

Taming Pythons with pyenv

Matt Behrens · <https://www.zigg.com/> · @zigg

Software Developer & Consultant, Atomic Object · Ann Arbor, Michigan · <https://atomicobject.com/>

PyOhio · August 1, 2015

brewdo · sandboxing for Homebrew
<https://www.zigg.com/code/brewdo/>

Installing things system-wide
can break things system-wide

Problems

- System Pythons get updated out from underneath you
- Homebrew Pythons get updated out from underneath you
- Virtualenvs break when Pythons get updated
- You need access to Pythons that aren't readily available
- Activating and deactivating is tedious and error-prone
- You can't give all this to a non-Pythonista to use on their MacBook

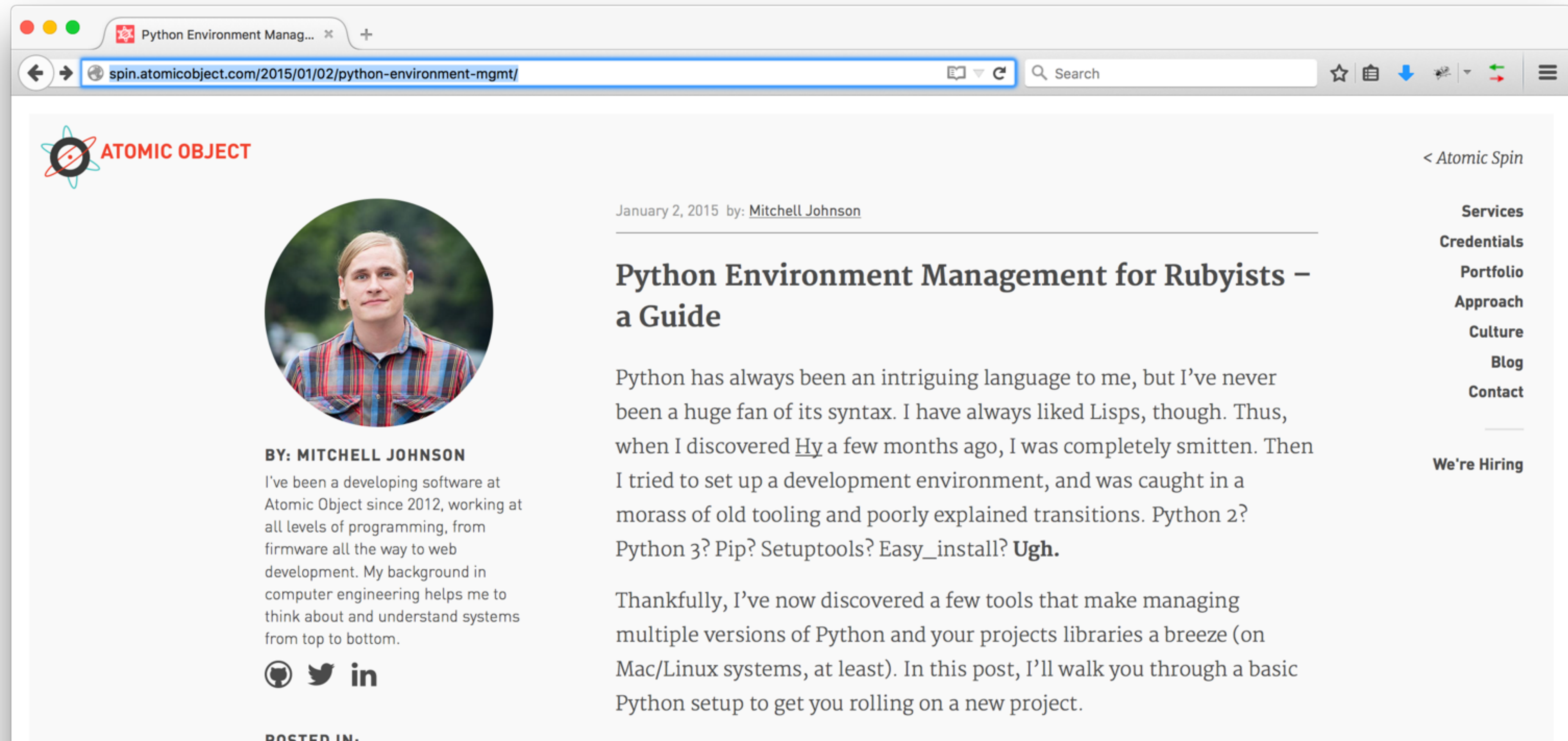
We need to tame the pythons



(tamed python courtesy [@quephird](#))

Enlightenment

<http://spin.atomicobject.com/2015/01/02/python-environment-mgmt/>



The screenshot shows a web browser window with the URL `spin.atomicobject.com/2015/01/02/python-environment-mgmt/`. The page features the Atomic Object logo in the top left, a navigation menu on the right, and a blog post by Mitchell Johnson. The post title is "Python Environment Management for Rubyists – a Guide". The text of the post discusses the author's experience with Python and the challenges of managing development environments, mentioning tools like Hy, Pip, Setuptools, and Easy_install.

ATOMIC OBJECT

BY: MITCHELL JOHNSON

I've been a developing software at Atomic Object since 2012, working at all levels of programming, from firmware all the way to web development. My background in computer engineering helps me to think about and understand systems from top to bottom.

[GitHub](#) [Twitter](#) [LinkedIn](#)

POSTED IN:

January 2, 2015 by: [Mitchell Johnson](#)

Python Environment Management for Rubyists – a Guide

Python has always been an intriguing language to me, but I've never been a huge fan of its syntax. I have always liked Lisps, though. Thus, when I discovered [Hy](#) a few months ago, I was completely smitten. Then I tried to set up a development environment, and was caught in a morass of old tooling and poorly explained transitions. Python 2? Python 3? Pip? Setuptools? Easy_install? **Ugh.**

Thankfully, I've now discovered a few tools that make managing multiple versions of Python and your projects libraries a breeze (on Mac/Linux systems, at least). In this post, I'll walk you through a basic Python setup to get you rolling on a new project.

[< Atomic Spin](#)

- [Services](#)
- [Credentials](#)
- [Portfolio](#)
- [Approach](#)
- [Culture](#)
- [Blog](#)
- [Contact](#)

[We're Hiring](#)

Taming your Pythons with pyenv

- Knows how to build lots of Pythons
- Keeps track of those built Pythons
- pyenv's Pythons are stable
- Can integrate with virtualenv
- Frictionless environment switching—just `cd`

Installation

- `brew install pyenv pyenv-virtualenv`
- Add shims to `.profile`
 - `if which pyenv > /dev/null; then eval "$(pyenv init -)"; fi`
 - `if which pyenv-virtualenv-init > /dev/null; then eval "$(pyenv virtualenv-init -)"; fi`
- Or follow instructions at <https://github.com/yyuu/pyenv>

Usage

- `pyenv install --list`
- `CFLAGS="-I$(brew --prefix openssl)/include" LDFLAGS="-L$(brew --prefix openssl)/lib" pyenv install -v 3.5.0b3`
- `mkdir ~/src/project && cd ~/src/project`
- `pyenv virtualenv 3.5.0b3 project`
- `pyenv local project`

