HexBytes Documentation

Release 1.2.0

The Ethereum Foundation

Apr 19, 2024

GENERAL

HexBytes is a *very* thin wrapper around the python built-in `bytes` class.

It adds these features:

1. Accepts more types for initializing values:
   - `bool`
   - `bytearray`
   - `bytes`
   - `int` (non-negative)
   - `str`
   - `memoryview`
2. The representation at console (`__repr__`) is 0x-prefixed
3. `to_0x_hex` returns a 0x-prefixed hex string

CHAPTER

ONE

INSTALLATION

```
python -m pip install hexbytes
```

1.1 Usage

Example *HexBytes* usage:

```
>>> from hexbytes import HexBytes

# convert from bytes to a prettier representation at the console
>>> HexBytes(b"\x03\x08wf\xbfh\xe7\x86q\xd1\xeaCj\xe0\x87\xdat\xa1'a\xda\xc0 \x01\x1a\
↪x9e\xdd\xc4\x90\x0b\xf1;")
HexBytes('0x03087766bf68e78671d1ea436ae087da74a12761dac020011a9eddc4900bf13b')

# HexBytes accepts the hex string representation as well, ignoring case and 0x prefixes
>>> hb = HexBytes('03087766BF68E78671D1EA436AE087DA74A12761DAC020011A9EDDC4900BF13B')
>>> hb
HexBytes('0x03087766bf68e78671d1ea436ae087da74a12761dac020011a9eddc4900bf13b')
>>> hb = HexBytes('0x03087766BF68E78671D1EA436AE087DA74A12761DAC020011A9EDDC4900BF13B')
>>> hb
HexBytes('0x03087766bf68e78671d1ea436ae087da74a12761dac020011a9eddc4900bf13b')

# HexBytes does not override the .hex() or __str__ methods of the parent bytes type
>>> hb = HexBytes('03087766BF68E78671D1EA436AE087DA74A12761DAC020011A9EDDC4900BF13B')
>>> hb.hex()
'03087766bf68e78671d1ea436ae087da74a12761dac020011a9eddc4900bf13b'
>>> print(hb)
b"\x03\x08wf\xbfh\xe7\x86q\xd1\xeaCj\xe0\x87\xdat\xa1'a\xda\xc0 \x01\x1a\x9e\xdd\xc4\x90\
↪x0b\xf1;"

# Use the to_0x_hex method to get a 0x-prefixed hex string
>>> hb.to_0x_hex()
'0x03087766bf68e78671d1ea436ae087da74a12761dac020011a9eddc4900bf13b'


# get the first byte:
>>> hb[0]
3
```

(continues on next page)

(continued from previous page)

```
# get the first 5 bytes:
>>> hb[:5]
HexBytes('0x03087766bf')

# show how many bytes are in the value
>>> len(hb)
32

# cast back to the basic bytes type
>>> bytes(hb)
b"\x03\x08wf\xbfh\xe7\x86q\xd1\xeaCj\xe0\x87\xdat\xa1'a\xda\xc0 \x01\x1a\x9e\xdd\xc4\x90\
↪x0b\xf1;"
```

1.2 HexBytes

class hexbytes.main.**HexBytes**(*val: bool | bytearray | bytes | int | str | memoryview*)

Bases: bytes

Thin wrapper around the python built-in bytes class.

It has these changes:

1. Accepts more initializing values: bool, bytearray, bytes, (non-negative) int, str, and memoryview
2. The representation at console (__repr__) is 0x-prefixed
3. to_0x_hex returns a 0x-prefixed hex string

to_0x_hex() → str

Convert the bytes to a 0x-prefixed hex string

1.3 Release Notes

1.3.1 hexbytes v1.2.0 (2024-03-19)

Features

- Add to_0x_hex() method to provide a quick, explicit way to get an 0x-prefixed string (#43)

1.3.2 hexbytes v1.1.0 (2024-03-01)

Internal Changes - for hexbytes Contributors

- Change the name of master branch to main (#40)
- Merge template updates, notably adding py312 support (#41)

1.3.3 hexbytes v1.0.0 (2023-11-07)

Breaking Changes

- Move HexBytes prepend of `0x` from `hex` method to `__repr__` to not break `hex` of parent `bytes` class (#38)
- Drop python 3.7 support (#39)

Internal Changes - for hexbytes Contributors

- Added missing *build* dependency. (#32)
- Add `build.os` config for readthedocs (#34)
- Change to using `pre-commit` to manage linting tools (#35)
- Merge project template updates and bump mypy to v1.5.1 (#36)
- Merge template - .gitignore updates and other fixes (#37)
- Merge template updates, including additional linting, move most lint config to pyproject.toml (#39)

1.3.4 HexBytes v0.3.1 (2023-06-08)

Improved Documentation

- pull in ethereum-python-project-template updates (#31)

Internal Changes - for hexbytes Contributors

- pull in ethereum-python-project-template updates (#31)

1.3.5 HexBytes v0.3.0 (2022-08-17)

Miscellaneous changes

- #28

Breaking changes

- Drop support for Python 3.6, update Sphinx doc dependency requirement (#27)

1.3.6 HexBytes v0.2.3 (2022-08-11)

Features

- Type signature now accepts *memoryview* when creating HexBytes. It converts to a *bytes* internally, so not any performance benefit. But at least it's *possible* now, and mypy stops complaining. (#22)

Performance improvements

- Improve speed of the check for 0x at the beginning of the hex string. Similar to changes in eth-utils. (Technically merged by #22, but first posted in #21) (#21)

Internal Changes - for hexbytes Contributors

- Merged in latest changes from project template (#24)

