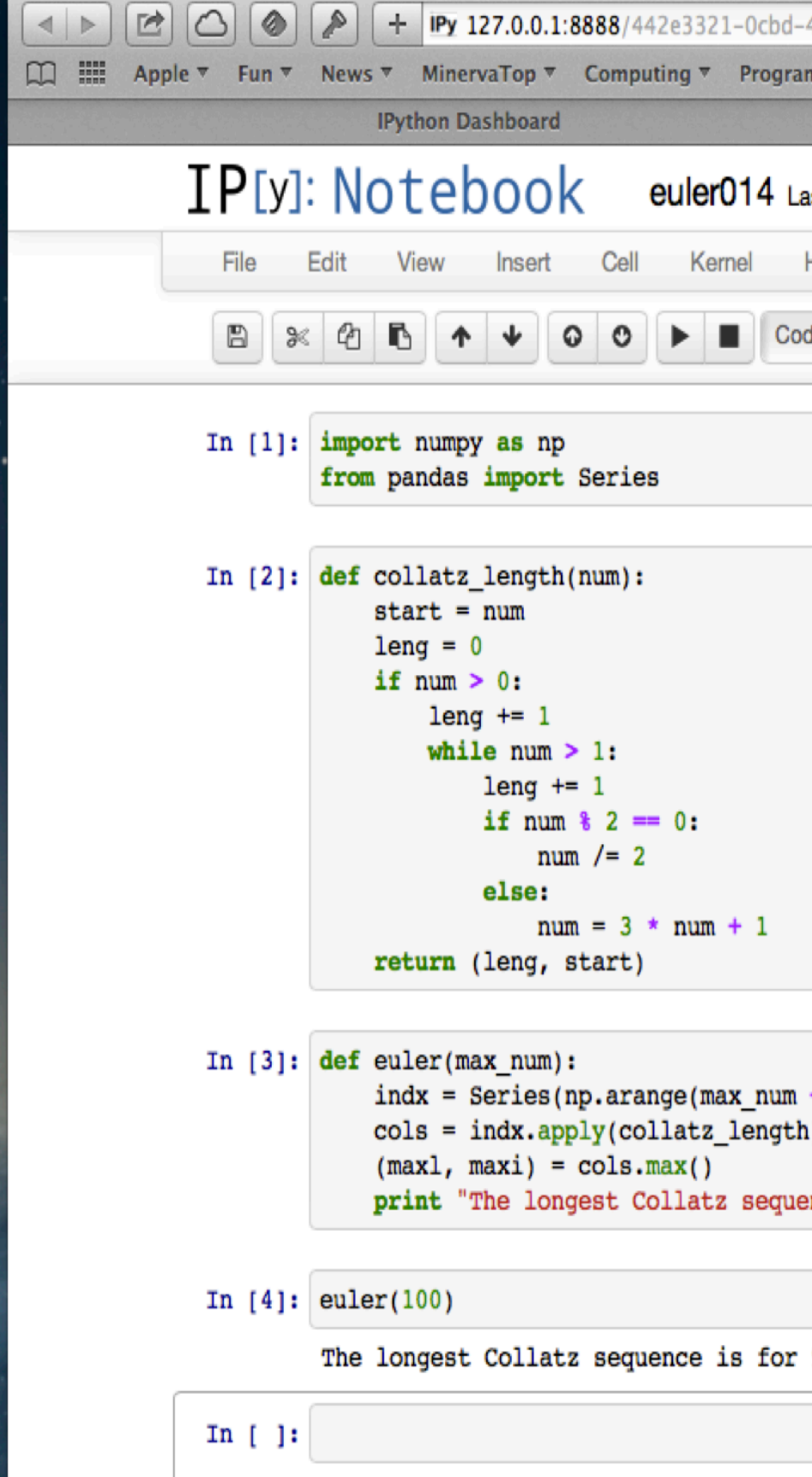


Setting Up Scientific Python for Mac OSX

Gabriel
Perdue



The screenshot shows a web-based IPython Notebook interface. At the top, there's a browser address bar with the URL 'IPy 127.0.0.1:8888/442e3321-0cbd-4...'. Below the browser, there's a navigation bar with links like 'Apple', 'Fun', 'News', 'MinervaTop', 'Computing', and 'Programs'. The main header of the notebook says 'IP[y]: Notebook' and 'euler014 La...'. A menu bar includes 'File', 'Edit', 'View', 'Insert', 'Cell', 'Kernel', and 'Help'. Below the menu, there's a toolbar with icons for saving, undo, redo, and running code. The notebook contains four input cells:

```
In [1]: import numpy as np
        from pandas import Series
```

```
In [2]: def collatz_length(num):
        start = num
        leng = 0
        if num > 0:
            leng += 1
            while num > 1:
                leng += 1
                if num % 2 == 0:
                    num /= 2
                else:
                    num = 3 * num + 1
            return (leng, start)
```

```
In [3]: def euler(max_num):
        indx = Series(np.arange(max_num + 1))
        cols = indx.apply(collatz_length)
        (maxl, maxi) = cols.max()
        print "The longest Collatz sequence is for", maxi
```

```
In [4]: euler(100)
```

The output of the fourth cell is: "The longest Collatz sequence is for"

```
In [ ]:
```

Setting Up Scientific Python for Mac OSX

A lean guide for installing the scientific Python stack on a Mac.

Gabriel Perdue

This book is for sale at <http://leanpub.com/settingupscientificpythonformacosx>

This version was published on 2014-07-06



This is a [Leanpub](#) book. Leanpub empowers authors and publishers with the Lean Publishing process. [Lean Publishing](#) is the act of publishing an in-progress ebook using lightweight tools and many iterations to get reader feedback, pivot until you have the right book and build traction once you do.

©2014 Gabriel Perdue

To my wife and daughter.

Contents

Installing Scientific Python	1
--	---

Installing Scientific Python

First, if you just want to get going as fast as possible, I recommend [Anaconda](https://store.continuum.io/cshop/anaconda/)¹. It unpacks as a single directory that has everything you need, and it doesn't interfere with anything on your system. All you have to do is prepend it to your path and you can be off and running. They have a free model and some add-ons you can pay for. The sheer number of packages they include is quite impressive and they also offer an update system that keeps everything coherent. This is how I set up computers quickly at work where I just need something quickly.

