Pybraries

Jeff Hale

Oct 06, 2021

CONTENTS:

CHAPTER

ONE

PYBRARIES

Pybraries is a Python wrapper for the libraries.io API. g You can use it to subscribe to email alerts for new versions of open source packages.

You can also use it to find information about many aspects of open source packages and repositories.

The full documentation is hosted at Read the Docs.

1.1 Quick Start

1.1.1 Install

Install from PyPI.:

```
pip install pybraries
```

1.1.2 Use

Get your API key from libraries.io.

Set your API key as to the `LIBRARIES_API_KEY` environment variable from the command line with

```
export LIBRARIES_API_KEY="your_libraries.io_api_key_goes_here"
```

Import the pybraries package and use it to subscribe to a package.

```
from pybraries.subscribe import Subscribe

s = Subscribe()

s.subscribe("pypi", "pandas")
```

Now you'll get an email update every time a new version of *pandas* is released.

Here's another example. Search for projects with *visualization* as a keyword and *python* as a language. Sort by the number of stars.

```
from pybraries.search import Search

search = Search()

info = search.project_search(keywords='visualization', sort='stars', platform='pypi')
print(info)
```

A list of dictionaries with project names and other project information is returned.

Note that the Libraries.io API is rate limited to 60 requests per minute.

All libraries.io methods are implemented, except updating a subscription to not include prereleases. This option can be toggled at the libraries.io website.

Search() and Subscribe() are the two classes in this package. See all their available methods by clicking on the method names in the left sidebar in the documentation.

1.1.3 Key Terms

host A repository host platform. e.g. GitHub

owner A repository owner. e.g. pandas-dev

repo A repository. e.g. pandas

user A repository user e.g. a GitHub username. e.g. discdiver

platform A package manager platform. e.g. PyPI

project A package or library distributed by a package manager platform. e.g. pandas

Note that many repos and projects share the same name. Many owners and repos also share the same name. Further, many owners are also users. Fun!

Pybraries methods that return one item generally return a dict with information.

Methods that return multiple items return a list of dicts.

1.1.4 Docs

- Check out the full pybraries documentation.

1.1.5 Getting Help

1. Check out the pybraries docs. 1. Check out the libraries.io docs. 1. Open an issue on GitHub or tag a question on Stack Overflow with "pybraries".

1.1.6 Contributing

- Contributions are welcome and appreciated! See Contributing.

1.1.7 License

- BSD-3-clause

CHAPTER

TWO

CONTRIBUTING

Contributions are welcome and greatly appreciated!

You can contribute in many ways:

We're a welcoming project! Please ensure you follow our Code of Conduct.

2.1 Types of Contributions

2.1.1 Report Security Vulnerabilities

If you think you have found a security vulnerability, please email jeffmshale at gmail dot com.

Please don't report it in an GitHub issue or in any other public forum.

Thank you!

2.1.2 Report Bugs and Make Feature Requests

Open an issue in our GitHub Repo.

2.1.3 Write Code

Feel free to look through the GitHub issues for open issues. Anything tagged with "help wanted" is available to fix.

Please ensure new and altered features have tests and are documented with DocStrings.

2.1.4 Write Documentation

We want pybraries users to have a great experience. Documentation is a huge part of Developer Experience.

Feel free to add to and improve the documentation. You can contribute to official pybraries docs, in docstrings, or by writing blog posts.

Small changes to the docs can be made by editing the code on GitHub in the browser and opening a PR.

If making more substantial changes or additions:

When in the docs folder, build the docs with the command:

```
make html
```

The built HTML docs will be created in the docs->_build folder.

Check for broken links by running the following command:

```
make linkcheck
```

2.2 Get Started!

Ready to contribute? Here's how to set up *pybraries* for local development.

1. Fork the *pybraries* repo on GitHub.
2. Clone your fork locally:

   ```
   git clone git@github.com:your_github_username_here/pybraries.git
   ```

3. Install your local copy into a virtual environment.
4. Create a branch for local development:

   ```
   git checkout -b name-of-your-bugfix-or-feature-branch
   ```

   Now you can make your changes locally.

