Installing Beancount

[Martin Blais](mailto:blais@furius.ca) - Updated: June 2015

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

*Instructions for downloading and installing Beancount on your computer.*

# Releases

Beancount is a mature project: the first version was written in 2008. The current rewrite of Beancount is stable. (Technically, this is what I call version 2.x beta).

I’m still working on this Beancount code every weekend these days, so it is very much in active development and evolving, though the great majority of the basic features are basically unchanging. I’ve built an extensive suite of tests so you can consider the “default” branch of the repository as stable. New features are developed in branches and only merged in the “default” branch when fully stable (the entire battery of tests passes without failures). Changes to “default” are posted to the [CHANGES](https://bitbucket.org/blais/beancount/src/tip/CHANGES) file and a corresponding email is sent to the [mailing-list](https://groups.google.com/forum/#!forum/beancount).

***So I’m not cutting releases yet.*** You have to install or run from source. I don’t update the [PyPI page](https://pypi.python.org/pypi/beancount) actively either.

Until development slows down and I decide to bake numbered releases, I recommend cloning and updating regularly from the bitbucket repository and building and running in place the “default” branch (if you’re not familiar with Mercurial, “default” is the default, same as master under Git).

# Where to Get It

This is the official location for the source code:

[http](https://www.google.com/url?q=https%3A%2F%2Fbitbucket.org%2Fblais%2Fbeancount%2F&sa=D&sntz=1&usg=AFQjCNHG5OW0QtMlh8rKR1VKgAMeQGZQfw)[s](https://bitbucket.org/blais/beancount/)[://bitbucket.org/blais/beancount/](https://www.google.com/url?q=https%3A%2F%2Fbitbucket.org%2Fblais%2Fbeancount%2F&sa=D&sntz=1&usg=AFQjCNHG5OW0QtMlh8rKR1VKgAMeQGZQfw)

Download it like this, by using Mercurial to make a clone on your machine:

**hg clone** [**https://bitbucket.org/blais/beancount**](https://bitbucket.org/blais/beancount)

The author tries to maintain a [GitHub](http://github.com) clone for this project, but it’s often a bit out-of-date:

**git clone** [**https://github.com/beancount/beancount**](https://github.com/beancount/beancount)

Other GitHub clones that can be found in the wild are undoubtedly out of date. Don’t use them.

# How to Install

## Installing Python

Beancount uses Python 3.5[[1]](#footnote-0) or above, which is a pretty recent version of Python (as of this writing), and a few common library dependencies. I try to minimize dependencies, but you do have to install a few. This is very easy.

First, you should have a working Python install. Install the latest stable version >=3.5 using the download from [python.org](http://python.org). Make sure you have the development headers and libraries installed as well (e.g., the “Python.h” header file). For example, on a Debian/Ubuntu system you would install the **python3-dev** package.

Beancount supports setuptools since Feb 2016, and you will need to install dependencies. You will want to have the “pip3” tool installed. It’s probably installed by default along with Python3—test this out by invoking “pip3” command. In any case, under a Debian/Ubuntu system you would simply install the **python3-pip** package.

*Note for* [*Arch Linux*](https://www.archlinux.org/)*: Arch’s Python package does not correctly install all of its runtime dependencies. With the Python installation that comes with it, you have to install the “mpdecimal” package in order for the C decimal implementation to work. See* [*this issue*](https://bitbucket.org/blais/beancount/issues/90/beancount-should-consider-warning-when-the) *for a discussion. Others say easiest might just be to use the* [*AUR package*](https://www.google.com/url?q=https://aur.archlinux.org/packages/?K%3Dbeancount&sa=D&ust=1481689409138000&usg=AFQjCNFJwPzC-OxbzSlvTanMYfqc_cizEw)*, which installs all the required dependencies.*

### Python Dependencies

Note that in order to build a full working Python install from source, you will probably need to install a host of other development libraries and their corresponding header files, e.g., libxml2, libxslt1, libgdbm, libmp, libssl, etc. Installing those is dependent on your particular distribution and/or OS. Just make sure that your Python installation has all the basic modules compiled for its default configuration.

## Installing Beancount using pip

This is the easiest way to install Beancount. You just install Beancount using

**sudo pip3 install beancount**

This should automatically download and install all the dependencies.

Note, however, that this will install the latest version that was pushed to the [PyPI repository](https://pypi.python.org/pypi/beancount/) and not the very latest version available from source. Releases to PyPI are made sporadically but frequently enough not to be too far behind. Consult the [CHANGES file](https://bitbucket.org/blais/beancount/src/tip/CHANGES) if you’d like to find out what is not included since the release date.

