel

Feb 23, 2021

Contents:

CHAPTER 1

pykitopen

A python wrapper for the *KITOpen* database!

- Free software: MIT license
- Documentation: https://pykitopen.readthedocs.io.

1.1 Getting Started

1.1.1 Installation

The package is best installed using pip, as it will also install all the necessary dependencies

```
$ pip install pykitopen
```

1.2 Usage

To query the KITOpen search function, simply create a `KitOpen` wrapper object with the desired configuration and call the `search` function with the relevant parameters. The returned `SearchResults` object can be iterated for all the publications.

```
from pykitopen import KitOpen, Publication
from pykitopen.config import DEFAULT

kitopen = KitOpen(DEFAULT)
results = kitopen.search({
    'author':       'MUSTERMANN, MAX',
```

(continues on next page)

(continued from previous page)

```
    'start':        '2012',
    'stop':         '2016',
    'view':         Publication.VIEWS.FULL
})

for publication in results:
    print(publication.data)
```

1.3 Features

The library is still under development, which is why this first version only provides some basic functionality. At the moment only a publication search is supported:

- Searching by author and by year
- Customizable publication "views", which define the fields to be included.

1.4 Planned

- Support more search parameters such as publication type, open access availability etc.
- Add support for the metrics generation feature of KITOpen.
- Add additional batching strategies
- Add export of the result to different formats such as CSV, JSON...

1.5 License

Distributed under the MIT License. See `LICENSE` for more information

1.6 Contact

Jonas Teufel - jonseb1998@gmail.com

1.7 Credits

This package was created with Cookiecutter and the audreyr/cookiecutter-pypackage project template.

CHAPTER 2

Installation

2.1 Stable release

To install pykitopen, run this command in your terminal:

```
$ pip install pykitopen
```

This is the preferred method to install pykitopen, as it will always install the most recent stable release.

If you don't have pip installed, this Python installation guide can guide you through the process.

2.2 From sources

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

You can either clone the public repository:

```
$ git clone git://github.com/the16thpythonist/pykitopen
```

Or download the tarball:

```
$ curl -OJL https://github.com/the16thpythonist/pykitopen/tarball/master
```

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

```
$ python setup.py install
```

CHAPTER 3

Usage

3.1 Basic Usage

The most simple use case is to perform a simple search. To do this simply create an instance of a `KitOpen` wrapper object with the desired configuration and then call the `search` method on it with the proper parameters.

A simple search can be constructed by passing a string author argument and the start/end years for the search also as strings.

The resulting `SearchResult` object can be iterated to get all the publication objects.

```
from pykitopen import KitOpen, Publication
from pykitopen.config import DEFAULT

kitopen = KitOpen(DEFAULT)
results = kitopen.search({
    'author':           'MUSTERMANN, M*',
    'start':            '2012',
    'end'               '2016',
    'view'              Publication.VIEWS.FULL
})

for publication in results:
    print(publication.data)
```

3.2 Publication Views

As you might have noticed, there is an additional parameter *'view'*, which can be passed to the search parameters. This parameter is supposed to be an object of the type `PublicationView`. This parameter influences, what kind of data fields are requested for each publication in the search.

Some standard options are available as constant members of the `Publication.VIEWS` class. This included for example the `FULL` view, which will request *all* of the fields and the `BASIC` view which will only contain the most

basic information such as ID, author, title etc. Choosing the appropriate view might help to reduce response times.

3.2.1 Custom Views

The user is not limited to the predefined views though, it is also possible to define custom views with only the required fields. First of all, a list of all the available fields can be displayed like this:

```
from pykitopen.publication import PublicationView

print(PublicationView.FIELDS)
```

A custom view can be created, by simply creating a new instance of the `PublicationView` class. A string name and a subset of the fields list have to be passed to the constructor. This object can then be used to be passed as a search parameter or even set as a default in the configuration dict.

```
from pykitopen import KitOpen
from pykitopen.config import DEFAULT
from pykitopen.publication import PublicationView

# Set it as a default
custom_view = PublicationView('MyCustomView', ['author', 'title'])

config = DEFAULT.copy()
config['default_view'] = custom_view

kitopen = KitOpen(config)

# Or use it for a search request directly
kitopen.search({
    'author':       'MUSTERMANN, M*,
    'view':         custom_view
})
```

3.3 Request Batching

3.3.1 The problem

So the problem is, that the used KITOpen interface at KITOpen Auswertungen does not expose a REST API. The only way to export the more detailed information data is through the download of a ZIP file, which then in turn contains a CSV file.

So the way *pykitopen* works in the background is: It downloads the zip file, unpacks it into a temporary folder and parses the csv for the actual data.

This creates a practical complication: If the amount of requested data is high, the server takes a long time to create corresponding csv and zip files, which then leads to a timeout for the request...

3.3.2 Batching Strategies

To work around this problem, it is possible to get the desired data in batches, instead of everything at once. A single request will be split into multiple different requests based on some criteria. This behaviour can be controlled with the `"batching_strategy"` key the configuration dict, which is being passed to the `KitOpen` wrapper object. The default behaviour being the `NoBatching` strategy, which will request all the data at once.

A good alternative would be the `YearBatching` strategy, which will request the data for every year individually.

```
from pykitopen import KitOpen
from pykitopen.search import YearBatching
from pykitopen.config import DEFAULT

# It is good practice to base a custom configuration on a copy of the default
config = DEFAULT.copy()
config['batching_strategy'] = YearBatching

pykitopen = KitOpen(config)
```

Changing the batching strategy does not change anything on the behaviour of `SearchResult`, since the batching is implemented in the background. Each batch is executed, once the iterator reaches the corresponding point.

CHAPTER 4

Credits

4.1 Development Lead

- Jonas Teufel <jonseb1998@gmail.com>

4.2 Contributors

None yet. Why not be the first?

CHAPTER 5

History

5.1 0.1.0 (17.06.2020)

- Initial release

5.2 0.1.1 (23.01.2021)

- changed datetime format in HISTORY.rst
- fixed error
- Added VERSION file

CHAPTER 6

Indices and tables

- genindex
- modindex
- search