hyperliquid-dex /
hyperliquid-python-sdk

Code Issues 6 Pull requests 11 Actions Projects Security Insights

SDK for Hyperliquid API trading with Python.

MIT license
Security policy
646 stars 217 forks 17 watching Branches Activity Custom properties
Tags
Public repository

m... 2 Branches 31 Tags Go to file t Go to file + Add file Code

lmlmt add deployer freeze functionality to client and spot deploy ✓ 719c002 · last week

.github/workflows	Refactor linting + enable via Github workflo...	2 months ago
api	Refactor linting + enable via Github workflo...	2 months ago
examples	add deployer freeze functionality to client a...	last week
hyperliquid	add deployer freeze functionality to client a...	last week
tests	Refactor linting + enable via Github workflo...	2 months ago
.gitignore	Refactor linting + enable via Github workflo...	2 months ago
.pre-commit-config.yaml	Refactor linting + enable via Github workflo...	2 months ago
LICENSE.md	Various small updates	3 months ago
Makefile	Refactor linting + enable via Github workflo...	2 months ago
README.md	Refactor linting + enable via Github workflo...	2 months ago
SECURITY.md	Version 0.1.6	2 years ago
poetry.lock	Add evm block indexer example	2 months ago
pyproject.toml	Version 0.11.0	2 weeks ago

README MIT license Security

hyperliquid-python-sdk

dependencies up to date

code style black security bandit pre-commit enabled semantic-versions license MIT

SDK for Hyperliquid API trading with Python.

Installation

```
pip install hyperliquid-python-sdk
```

Configuration

- Set the public key as the `account_address` in examples/config.json.
- Set your private key as the `secret_key` in examples/config.json.
- See the example of loading the config in examples/example_utils.py

[Optional] Generate a new API key for an API Wallet

Generate and authorize a new API private key on https://app.hyperliquid.xyz/API, and set the API wallet's private key as the `secret_key` in examples/config.json. Note that you must still set the public key of the main wallet *not* the API wallet as the `account_address` in examples/config.json

Usage Examples

```
from hyperliquid.info import Info
from hyperliquid.utils import constants

info = Info(constants.TESTNET_API_URL, skip_ws=True)
user_state = info.user_state("0xcd5051944f780a621ee62e39e493c489668acf4d")
print(user_state)
```

See examples for more complete examples. You can also checkout the repo and run any of the examples after configuring your private key e.g.

```
cp examples/config.json.example examples/config.json
vim examples/config.json
python examples/basic_order.py
```

Getting started with contributing to this repo

1. Download `Poetry` : https://python-poetry.org/.
   - Note that in the install script you might have to set `symlinks=True` in `venv.EnvBuilder` .
   - Note that Poetry v2 is not supported, so you'll need to specify a specific version e.g. curl -sSL https://install.python-poetry.org | POETRY_VERSION=1.4.1 python3 -
2. Point poetry to correct version of python. For development we require python 3.10 exactly. Some dependencies have issues on 3.11, while older versions don't have correct typing support. `brew install python@3.10 && poetry env use /opt/homebrew/Cellar/python@3.10/3.10.16/bin/python3.10`
3. Install dependencies:

```
make install
```

Makefile usage

CLI commands for faster development. See `make help` for more details.

```
check-safety         Run safety checks on dependencies
cleanup              Cleanup project
install              Install dependencies from poetry.lock
install-types        Find and install additional types for mypy
lint                 Alias for the pre-commit target
lockfile-update      Update poetry.lock
lockfile-update-full Fully regenerate poetry.lock
poetry-download      Download and install poetry
pre-commit           Run linters + formatters via pre-commit, run "make pre-commit hook=black" to run only black
test                 Run tests with pytest
update-dev-deps      Update development dependencies to latest versions
```

Releases

You can see the list of available releases on the GitHub Releases page.

We follow the Semantic Versions specification and use `Release Drafter`. As pull requests are merged, a draft release is kept up-to-date listing the changes, ready to publish when you're ready. With the categories option, you can categorize pull requests in release notes using labels.

List of labels and corresponding titles

Label	Title in Releases
`enhancement`, `feature`	Features
`bug`, `refactoring`, `bugfix`, `fix`	Fixes & Refactoring
`build`, `ci`, `testing`	Build System & CI/CD
`breaking`	Breaking Changes
`documentation`	Documentation
`dependencies`	Dependencies updates

Building and releasing

Building a new version of the application contains steps:

- Bump the version of your package with `poetry version <version>`. You can pass the new version explicitly, or a rule such as `major`, `minor`, or `patch`. For more details, refer to the Semantic Versions standard.
- Make a commit to `GitHub`
- Create a `GitHub release`
- `poetry publish --build`

License

This project is licensed under the terms of the `MIT` license. See LICENSE for more details.

```
@misc{hyperliquid-python-sdk,
  author = {Hyperliquid},
  title = {SDK for Hyperliquid API trading with Python.},
  year = {2024},
  publisher = {GitHub},
  journal = {GitHub repository},
  howpublished = {\url{https://github.com/hyperliquid-dex/hyperliquid-python-sdk}}
}
```

Credits

This project was generated with `python-package-template`

Releases 30

0.11.0 Latest
2 weeks ago

+ 29 releases

Packages

No packages published

Used by 133

+ 125

Contributors 33

+ 19 contributors

Languages

● Python 98.3% ● Makefile 1.7%