## Installing Beancount from Source

Installing from source offers the advantage of providing you with the very latest version of the stable branch (“default”). The default branch is as stable as the released version.

### Obtain the Source Code

Get the source code from the official repository:

**hg clone https://bitbucket.org/blais/beancount**

### Install third-party dependencies

You might need to install some non-Python library dependencies, such as libxml2-dev, libxslt1-dev, and perhaps a few more. Try to build, it should be obvious what’s missing. If on Ubuntu, use apt-get.

If installing on Windows, see the Windows section below.

### Install Beancount from source using pip3

You can then install all the dependencies and Beancount itself using pip:

**cd beancount  
sudo pip3 install .**

### Install Beancount from source using setup.py

First you’ll need to install dependent libraries. You can do this using pip:

**pip3 install python-dateutil bottle ply lxml python-magic[[2]](#footnote-1) beautifulsoup4**

Or equivalently, you may be able to do that using your distribution, e.g., on Ubuntu/Debian:

**sudo apt-get install python3-dateutil python3-bottle python3-ply python3-lxml python3-bs4 …**

Then, you can install the package in your Python library using the usual setup.py invocation:

**cd beancount  
sudo python3 setup.py install**

Or you can install the package in your user-local Python library using this:

**sudo python3 setup.py install --user**

Remember to add ~/.local/bin to your path to access the local install.

### Installing for Development

If you want to execute the source in-place for making changes to it, you can use the setuptools “develop” command to point to it:

**cd beancount  
sudo python3 setup.py develop**

Warning: This modifies a .pth file in your Python installation to point to the path to your clone. You may or may not want this.

The way I work myself is *old-school*; I build it locally and setup my environment to find its libraries. You build it like this:

**cd beancount  
python3 setup.py build\_ext -i**

Finally, both the PATH and PYTHONPATH environment variables need to be updated for it:

**export PATH=$PATH:/path/to/beancount/bin  
export PYTHONPATH=$PYTHONPATH:/path/to/beancount**

## Installing from Packages

Various distributions may package Beancount. Here are links to those known to exist:

* Arch: <https://aur.archlinux.org/packages/beancount/>

## Windows Installation

### Native

Installing this package by pip requires compiling some C++ code during the installation procedure which is only possible if an appropriate compiler is available on the computer, otherwise you will receive an error message about missing *vsvarsall.bat* or *cl.exe*.

To be able to compile C++ code for Python you will need to install the same major version of the C++ compiler as your Python installation was compiled with. By running *python* in a console and looking for a text similar to *[MSC v.1900 64 bit (AMD64)]* you can determine which compiler was used for your particular Python distribution. In this example it is *v.1900*.

Using this number find the required Visual C++ version [here](https://stackoverflow.com/questions/2676763/what-version-of-visual-studio-is-python-on-my-computer-compiled-with). Since different versions seem to be compatible as long as the first two digits are the same you can in theory use any Visual C++ compiler between 1900 and 1999.

According to my experience both Python 3.5 and 3.6 was compiled with MSC v.1900 so you can do either of the following to satisfy this requirement:

* Install the standalone [Build Tools for Visual Studio 2017](https://visualstudio.microsoft.com/thank-you-downloading-visual-studio/?sku=BuildTools&rel=15) or
* Install the standalone [Visual C++ Build Tools 2015](http://go.microsoft.com/fwlink/?LinkId=691126) or
* Modify an existing Visual Studio 2017 installation
  + Start the Visual Studio 2017 installer from *Add or remove programs*
  + Select *Individual components*
  + Check *VC++ 2017 version 15.9 v14.16 latest v141 tools* or newer under *Compilers, build tools, and runtimes*
  + Install
* Visual Studio 2019
  + add C++ build tools: C++ core features, MSVC v142 build tools

If cl.exe is not in your path after installation, run Developer Command Prompt for Visual Studio and run the commands there.

### With Cygwin

Windows installation is, of course, a bit different. It’s a breeze if you use Cygwin. You just have to prep your machine first. Here’s how.

