The Journal of Open Source Software

byteparsing: a functional parser combinator for mixed ASCII/binary data

Johan Hidding [1*¶] **and Pablo Rodríguez-Sánchez** [1*]

1 Netherlands eScience Center ¶ Corresponding author * These authors contributed equally.

DOI: 10.21105/joss.05293

Software

- Review
- Repository
- Archive

Editor: Aoife Hughes

Reviewers:

- @inakleinbottle
- @dvberkel

Submitted: 08 December 2022
Published: 21 April 2023

Summary

Byteparsing is a functional parser combinator for Python. It was originally motivated by the problem of parsing OpenFOAM files. OpenFOAM[1] is a free, open source computational fluid dynamics software package whose input and output files contain both ASCII and binary data. Other common formats combining ASCII and binary are PLY triangle data or PPM images[2]. This makes them particularly hard to parse using traditional tools. Byteparsing is a flexible tool capable of dealing with generic formats.

Statement of need

There are already a few accessible options for parsing and manipulating data in Python. For instance:

- `pyparsing` is the *de facto* standard for text parsing in Python. It seems to have no features for dealing with binary data though.
- `construct` deals mostly with pure binary data.
- `Kaitai Struct` and `Antlr` both require a large time investment to learn.

The major downside of the remaining binary parser Python packages we could find is that they focus mostly either on parsing network traffic or on data structures that can be described in a fixed declarative language.

Our research problem, namely, manipulating OpenFOAM files from Python, required a parser with the following characteristics:

- Capable of dealing with files combining ASCII and binary.
- Easy to program, using concepts similar to those found in other functional parser combinators like `pyparsing`.
- Composable and testable at all levels of complexity.
- Capable of dealing transparently with Python objects that support the buffer protocol (i.e., memory mapped file access is trivially supported).
- Performant enough, considering the use case where we have small ASCII headers and large contiguous blocks of floating point data.
- Compliant with best practices, such as automated unit testing and thorough documentation.

Byteparser is the solution we developed based on this list of requirements.

[1] https://www.openfoam.com/
[2] https://parallelwindfarms.github.io/byteparsing/advanced.html

A note on architecture

Writing functional parser combinators is a staple of functional languages like Haskell or Ocaml (Frost & Launchbury, 1989; Hutton, 1992). The paper "Monadic Parsing in Haskell" (Hutton & Meijer, 1998) gives a complete tutorial on how to write a basic recursive descent parser. Most of what Hutton and Meijer teach carries over nicely to Python once we take care of a few details. We've replaced some Haskell idioms by features that are considered more 'pythonic'.

An extended description of the concept of functional parser combinators can be found in the documentation[3]. Those readers more interested in starting working right away will probably find our lists of examples[4] very practical.

Conclusion

In research software it is unfortunately still quite common to encounter non-standard data formats. For those data formats where a mix of ASCII and binary parsing is needed, Byteparsing can make a useful addition to the existing landscape of parser libraries in Python. Development of a parser using Byteparsing can be relatively quick, as it is easy to build up parsers from smaller testable components.

Acknowledgements

This project was supported by funding from the Netherlands eScience Center and NWO as part of the Joint Call for Energy Research, Project Number CSER.JCER.025. We also want to acknowledge Dr. Nicolas Renaud for his support and suggestions.

References

Frost, R., & Launchbury, J. (1989). Constructing natural language interpreters in a lazy functional language. *The Computer Journal*, *32*(2), 108–121. https://doi.org/10.1093/comjnl/32.2.108

Hutton, G. (1992). Higher-order functions for parsing. *Journal of Functional Programming*, *2*(3), 323–343. https://doi.org/10.1017/S0956796800000411

Hutton, G., & Meijer, E. (1998). Monadic parsing in Haskell. *Journal of Functional Programming*, *8*(4), 437–444. https://doi.org/10.1017/S0956796898003050

[3]https://parallelwindfarms.github.io/byteparsing/functional.html
[4]https://parallelwindfarms.github.io/byteparsing/examples.html