If, however, you want to install all the packages and do the configuration yourself (as I prefer to on my personal computers, so I better understand what I'm using), it can be a bit of work. It is a tough call between Anaconda, which really is great, and setting things up on your own. I wanted to set things up for myself just to be sure I had a good fallback in case Anaconda becomes a problem somehow and to understand what is under the hood a bit better.

I found this guide from [Lowin Data Company](http://www.lowindata.com/2013/installing-scientific-python-on-mac-os-x/)² to be decisive in getting everything done. There were a couple of extra snags I had to work out independently, but basically, this guide is based on their recommendations.

Lowin recommends using `virtualenv`, but I don't because I like `$PATH` stability. Most likely I just misunderstand how cool `virtualenv` is - I'm sure I'll come around someday. They also recommend [Homebrew](http://brew.sh)³ and this I agree with and highly endorse. You can, of course, use a different package manager or build from source, but I've tried [Fink](http://www.finkproject.org)⁴, [MacPorts](http://www.macports.org)⁵, and [Homebrew](http://brew.sh)⁶ and found [Homebrew](http://brew.sh)⁷ to be the simplest and easiest to use.

The first "real" thing to do is go get a copy of the latest version of Python. Actually I guess the first real thing is getting the Xcode command line tools, but if you're reading this, I assume you have them. If not, and you're not sure what they are, etc., a quick internet search will explain everything.

At any rate, I tried this with both Python 2.7.6 and Python 3.3.3 and everything essentially works exactly the same. The system Python on my Mac (Mountain Lion) is 2.7.2 and I just leave it alone through all of this. I simply downloaded the `.dmg` file from the [Python downloads](https://www.python.org/downloads/)⁸ and double-clicked and that was that.

Note: if you are using Python 2.x, you can just type `python` or `ipython` when following along with the instructions below. If, on the other hand, you are using Python 3.x, you probably need to say

¹<https://store.continuum.io/cshop/anaconda/>

²<http://www.lowindata.com/2013/installing-scientific-python-on-mac-os-x/>

³<http://brew.sh>

⁴<http://www.finkproject.org>

⁵<http://www.macports.org>

⁶<http://brew.sh>

⁷<http://brew.sh>

⁸<https://www.python.org/downloads/>

python3 or ipython3 instead.