Miscellaneous changes

- #25, #26

1.3.7 HexBytes v0.2.2 (2021-08-25)

Miscellaneous changes

- Pass mypy tests with *–no-implicit-reexport* (#15)

Internal Changes - for hexbytes Contributors

- Merge in template changes from the last year. Pass pydocstyle tests at the new major version. (#16)

1.3.8 HexBytes v0.2.1 (2020-06-02)

Features

- Officially support bytearray, int, and bool as inputs to `HexBytes`. Drop the dependency on eth-utils, for a much smaller & faster install. (#12)

1.3.9 v0.2.0

Released June 3, 2019

- **Breaking Changes**
  - Dropped Python3.5 support (only in name at this release, but py3.6 features will be used soon #10
- Features
  - A slice of HexBytes will now produce another HexBytes object #9
- Maintenance
  - Added type hints #7

1.3.10 v0.1.0

Released Mar 1, 2018

- Marked stable
- eth-utils v1.0.1 support

1.3.11 v0.1.0-beta.1

Released Feb 21, 2018

- pypy3 support
- eth-utils v1-beta.2 support
- Some generic template updates

1.3.12 v0.1.0-beta.0

Released Jan 30, 2018

- Tested a basic integration with eth-rlp
- Given the simplicity of the project and the longer usage history in web3.py, it is reasonable to bump to beta immediately.

1.3.13 v0.1.0-alpha.2

Released Jan 30, 2018

- Added hypothesis tests
- Added some docs
- Update eth-utils to get all required functionality
- Passes all tests

1.3.14 v0.1.0-alpha.1

- Launched repository, claimed names for pip, RTD, github, etc

1.4 Contributing

Thank you for your interest in contributing! We welcome all contributions no matter their size. Please read along to learn how to get started. If you get stuck, feel free to ask for help in Ethereum Python Discord server.

1.4.1 Setting the stage

To get started, fork the repository to your own github account, then clone it to your development machine:

```
git clone git@github.com:your-github-username/hexbytes.git
```

Next, install the development dependencies. We recommend using a virtual environment, such as virtualenv.

```
cd hexbytes
virtualenv -p python venv
. venv/bin/activate
python -m pip install -e ".[dev]"
pre-commit install
```

1.4.2 Running the tests

A great way to explore the code base is to run the tests.

We can run all tests with:

```
pytest tests
```

1.4.3 Code Style

We use pre-commit to enforce a consistent code style across the library. This tool runs automatically with every commit, but you can also run it manually with:

```
make lint
```

If you need to make a commit that skips the `pre-commit` checks, you can do so with `git commit --no-verify`.

This library uses type hints, which are enforced by the `mypy` tool (part of the `pre-commit` checks). All new code is required to land with type hints, with the exception of code within the `tests` directory.

1.4.4 Documentation

Good documentation will lead to quicker adoption and happier users. Please check out our guide on how to create documentation for the Python Ethereum ecosystem.

1.4.5 Pull Requests

It's a good idea to make pull requests early on. A pull request represents the start of a discussion, and doesn't necessarily need to be the final, finished submission.

GitHub's documentation for working on pull requests is available here.

Once you've made a pull request take a look at the Circle CI build status in the GitHub interface and make sure all tests are passing. In general pull requests that do not pass the CI build yet won't get reviewed unless explicitly requested.

If the pull request introduces changes that should be reflected in the release notes, please add a *newsfragment* file as explained here.

If possible, the change to the release notes file should be included in the commit that introduces the feature or bugfix.

1.4.6 Releasing

Releases are typically done from the `main` branch, except when releasing a beta (in which case the beta is released from `main`, and the previous stable branch is released from said branch).

Final test before each release

Before releasing a new version, build and test the package that will be released:

```
git checkout main && git pull

make package-test
```

This will build the package and install it in a temporary virtual environment. Follow the instructions to activate the venv and test whatever you think is important.

You can also preview the release notes:

```
towncrier --draft
```

Build the release notes

Before bumping the version number, build the release notes. You must include the part of the version to bump (see below), which changes how the version number will show in the release notes.

```
make notes bump=$$VERSION_PART_TO_BUMP$$
```

If there are any errors, be sure to re-run make notes until it works.

Push the release to github & pypi

After confirming that the release package looks okay, release a new version:

```
make release bump=$$VERSION_PART_TO_BUMP$$
```

This command will:

- Bump the version number as specified in `.pyproject.toml` and `setup.py`.
- Create a git commit and tag for the new version.
- Build the package.
- Push the commit and tag to github.
- Push the new package files to pypi.

Which version part to bump

`$$VERSION_PART_TO_BUMP$$` must be one of: `major`, `minor`, `patch`, `stage`, or `devnum`.

The version format for this repo is `{major}.{minor}.{patch}` for stable, and `{major}.{minor}.{patch}-{stage}.{devnum}` for unstable (`stage` can be alpha or beta).

If you are in a beta version, `make release bump=stage` will switch to a stable.

To issue an unstable version when the current version is stable, specify the new version explicitly, like `make release bump="--new-version 4.0.0-alpha.1"`

You can see what the result of bumping any particular version part would be with `bump-my-version show-bump`

1.5 Code of Conduct

1.5.1 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, gender identity and expression, level of experience, education, socio-economic status, nationality, personal appearance, race, religion, or sexual identity and orientation.

1.5.2 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

1.5.3 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.

1.5.4 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.

1.5.5 Enforcement

Instances of abusive, harassing, or otherwise unacceptable behavior may be reported by contacting the project team at snakecharmers@ethereum.org. 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.

1.5.6 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

PYTHON MODULE INDEX

h

INDEX

H

M

T