The Journal of Open Source Software

Hickle: A HDF5-based python pickle replacement

Danny C. Price[1, 2], **Ellert van der Velden**[2], **Sébastien Celles**[3], **Pieter T. Eendebak**[4, 5], **Michael M. McKerns**[6], **Eben M. Olson**[7], **Colin Raffel**[8], **Bairen Yi**[9], **and Elliott Ash**[10]

1 Department of Astronomy, University of California Berkeley, Berkeley CA 94720 **2** Centre for Astrophysics & Supercomputing, Swinburne University of Technology, Hawthorn, VIC 3122, Australia **3** Thermal Science and Energy Department, Institut Universitaire de Technologie de Poitiers - Université de Poitiers, France **4** QuTech, Delft University of Technology, P.O. Box 5046, 2600 GA Delft, The Netherlands **5** Netherlands Organisation for Applied Scientific Research (TNO), P.O. Box 155, 2600 AD Delft, The Netherlands **6** Institute for Advanced Computational Science, Stony Brook University, Stony Brook, NY 11794-5250 **7** Department of Laboratory Medicine, Yale University, New Haven CT 06510 USA **8** Google Brain, Mountain View, CA, 94043 **9** The Hong Kong University of Science and Technology **10** ETH Zurich

DOI: 10.21105/joss.01115

Software

- Review
- Repository
- Archive

Submitted: 13 November 2018
Published: 17 December 2018

Summary

`hickle` is a Python 2/3 package for quickly dumping and loading python data structures to Hierarchical Data Format 5 (HDF5) files (n.d.). When dumping to HDF5, `hickle` automatically convert Python data structures (e.g. lists, dictionaries, `numpy` arrays (Oliphant, 2007)) into HDF5 groups and datasets. When loading from file, `hickle` automatically converts data back into its original data type. A key motivation for `hickle` is to provide high-performance loading and storage of scientific data in the widely-supported HDF5 format.

`hickle` is designed as a drop-in replacement for the Python `pickle` package, which converts Python object hierarchies to and from Python-specific byte streams (processes known as 'pickling' and 'unpickling' respectively). Several different protocols exist, and files are not designed to be compatible between Python versions, nor interpretable in other languages. In contrast, `hickle` stores and loads files from HDF5, for which application programming interfaces (APIs) exist in most major languages, including C, Java, R, and MATLAB.

Python data structures are mapped into the HDF5 abstract data model in a logical fashion, using the `h5py` package (Collette, 2013). Metadata required to reconstruct the hierarchy of objects, and to allow conversion into Python objects, is stored in HDF5 attributes. Most commonly used Python iterables (dict, tuple, list, set), and data types (int, float, str) are supported, as are `numpy` N-dimensional arrays. Commonly-used `astropy` data structures and `scipy` sparse matrices are also supported.

`hickle` has been used in many scientific research projects, including:

- Visualization and machine learning on volumetric fluorescence microscopy datasets from histological tissue imaging (Durant, Olson, Schulz, & Torres, 2017).
- Caching pre-computed features for MIDI and audio files for downstream machine learning tasks (Raffel, 2016).

- Storage and transmission of high volume of shot-gun proteomics data, such as mass spectra of proteins and peptide segments (Zhang et al., 2016).
- Storage of astronomical data and calibration data from radio telescopes (Price, Greenhill, Fialkov, Bernardi, & others, 2018).

`hickle` is released under the MIT license, and is available from PyPi via `pip`; source code is available at https://github.com/telegraphic/hickle.

References

Collette, A. (2013). *Python and hdf5*. O'Reilly.

Durant, T. J. S., Olson, E. M., Schulz, W. L., & Torres, R. (2017). Very deep convolutional neural networks for morphologic classification of erythrocytes. *Clinical Chemistry*, *63*(12), 1847–1855. doi:10.1373/clinchem.2017.276345

Oliphant, T. E. (2007). Python for scientific computing. *Computing in Science Engineering*, *9*(3), 10–20. doi:10.1109/MCSE.2007.58

Price, D. C., Greenhill, L. J., Fialkov, A., Bernardi, G., & others. (2018). Design and characterization of the Large-aperture Experiment to Detect the Dark Age (LEDA) radiometer systems. *Monthly Notices of the Royal Astronomy Society*, *478*, 4193–4213. doi:10.1093/mnras/sty1244

Raffel, C. (2016). *Learning-based methods for comparing sequences, with applications to audio-to-midi alignment and matching* (PhD thesis). Columbia University.

Zhang, H., Chen, L., Yi, B., Chen, K., Chowdhury, M., & Geng, Y. (2016). CODA: Toward automatically identifying and scheduling coflows in the dark. In *Proceedings of the 2016 acm sigcomm conference*, SIGCOMM '16 (pp. 160–173). New York, NY, USA: ACM. doi:10.1145/2934872.2934880

(n.d.). Retrieved from https://support.hdfgroup.org/HDF5/doc/index.html