
Welcome to Python Boot Camp!
Fall
2015!

Objectives

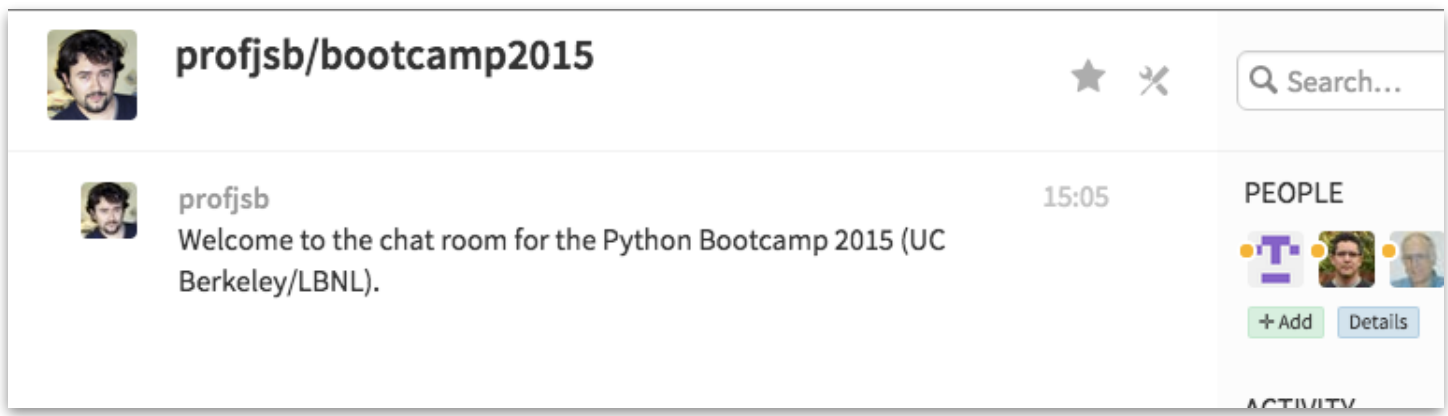
- Introduce you to the Python language
- Get you writing Python code. Build Python hacking muscle memory
- Convince you of its utility in your research life
- Instill good coding and curation practices
- We try not to proselytize, but sometimes it's too hard to resist

Organization

- 3 days of modules (~1 hour) lectures + demos
  http://www.pythonbootcamp.info/schedule
- Breakout coding sessions (supervised) after each module
- Caffeine provided, Quick lunch on your own
- Homework (small code projects)
- Blood, sweat, tears → a more productive you

Connecting

#pyboot



https://gitter.im/profjsb/bootcamp2015

Shout outs

BIDS

Ali Ferguson, Kevin Koy

LBL

Dani Ushizima, Sophia Pasadis

BIGDATA Grant

Something you should know about us...

@profjsb, @fperez_org

Introduction

- What is Python?
- Why Python?
- Getting Started...

What is Python?

Python is an interpreted, object-oriented, high-level programming language with dynamic semantics. Its high-level built in data structures, combined with dynamic typing and dynamic binding, make it very attractive for Rapid Application Development, as well as for use as a scripting or glue language to connect existing components together. Python's simple, easy to learn syntax emphasizes readability and therefore reduces the cost of program maintenance. Python supports modules and packages, which encourages program modularity and code reuse. The Python interpreter and the extensive standard library are available in source or binary form without charge for all major platforms, and can be freely distributed.

http://www.python.org/doc/essays/blurb/

What is Python?

interpreted	no need for a compiling stage
object-oriented	programming paradigm that uses objects (complex data structures with methods)
high level	abstraction from the way machine interprets & executes
dynamic semantics	can change meaning on-the-fly
built in	core language (not external)
data structures	ways of storing/manipulating data
script/glue	programs that control other programs
typing	the sort of variable (int, string)
syntax	grammar which defines the language
library	reusable collection of code
binary	a file that you can run/execute

Development History

- started over the Christmas break 1989, by Guido van Rossum (now at Dropbox)
- developed in the early 1990s
- name comes from **Monty Python's Flying Circus**
- Guido is the Benevolent Dictator for Life (BDFL), meaning that he continues to oversee Python's development.

Development History

- Open-sourced development from the start (BSD licensed now)
  http://www.opensource.org/licenses/bsd-license.php
- Relies on large community input (bugs, patches) and 3rd party add-on software
- Version 2.0 (2000), 2.6 (2008), 2.7 (2010).
- Version 3.X (2008) is not backward compatible with 1.X & 2.X. But 2.7 code is "easily" migrated to 3.X
- We're using **3.4.3** in this class

Why Python?