4. Install requirements_dev.txt with:

   ```
   pip install -r requirements_dev.txt
   ```

5. We use black and Flake8 for style guide sanity.

Max line length is set to 100 characters and the following errors are ignored:

- F401 - module imported but unused
- F841 - local variable name is assigned to but never used
- W291 - trailing whitespace

6. When you're done making changes, check that your changes pass the test suite and Flake8:

```
pytest flake8
```

7. Commit your changes and push your branch to GitHub:

   ```
   git add .
   git commit -m "Your detailed description of your changes."
   git push origin name-of-your-bugfix-or-feature
   ```

8. Submit a pull request through GitHub.

If you are new to contributing to open source, check out this guide by Chalmer Lowe.

2.3 Pull Request Guidelines

Before you submit a pull request, check that it meets these guidelines:

1. If code was updated, the pull request should include tests.
2. If the pull request adds functionality, the docs should be updated. Put your new functionality into a function with a docstring, and add the feature to the list in README.rst.
3. The code should work for Python 3.8 and higher.

2.4 Deploying

A reminder for the maintainers on how to deploy.

1. Make sure all changes are committed (including an entry in history.rst).
2. Then run:

```
bumpversion2 patch      # possible: major / minor / patch
git push
git push --tags
```

3. Build with:

```
python setup.py sdist bdist_wheel
```

4. Use twine to upload to PyPI.
5. Update the Releases section on GitHub.

CHAPTER

THREE

CONTRIBUTOR COVENANT CODE OF CONDUCT

3.1 Our Pledge

In the interest of fostering an open and welcoming environment, we as contributors and maintainers pledge to making participation in our project and our community a harassment-free experience for everyone, regardless of age, body size, disability, ethnicity, sex characteristics, gender identity and expression, level of experience, education, socio-economic status, nationality, personal appearance, race, religion, or sexual identity and orientation.

3.2 Our Standards

Examples of behavior that contributes to creating a positive environment include:

- Using welcoming and inclusive language
- Being respectful of differing viewpoints and experiences
- Gracefully accepting constructive criticism
- Focusing on what is best for the community
- Showing empathy towards other community members

Examples of unacceptable behavior by participants include:

- The use of sexualized language or imagery and unwelcome sexual attention or advances
- Trolling, insulting/derogatory comments, and personal or political attacks
- Public or private harassment
- Publishing others' private information, such as a physical or electronic address, without explicit permission
- Other conduct which could reasonably be considered inappropriate in a professional setting

3.3 Our Responsibilities

Project maintainers are responsible for clarifying the standards of acceptable behavior and are expected to take appropriate and fair corrective action in response to any instances of unacceptable behavior.

Project maintainers have the right and responsibility to remove, edit, or reject comments, commits, code, wiki edits, issues, and other contributions that are not aligned to this Code of Conduct, or to ban temporarily or permanently any contributor for other behaviors that they deem inappropriate, threatening, offensive, or harmful.

3.4 Scope

This Code of Conduct applies both within project spaces and in public spaces when an individual is representing the project or its community. Examples of representing a project or community include using an official project e-mail address, posting via an official social media account, or acting as an appointed representative at an online or offline event. Representation of a project may be further defined and clarified by project maintainers.

3.5 Enforcement

Instances of abusive, harassing, or otherwise unacceptable behavior may be reported by contacting the project team at jeffmshale@gmail.com. All complaints will be reviewed and investigated and will result in a response that is deemed necessary and appropriate to the circumstances. The project team is obligated to maintain confidentiality with regard to the reporter of an incident. Further details of specific enforcement policies may be posted separately.

Project maintainers who do not follow or enforce the Code of Conduct in good faith may face temporary or permanent repercussions as determined by other members of the project's leadership.

