The Journal of Open Source Software

Type Enforced: A Python type enforcer for type annotations

Connor Makowski [1¶], **Willem Guter** [1], **and Timothy Russell** [1]

1 Massachusetts Institute of Technology Cambridge, United States ¶ Corresponding author

DOI: 10.xxxxxx/draft

Software

- Review
- Repository
- Archive

Editor: Open Journals

Reviewers:

- @openjournals

Submitted: 01 January 1970
Published: unpublished

License
Authors of papers retain copyright and release the work under a Creative Commons Attribution 4.0 International License (CC BY 4.0).

DRAFT

TODO - [T] Check your name - [ ] Who else are we going to include on this? - [ ] I read this as Connor needs to be the corresponding author - [T] Check the list of tags

JOSS Software Requirements - [X] Be stored in a repository that can be cloned without registration. - [X] Be stored in a repository that is browsable online without registration. - [ ] Have an issue tracker that is readable without registration. - [ ] Permit individuals to create issues/file tickets against your repository. —

Summary

`type_enforced` is a pure Python package designed to enforce type annotations at runtime without the need for a special compiler. It provides an intuitive decorator-based interface that allows developers to enforce explicit typing constraints on function and method inputs, return types, dataclasses, and class instances. The package supports a comprehensive set of Python's built-in types, typing module constructs (such as `List`, `Dict`, `Union`, `Optional`, and `Literal`), nested data structures, and custom constraints. By offering runtime validation of type annotations and constraints, `type_enforced` enhances code reliability, readability, and maintainability.

Statement of Need

Python's dynamic typing system offers flexibility but can lead to runtime errors that are difficult to diagnose in complex scientific software and research applications. Static type checking tools such as Mypy provide valuable compile-time validation; however, they do not prevent runtime type errors. Existing runtime enforcement libraries often require extensive boilerplate code or lack support for advanced typing features and nested structures.

The `type_enforced` package addresses these limitations by providing robust runtime enforcement of Python type annotations with minimal overhead. It supports advanced typing features including nested iterables, union types, dataclasses, inheritance-based validation (`WithSubclasses`), and custom constraints (`Constraint`, `GenericConstraint`). This makes it particularly suitable for research software development where correctness of data types is critical for reproducibility and reliability.

Functionality and Features

Key features provided by the package include: - **Decorator-based enforcement**: Easily apply enforcement to functions, methods, classes, static methods, class methods, and dataclasses. - **Comprehensive typing support**: Supports built-in Python types (`int`, `str`, `list`, `dict`, etc.), typing module constructs (`List`, `Dict`, `Union`, `Optional`, `Literal`), union types (`int | float`), nested structures (`dict[dict[int]]`), and deeply nested iterables (`list[set[str]]`).

- **Custom constraints**: Validate input values with built-in constraint classes (e.g., numerical bounds) or user-defined generic constraints (e.g., membership in a predefined set). - **Inheritance-aware validation**: Validate instances against class hierarchies using the provided utility class (`WithSubclasses`). - **Flexible enable/disable mechanism**: Enable or disable enforcement selectively at the function or class level to accommodate debugging versus production environments.

Summary

The forces on stars, galaxies, and dark matter under external gravitational fields lead to the dynamical evolution of structures in the universe. The orbits of these bodies are therefore key to understanding the formation, history, and future state of galaxies. The field of "galactic dynamics," which aims to model the gravitating components of galaxies to study their structure and evolution, is now well-established, commonly taught, and frequently used in astronomy. Aside from toy problems and demonstrations, the majority of problems require efficient numerical tools, many of which require the same base code (e.g., for performing numerical orbit integration).

`Gala` is an Astropy-affiliated Python package for galactic dynamics. Python enables wrapping low-level languages (e.g., C) for speed without losing flexibility or ease-of-use in the user-interface. The API for `Gala` was designed to provide a class-based and user-friendly interface to fast (C or Cython-optimized) implementations of common operations such as gravitational potential and force evaluation, orbit integration, dynamical transformations, and chaos indicators for nonlinear dynamics. `Gala` also relies heavily on and interfaces well with the implementations of physical units and astronomical coordinate systems in the `Astropy` package (Astropy Collaboration et al., 2013) (`astropy.units` and `astropy.coordinates`).

`Gala` was designed to be used by both astronomical researchers and by students in courses on gravitational dynamics or astronomy. It has already been used in a number of scientific publications (Pearson et al., 2017) and has also been used in graduate courses on Galactic dynamics to, e.g., provide interactive visualizations of textbook material (Binney & Tremaine, 2008). The combination of speed, design, and support for Astropy functionality in `Gala` will enable exciting scientific explorations of forthcoming data releases from the *Gaia* mission (Gaia Collaboration et al., 2016) by students and experts alike. The source code for `Gala` has been archived to Zenodo with the linked DOI: (Price-Whelan et al., 2017)

Acknowledgements

We acknowledge contributions from Brigitta Sipocz, Syrtis Major, and Semyeong Oh, and support from Kathryn Johnston during the genesis of this project.

References

Astropy Collaboration, Robitaille, T. P., Tollerud, E. J., Greenfield, P., Droettboom, M., Bray, E., Aldcroft, T., Davis, M., Ginsburg, A., Price-Whelan, A. M., Kerzendorf, W. E., Conley, A., Crighton, N., Barbary, K., Muna, D., Ferguson, H., Grollier, F., Parikh, M. M., Nair, P. H., … Streicher, O. (2013). Astropy: A community Python package for astronomy. *558*, A33. https://doi.org/10.1051/0004-6361/201322068

Binney, J., & Tremaine, S. (2008). *Galactic Dynamics: Second Edition*. Princeton University Press.

Gaia Collaboration, Prusti, T., de Bruijne, J. H. J., Brown, A. G. A., Vallenari, A., Babusiaux, C., Bailer-Jones, C. A. L., Bastian, U., Biermann, M., Evans, D. W., & al., et. (2016). The Gaia mission. *595*, A1. https://doi.org/10.1051/0004-6361/201629272

Pearson, S., Price-Whelan, A. M., & Johnston, K. V. (2017). Gaps in Globular Cluster Streams: Pal 5 and the Galactic Bar. *ArXiv e-Prints*. https://arxiv.org/abs/1703.04627

Price-Whelan, A., Sipocz, B., Major, S., & Oh, S. (2017). *Adrn/gala: v0.2.1*. https://doi.org/10.5281/zenodo.833339

DRAFT