Some of the Alternatives

C, C++, FORTRAN

Pros: great performance, backbone of legacy scientific computing codes

Cons: syntax not optimized for causal programming, no interactive facilities, difficult visualization, text processing, etc.

Mathematica, Maple, Matlab, IDL

Pros: interactive, great visuals, extensive libraries

Cons: costly, proprietary, unpleasant for large-scale programs and non-mathematical tasks.

Perl: http://strombergers.com/python/

Why Python?

- Free (BSD license), highly portable (Linux, OSX, Windows, lots...)
- Interactive interpreter provided.
- Extremely readable syntax ("executable pseudo-code").
- Simple: non-professional programmers can use it effectively
  - great documentation
  - total abstraction of memory management
- Clean object-oriented model, but not mandatory.
- Rich built-in types: lists, sets, dictionaries (hash tables), strings, ...
- Very comprehensive standard library (batteries included)
- Standard libraries for IDL/Matlab-like arrays (NumPy)
- Easy to wrap existing C, C++ and FORTRAN codes.

Why Python?

Amazingly Scalable

Interactive experimentation
build small, self-contained scripts or million-lines projects.
From occasional/novice to full-time use (try that with C++).

The Kitchen Sink (in a good way)

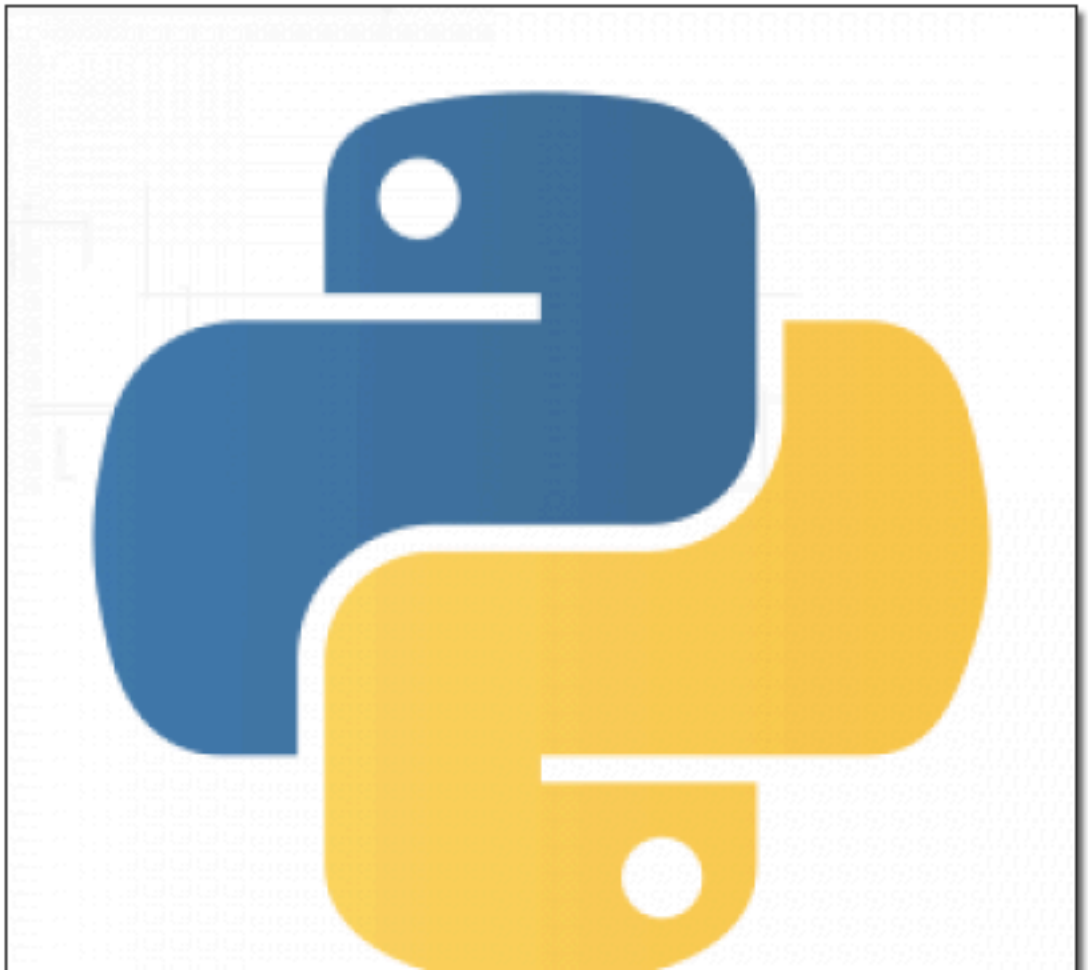
really can do anything you want, with impressive simplicity

