Beancount History and Credits

[Martin Blais](http://plus.google.com/+MartinBlais), July 2014

<http://furius.ca/beancount/doc/history>

*A history of the development of Beancount and credits for contributors..*

# History of Beancount

John Wiegley's Ledger was the inspiration for the first version of Beancount. His system is where much of the original ideas for this system came from. When I first learned about double-entry bookkeeping and realized that it could be the perfect method to solve many of the tracking problems I was having in counting various installments for my company, and after a quick disappointment in the solutions that were available at the time (including GnuCash, which I could break very easily), I was quickly led to the Ledger website. There, John laid out his vision for a text-based system, in particular, the idea of doing away with credits and debits and just the signs, and the basics of a convenient input syntax which allows you to omit the amount of one of the postings. I got really excited and had various enthusiastic discussions with him about Ledger and how I wanted to use it. There was some cross-pollination of ideas and John was very receptive to proposals for adding new features.

I was so intensely curious about bookkeeping that I began writing a Python interface to Ledger. But in the end I found myself rewriting the entire thing in Python–not for dislike of Ledger but rather because it was simple enough that I could do most of it in little time, and immediately add some features I thought would be useful. One reason for doing so was that instead of parsing the input file every time and generating one report to the console, I would parse it once and then serve the various reports from the in-memory database of transactions, requested via a web page. Therefore, I did not need processing speed, so having to use C++ for performance reasons was not necessary anymore, and I chose to just stick with a dynamic language, which allowed me to add many features quickly. This became Beancount version 1, which stood on its own and evolved its own set of experimental features.

My dream was to be able to quickly and easily manipulate these transaction objects to get various views and breakdowns of the data. I don't think the first implementation pushed the limits far enough, however; the code was substandard, to be honest—I wrote it really quickly—and making modifications to the system was awkward. In particular, the way I originally implemented the tracking of capital gains was inelegant and involved some manual counting. I was unhappy with this, but it worked. It was also using ugly ad-hoc parser code in order to remain reasonably compatible with Ledger syntax—I thought it would be interesting to be able to share some common input syntax and use either system for validation and maybe even to convert between them–and that made me wary of making modifications to the syntax to evolve new features, so it stabilized for a few years and I lost interest in adding new features.

But it was correct and it worked, mostly, so I used the system continuously from 2008 to 2012 to manage my own personal finances, my company's finances, and joint property with my wife, with detailed transactions; this was great. I learned a lot about how to keep books during that time (the cookbook document is meant to help you do the same). In the summer of 2013, I had an epiphany and realized a correct and generalizable way to implement capital gains, how to merge the tracking of positions held at a cost and regular positions, and a set of simple rules for carrying out operations on them sensibly (the design of how inventories work). I also saw a better way to factor out the internal data structures, and decided to break from the constraint of compatibility with Ledger and redesign the input syntax in order to parse the input language using a lex/yacc generator, which would allow me to easily evolve the input syntax without having to deal with parsing issues, and to create ports to other languages more easily. In the process, a much simpler and more consistent syntax emerged, and in a fit of sweat and a few intense weekends I re-implemented the entire thing from scratch, without even looking at the previous version, clean-room. Beancount version 2 was born, much better than the last, modular, and easy to extend with plugins.

The result is what I think is an elegant design involving a small set of objects, a design that could easily be a basis for reimplementation in other computer languages. This is described in the accompanying design doc, for those who would have an interest in having a go at it (this would be welcome and I'm expecting this will happen). While Ledger remains an interesting project bubbling with ideas for expressing the problem of bookkeeping, the second version of Beancount proposes a simpler design that leaves out features that are not strictly necessary and aims at maximum usability through a simple web interface and a very small set of command-line options. Since I had more than 5 years worth of real-world usage experience with the first version, I set a goal for myself to remove all the features that I thought weren't actually useful and introduced unnecessary complexity (like virtual transactions, allowing accounts not in the five types, etc.), and to simplify the system as much as possible without compromising on its usability. The resulting language and software were much simpler to use and understand, the resulting data structures are much simpler to use, the processing more “functional” in nature, and the internals of Beancount are very modular. I converted all my 6 years worth of input data–thanks to some very simple Python scripts to manipulate the file–and began using the new version exclusively. It is now in a fully functional state.

Ledger's syntax implements many features that trigger a lot of implicitly-defined behaviour; I find this confusing and some the reasons for this are documented in the many improvement proposals. I don’t like command-line options. In contrast, Beancount's design provides a less expressive, lower-level syntax but one that closely matches the generated in-memory data structure, and that is hopefully more explicit in that way. I think both projects have strengths and weaknesses. Despite its popularity, the latest version of Ledger remains with a number of shortcomings in my view. In particular, reductions in positions are not booked at the moment they occur, but rather they appear to simply accumulate and get matched only at display time, using different methods depending on the command-line options. Therefore, it is possible in Ledger to hold positive and negative lots of the same commodity in an account simultaneously. I believe that this can lead to confusing and even [incorrect accounting](https://groups.google.com/d/msg/ledger-cli/aQvbjTZa7HE/iMisMBkaI6UJ) for trading lots. I think this is still being figured out by the Ledger look-alikes community and that they will eventually converge to the same solution I have, or perhaps even figure out a better solution.

In the redesign, I separated out configuration directives that I had used for importing and over many iterations eventually figured out an elegant way to mostly automate imports and automatically detect the various input files and convert them into my input syntax. The result is the [LedgerHub design doc](https://docs.google.com/document/d/11u1sWv7H7Ykbc7ayS4M9V3yKqcuTY7LJ3n1tgnEN2Hk/). LedgerHub is currently implemented and I’m using it exclusively, but has insufficient testing for me to stamp a public release [July 2014]. You are welcome to try it out for yourself.

In June and July 2014, I decided to dump seven years’ worth of thinking about command-line accounting in a set of Google Docs and this now forms the basis for the current documentation of Beancount. I hope to be evolving it gradually from here on.

# Chronology

* Ledger was begun in August 2003  
  [http://web.archive.org/web/\*/http://www.newartisans.com/ledger/](http://web.archive.org/web/*/http://www.newartisans.com/ledger/)
* Beancount was begun in 2008  
  [http://web.archive.org/web/\*/furius.ca/beancount](http://web.archive.org/web/*/furius.ca/beancount)
* HLedger was also begun in 2008  
  <https://github.com/simonmichael/hledger/graphs/contributors?from=2007-01-21&to=2014-09-08&type=c>

# Credits

So far I’ve been contributing all the code to Beancount. Some users have made significant contributions in other ways:

* Daniel Clemente has been reporting all the issues he came across while using Beancount to manage his company and personal finances. His relentless perseverance and attention to detail has helped me put a focus on fixing the rough corners of Beancount that I knew to avoid myself.
* After many years of prodding, my old friend [Filippo Tampieri](http://plus.google.com/+FilippoTampieri) has finally decided to convert his trading history in Beancount format. He has contributed a number of sophisticated reviews of my documentation and is working on adding various methods for evaluating returns on assets.