* Install the latest [Cygwin](https://www.cygwin.com/). This may take a while (it downloads a lot of stuff), but it is well worth it in any case. But before you kick off the install, make sure the following packages are all manually enabled in the interface provided by setup.exe (they’re not selected by default):
  + python3
  + python3-devel
  + python3-setuptools
  + mercurial
  + make
  + gcc-core
  + flex
  + bison
  + lxml
  + ply
* Start a new Cygwin bash shell (there should be a new icon on your desktop) and install the pip3 installer tool by running this command:  
  **easy\_install-3.4 pip**Make sure you invoke the version of easy\_install which matches your Python version, e.g. easy\_install-3.5 if you have Python 3.5 installed, or more.

At this point, you should be able to follow the instructions from the previous sections as is, starting from “Install Beancount using pip”.

### With WSL

The newly released Windows 10 Anniversary Update brings WSL 'Windows Subsystem for Linux' with [bash on Ubuntu on Windows](https://blogs.msdn.microsoft.com/wsl/2016/07/08/bash-on-ubuntu-on-windows-10-anniversary-update/) (installation instructions and more at <https://msdn.microsoft.com/commandline/wsl/about>).

This makes beancount installation easy, from bash:

sudo apt-get install python3-pip  
sudo apt-get install python3-lxml  
sudo pip3 install m3-cdecimal  
sudo pip3 install beancount --pre

This is not totally “Windows compatible”, as it is running in a pico-process, but provides a convenient way to get the Linux command-line experience on Windows. *(Contrib: willwray)*

### Notes on lxml

Some users have reported problems installing lxml, and a solution: when installing lxml with pip (under Cygwin), using this may help:

STATIC\_DEPS=true pip install lxml

## Checking your Install

You should be able to run the binaries from [this document](http://furius.ca/beancount/doc/running). For example, running bean-check should produce something like this:

$ **bean-check**

usage: bean-check [-h] [-v] filename  
bean-check: error: the following arguments are required: filename

If this works, you can now go to the [tutorial](https://docs.google.com/document/d/1G-gsmwK551lSyuHboVLW3xbLhh99JfoKIbNnZSJxteE/edit?pli=1#) and begin learning how Beancount works.

## Reporting Problems

If you need to report a problem, either send email on the mailing-list or [file a ticket](https://bitbucket.org/blais/beancount/issues) on Bitbucket. Running the following command lists the presence and versions of dependencies installed on your computer and it might be useful to include the output of this command in your bug report:

bean-doctor checkdeps

# Editor Support

There is support for some editors available:

* Emacs support is [provided in the distribution](https://bitbucket.org/blais/beancount/src/tip/src/elisp/). See the [Getting Started](http://furius.ca/beancount/doc/getting-started) text for installation instruction.
* Support for [editing with Sublime](https://sublime.wbond.net/packages/Beancount) has been contributed by [Martin Andreas Andersen](https://groups.google.com/d/msg/beancount/WvlhcCjNl-Q/s4wOBQnRVxYJ). See [his github repo](https://github.com/draug3n/sublime-beancount).
* Support for editing with Vim has been contributed by [Nathan Grigg](https://github.com/nathangrigg). See [his GitHub repo](https://github.com/nathangrigg/vim-beancount).
* Visual Studio Code currently has two extensions available. Both have been tested on Linux.
  + [Beancount](https://marketplace.visualstudio.com/items?itemName=Lencerf.beancount), with syntax checking (bean-check) and support for accounts, currencies, etc. It not only allows selecting existing open accounts but also displays their balance and other metadata. Quite helpful!
  + [Beancount Formatter](https://marketplace.visualstudio.com/items?itemName=dongfg.vscode-beancount-formatter), which can format the whole document, aligning the numbers, etc. using bean-format.

# If You Have Problems

If you run into any installation problems, [file a ticket](https://bitbucket.org/blais/beancount/issues), [email me](mailto:blais@furius.ca), or hit the [mailing-list](https://groups.google.com/forum/#!forum/beancount).

# Post-Installation Usage

Normally the [scripts located under beancount/bin](https://bitbucket.org/blais/beancount/src/tip/bin/) should be automatically installed somewhere in your executable path. For example, you should be able to just run “bean-check” on the command-line.

# Appendix

**If everything worked, you can stop reading here.** Here I just discuss the various dependencies and why you need them (or why you don’t, some of them are optional). This is of interest to developers and some of this info might help troubleshoot problems if you encounter any.

## Notes on Dependencies

### Python 3.5 or greater

Python 3.5 is widely available at this point, released more than a year ago. However, my experience with open source distribution tells me that a lot of users are running on old machines that won’t be upgraded for a while. So for those users, you might have to install your own Python… don’t worry, installing Python manually is pretty straightforward.

On Mac, it can also be installed with one of the various package management suits, like Brew, e.g., with “brew install python3”.