[Lowin⁹](#) recommends using Homebrew for Python also, and for a while I managed my Python that way, but in general, I've found just getting the libraries directly works best because if I decide I want to clean things up, it is easier to just dump the whole Python directory and then clean up a few symlinks in `/usr/local`. Perhaps if I used `virtualenv` I would have found Homebrew for Python to be just fine.

Doing things the way I like to puts Python (2.7.6) in `/Library/Frameworks/Python.framework/Versions/2.7`. If you have other 2.7's, (aside from your system Python), they'll be there too. The installer also puts symlinks in `/usr/local/bin` which I'm less keen on because I regard `/usr/local` as Homebrew's exclusive playground. Actually, it is worse than that, because to get the bundled version of IDLE to work, you may need to also download and install a new version of [TCL¹⁰](#) (I picked up 8.5.15.1 to go with Python 3), and that goes to `/usr/local/` also. But, so far I've limited myself to just grumbling. `brew doctor` will complain about TCL no matter what, so leaving some Python symlinks is no big deal. Just remember if you want to clean things out that you need to check `/usr/local` also.

If you grab Python 3 instead of Python 2, everything is pretty much the same from here out, so grab that if you prefer. You can have both! (I do.)

Okay, next we need to get a bunch of stuff from Homebrew:

- `brew install apple-gcc42` # I actually don't know if this is needed. But I have it.
- `brew install gfortran`
- `brew install freetype`

We'll install some more stuff later (I actually don't know if it is order sensitive, but it is plausible that it is, and I haven't dug into the code or experimented to find out).

Later on, when we're trying to install `matplotlib`, we're going to run into a problem with `freetype`:

```
1 /usr/X11/include/ft2build.h:56:38: error:
2   freetype/config/ftheader.h: No such file or directory
```

Let's just cut that off now. This way, if you're building all of this into a script, you can just go get coffee when you start it.

Type: `sudo ln -s /usr/local/include/freetype2/ /usr/include/freetype`

This will put the include directory (via softlink) where `matplotlib` expects to find it. Someday this may be fixed deep down, but having this link won't hurt you in any case.

Next we need to install a bunch of scientific Python packages! In the examples below I am piping everything by default, but for Python 3 I used `pip3`. After each of these, I recommend firing up

⁹<http://www.lowindata.com/2013/installing-scientific-python-on-mac-os-x/>

¹⁰<http://www.activestate.com/activetcl/downloads>

Python and attempting to import the packages to make sure everything worked. Note: very rarely when installing Python packages, I find I can't import the package in the directory where I just ran `pip` or `easy_install`, but if I open a new shell, everything works there. (Sometimes of course, `pip` fails and then no amount of extra shells will save you.)

- `pip install numpy`
- `pip install scipy`
- `pip install matplotlib`
- `pip install ipython`
- `easy_install readline`

At this point, you should be able to `ipython --pylab=inline` and have IPython running. The notebook and Qt modes won't work yet though.

To get PyQt working we first need Qt. I went to:

1 <https://qt-project.org/downloads>

and got 4.8.5. Do not get Qt 5! IPython appears to work only with Qt 4.x. I tried Qt 5.x first and it didn't work. Later, I found this post:

1 <http://infamousheelfilcher.blogspot.com/2013/05/notes-on-installing-ipython.html>

Installing Qt is very easy though. Download a .dmg file and double-click.

After getting Qt installed, we need SIP. I went to:

1 <http://www.riverbankcomputing.com/software/sip/download>

and pulled down source. Untar it and do the usual install: `$ python configure.py $ make $ make install`

Then, we get PyQt4.

1 <http://www.riverbankcomputing.com/software/pyqt/download>

Download the source, untar it, and do the usual song and dance. Well, it is almost the usual song and dance - you will probably need to tell Python where the `qmake` file is for your Qt distribution. Mine ended up in `/usr/bin`: `$ python configure.py -q /usr/bin/qmake $ make $ make install`

After finishing up with QT, we'll also brew some new packages:

- `brew install pyqt`
- `brew install zmq`

Then we need to get some Python packages:

- `pip install pyzmq`
- `pip install pygments`
- `pip install jinja2`

At that point the notebook and the Qt console should both work:

- `ipython qtconsole --pylab=inline`
- `ipython notebook --pylab=inline`