3.6 Attribution

This Code of Conduct is adapted from the Contributor Covenant, version 1.4, available at https://www.contributor-covenant.org/version/1/4/code-of-conduct.html

For answers to common questions about this code of conduct, see https://www.contributor-covenant.org/faq

CHAPTER

FOUR

HISTORY

4.1 0.0.1 (2019-05-9)

- First pre-release

4.2 0.0.5 (2019-05-22)

- Alpha release
- All libraries.io methods implemented except update

4.3 0.1.0 (2019-05-24)

- Better argument docs
- Type hinting
- Type checking

4.4 0.1.1 (2019-05-25)

- Remove subscribe extra print statements

4.5 0.2.0 (2019-07-10)

- Fix typo in two search functions
- Update dependencies

4.6 0.2.2 (2020-04-04)

- Refactored code - (thanks victorgveloso!)
- Fixed naming divergences between libraries.io API and pybraries (manager->platform and package->project)

4.7 0.3.0 (2020-09-19)

- Fixed bug in project_search. Now pass keywords="my keywords to search" as an argument. Must be passed as a keyword argument.

4.8 0.3.1 (2020-09-20)

- project_search. Other project_search functionality with platforms regression fixed.

4.9 0.3.2 (2020-09-26)

- Fix regression bug in kwargs

4.10 0.3.3 (2020-09-26)

- remove stdout from project_search

4.11 0.3.4 (2020-09-26)

- fix filter kwargs

pybraries.search.Search()	Class for wrapping the libraries.io API for platform, project, repo, and user GET actions
pybraries.subscribe.Subscribe()	Class for libraries.io API for changing user's libraries.io subscriptions

CHAPTER

FIVE

PYBRARIES.SEARCH.SEARCH

class pybraries.search.**Search**

Class for wrapping the libraries.io API for platform, project, repo, and user GET actions

__init__()

Initialize self. See help(type(self)) for accurate signature.

Methods

__init__()	Initialize self.
platforms()	Return a list of supported package managers.
project(platforms, name)	Return information about a project and its versions from a platform (e.g.
project_contributors(platforms, project)	Get users that have contributed to a given project.
project_dependencies(platforms, project[, ...])	Get dependencies for a version of a project.
project_dependent_repositories(platforms, ...)	Get repositories that depend on a given project.
project_dependents(platforms, project[, version])	Get projects that have at least one version that depends on a given project.
project_search(**kwargs)	Search for projects. Args - keywords only: keywords (str): required argument: keywords to search languages (str): optional programming languages to filter licenses (str): license type to filter platforms (str):, platforms to filter.
project_sourcerank(platforms, project)	Get breakdown of SourceRank score for a given project.
project_usage(platforms, project)	Get breakdown of usage for a given project.
repository(host, owner, repo)	Return information about a repository and its versions.
repository_dependencies(host, owner, repo)	Return information about a repository's dependencies.
repository_projects(host, owner, repo)	Get a list of projects referencing the given repository.
user(host, user)	Return information about a user.
user_dependencies(host, user)	Return a list of unique user's repositories' dependencies.
user_projects(host, user)	Return information about projects using a user's repos.

continues on next page

Table 1 – continued from previous page

user_projects_contributions(host, user)	Return information about projects a user has contributed to.
user_repositories(host, user)	Return information about a user's repos.
user_repository_contributions(host, user)	Return information about repositories a user has contributed to.

CHAPTER

SIX

PYBRARIES.SUBSCRIBE.SUBSCRIBE

class pybraries.subscribe.**Subscribe**

Class for libraries.io API for changing user's libraries.io subscriptions

__init__()

Initialize self. See help(type(self)) for accurate signature.

Methods

__init__()	Initialize self.
check_subscribed(manager, package)	Check if a user is subscribed to notifications for new project releases.
list_subscribed()	Return a list of packages a user is subscribed to for release notifications.
subscribe(manager, package)	Subscribe to receive notifications about new releases of a project.
unsubscribe(manager, package)	Stop receiving release notifications from a project.
update_subscribe(manager, package[, ...])	NOT IMPLEMENTED due to possible bug in libraries.io Update the options for a subscription.

CHAPTER

SEVEN

INDICES AND TABLES

- genindex
- modindex
- search

INDEX

Symbols

S