On an old Linux, download the source from [python.org](https://www.python.org/downloads/), and then build it like this:

tar zxcf Python-3.5.2.tgz  
cd Python-3.5.2  
./configure  
make  
sudo make install

This should just work. I recommend you install the latest 3.x release (3.5.2 at the time of this writing).

Note: The reason I require at least version 3.5 is because the cdecimal library which supplies Beancount with its implementation of a Decimal representation has been added to the standard library. We need this library to represent all our numbers. Also, Beancount uses the “typing” type annotations which are included in 3.5 (note: You may be able to install “typing” explicitly if you use an older version; no guarantees however).

Some users have reported that there are distributions which package the cdecimal support for Python3 separately. This is the case for Arch, and I’ve witnessed missing cdecimal support in some Ubuntu installs as well. A check has been inserted into Beancount in December 2015 for this, and a warning should be issued if your Python installation does not have fast decimal numbers. If you are in such a situation, you can try to install cdecimal explicitly, like this:

sudo pip3 install m3-cdecimal

and Beancount will then use it.

### Python Libraries

If you need to install Python libraries, there are a few different ways. First, there is the easy way: there is a package management tool called “pip3” (or just “pip” for version 2 of Python) and you install libraries like this:

sudo pip3 install <library-name>

Installing libraries from their source code is also pretty easy: download and unzip the source code, and then run this command from its root directory:

sudo python3 setup.py install

### Just make sure the “python3” executable you run when you do that is the same one you will use to run Beancount.

### Here are the libraries Beancount depends on and a short discussion of why.

#### python-dateutil

This library provides Beancount with the ability to parse dates in various formats, it is very convenient for users to have flexible options on the command-line and in the SQL shell.

#### bottle

The Beancount web interface (invoked via bean-web) runs a little self-contained web server accessible locally on your machine. This is implemented using a tiny library that makes it easy to implement such web applications: [bottle.py](http://bottlepy.org/).

#### ply

The query client (bean-query) which is used to extract data tables from a ledger file depends on a parser generator: I use Dave Beazley’s popular [PLY](http://www.dabeaz.com/ply/) library (version 3.4 or above) because it makes it really easy to implement a custom SQL language parser.

#### lxml

#### A tool is provided to bake a static HTML version of the web interface to a zip file (bean-bake). This is convenient to share files with an accountant who may not have your software installed. The web scraping code that is used to do that used the lxml HTML parsing library.

### Python Libraries for Export (optional)

#### google-api-python-client

Some of the scripts I use to export outputs to Google Drive (as well as scripts to maintain and download documentation from it) are checked into the codebase. You don’t have to install these libraries if you’re not exporting to Google Drive; everything else should work fine without them. These are thus optional.

If you do install this library, it require recent (as of June 2015) installs of

* [google-api-python-client](https://github.com/google/google-api-python-client): A Python client library for Google's discovery based APIs.
* [oauth2client](https://github.com/google/oauth2client): This is a client library for accessing resources protected by OAuth 2.0, used by the Google API Python client library.
* [httplib2](https://github.com/jcgregorio/httplib2): A comprehensive HTTP client library. This is used by oauth2client.

These are best installed using the pip3 tool.

Update: as of 2016, you should be able to install all of the above like this:

pip3 install google-api-python-client

**IMPORTANT:** The support for Python3 is fairly recent, and if you have an old install you might experience some failures, e.g. with missing dependencies, such as a missing “gflags” import. Please install those from recent releases and you should be fine. You can install them manually.

### Python Libraries for Imports (Optional)

Support for importing identifying, extracting and filing transactions from externally downloaded files is built into Beancount. (This used to be in the LedgerHub project, which has been deprecated.) If you don’t take advantage of those importing tools and libraries, you don’t need these imports.

#### python-magic (optional)

Beancount attempt to identify the types of files using the stdlib “mimetypes” module and some local heuristics for types which aren’t supported, but in addition, it is also useful to install libmagic, which it will use if it is not present:

pip3 install python-magic

Note that there exists another, older library which provides libmagic support called “filemagic”. You need “python-magic” and not “filemagic.” More confusingly, under Debian the “python-magic” library is called “filemagic.”

#### Other Tools

It is expected that the user will build their own importers. However, Beancount will provide some modules to help you invoke various external tools. The dependencies you need for these depends on which tool you’ve configured your importer to use.

### Virtualenv Installation

If you’d like to use virtualenv, you can try this (suggestion by Remy X). First install Python 3.5 or beyond[[3]](#footnote-2):

sudo add-apt-repository ppa:fkrull/deadsnakes  
sudo apt-get update  
sudo apt-get install python3.5  
sudo apt-get install python3.5-dev  
sudo apt-get install libncurses5-dev

Then install and activate virtualenv:

sudo apt-get install virtualenv  
virtualenv -p /usr/bin/python3.5 bean  
source bean/bin/activate  
pip install package-name  
sudo apt-get install libz-dev[[4]](#footnote-3) # For lxml to build  
pip install lxml  
pip install beancount

## Development Setup

### Installation for Development

For development, you will want to avoid installing Beancount in your Python distribution and instead just modify your PYTHONPATH environment variable to run it from source:

export PYTHONPATH=/path/to/install/of/beancount

Beancount has a few compiled C extension modules. This is just portable C code and should work everywhere. For development, you want to compile the C modules in-place, within the source code, and load them from there. Build the C extension module in-place like this:

python3 setup.py build\_ext -i

Or equivalently, the root directory has a Makefile that does the same thing and that will rebuild the C code for the lexer and parser if needed (you need flex and bison installed for this):

make build

With this, you should be able to run the executables under the bin/ subdirectory. You may want to add this to your PATH as well.

### Dependencies for Development

Beancount needs a few more tools for development. If you’re reading this, you’re a developer, so I’ll assume you can figure out how to install packages, skip the detail, and just list what you might need:

* [**nosetests**](https://pypi.python.org/pypi/nose/) (**nose**, or **python3-nose**): This is the test driver to use for running all the unit tests. You definitely need this.
* [**GNU flex**](https://www.gnu.org/software/flex/): This lexer generator is needed if you intend to modify the lexer. It generated C code that chops up the input files into tokens for the parser to consume.
* [**GNU bison**](http://www.gnu.org/software/bison/): This old-school parser generator is needed if you intend to modify the grammar. It generates C code that parses the tokens generated by flex. (I like using old tools like this because they are pretty low on dependencies, just C code. It should work everywhere.)
* [**GNU tar**](http://www.gnu.org/software/tar/): You need this to test the archival capabilities of bean-bake.
* [**InfoZIP zip**](http://www.info-zip.org/) (comes with Ubuntu): You need this to test the archival capabilities of bean-bake.
* [**lxml**](http://lxml.de): This XML parsing library is used in the web tests. (Unfortunately, the built-in one fails to parse large XML files.)
* [**pylint**](http://www.pylint.org/) >= 1.2: I’m running all the source code through this linter. If you contribute code with the intent of it being integrated and released, it has to pass the linter tests (or I’ll have to make it pass myself).
* [**pyflakes**](https://github.com/pyflakes/pyflakes): I’m running all the source code through this logical error detector. If you contribute code with the intent of it being integrated and released, it has to pass those tests (or I’ll have to make it pass myself).
* [**snakefood**](http://furius.ca/snakefood/): I use snakefood to analyze the dependencies between the modules and enforce some ordering, e.g. core cannot import from plugins, for example. There’s a target to run it from the root Makefile.
* [**graphviz**](http://www.graphviz.org/): The dependency tree of all the modules is generated by snakefood but graphed by the graphviz tool. You need to install it if you want to look at dependencies.

I think that’s about it. You certainly don’t need *everything* above, but that’s the list of tools I use. If you find anything missing, please leave a comment, I may have missed something.

1. Some people have [reported](https://www.google.com/url?q=https://groups.google.com/d/msgid/beancount/51fec791-1c38-46e8-b152-08c49e7686c3%2540googlegroups.com?utm_medium%3Demail%26utm_source%3Dfooter&sa=D&ust=1461188801128000&usg=AFQjCNH8StkxqPYKLKMhir6YAz6rXCOKsQ) bugs with the cdecimal library in 3.4. I would recommend actually installing 3.5, which appears to have fixed the problem. Technically, 3.3 will still run, but I’ll deprecate it for 3.5 at some point, probably when the Ubuntu and Mac OS X distributions have it installed by default. [↑](#footnote-ref-0)
2. Under Debian this is called “filemagic”, not “python3-magic.” [↑](#footnote-ref-1)
3. Installing py3.5: <http://www.jianshu.com/p/4f4b2ed568f4> [↑](#footnote-ref-2)
4. If /usr/bin/ld: cannot find -lxxx: <http://eminzhang.blog.51cto.com/5292425/1285705> [↑](#footnote-ref-3)