Performance, if you need it

As an interpreted language, Python is slow.
But...if you need speed you can do the heavy lifting in C or FORTRAN
or you can use a Python compiler (e.g., Cython)

Readers' Choice Awards 2011

Dec 01, 2011 By Shawn Powers



Best Programming Language

Python

Runner-up: C++

A three-time winner in our best programming category, Python continues to dominate. Close on its heels this year, however, is C++. In fact, a scant 6% separated the two. It's quite obvious, however, that our readers don't suffer from ophidiophobia in the least —*hiss*.

My group uses it for....

Running a robotic telescope

- interfacing with legacy hardware device drivers
- talking over serial & parallel lines to telescope control hardware
- oversee functioning of all sub-systems (themselves written in Python)
- sending email and pager alerts when distressed
- writing real-time web pages (for data display, weather)
- moving image data over the network
- interacting with databases

http://pairitel.org

My group uses it for....

Data reduction & Analysis

- processing FITS images quickly
- wrapping around 3rd party software

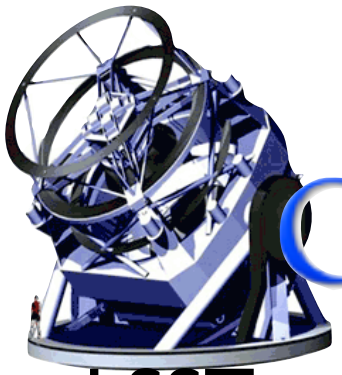
A Handy & Quick Calculator

Prototyping new algorithms/ideas

Making plots for papers

Making fast, parallel, and efficient webservices

Many Others Use it Too



LSST

Google

reddit

Honeywell

REALESTATEAGENT.com

resolver systems ltd

devis

YouTube
Broadcast Yourself

EVE ONLINE

USA
United Space Alliance

INDUSTRIA
LIGHT & MAGI

AstraZeneca

Quora

PollenationInternet

PHILIPS

Eventbrite

NASA NATIONAL AERONAUTICS AND SPACE ADMINISTRATION

Verity

GravityZoo

Dropbox

your grandmother

http://www.quora.com/Python-programming-language-1/Which-Internet-companies-use-Python

IP[y]: Notebook bootcamp fun (unsaved changes)

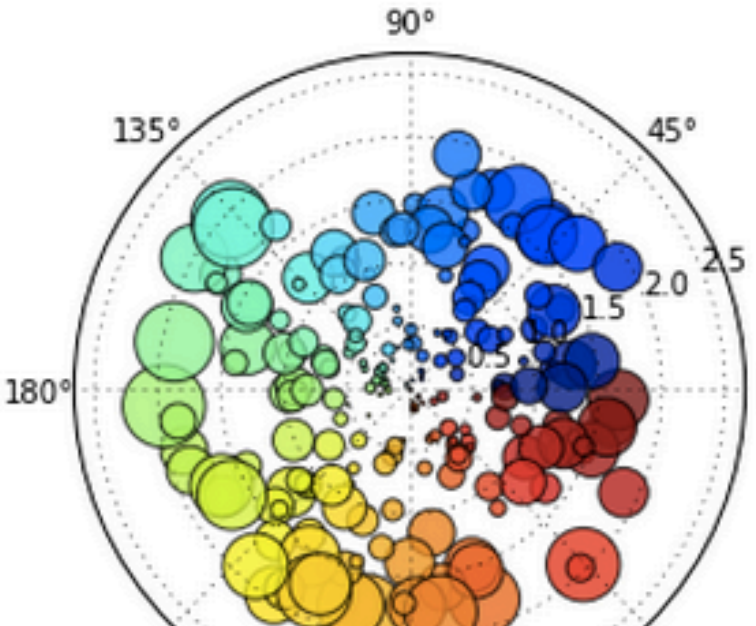
File Edit View Insert Cell Kernel Help

Code Cell Toolbar: None

In [6]:
```
%pylab inline
```

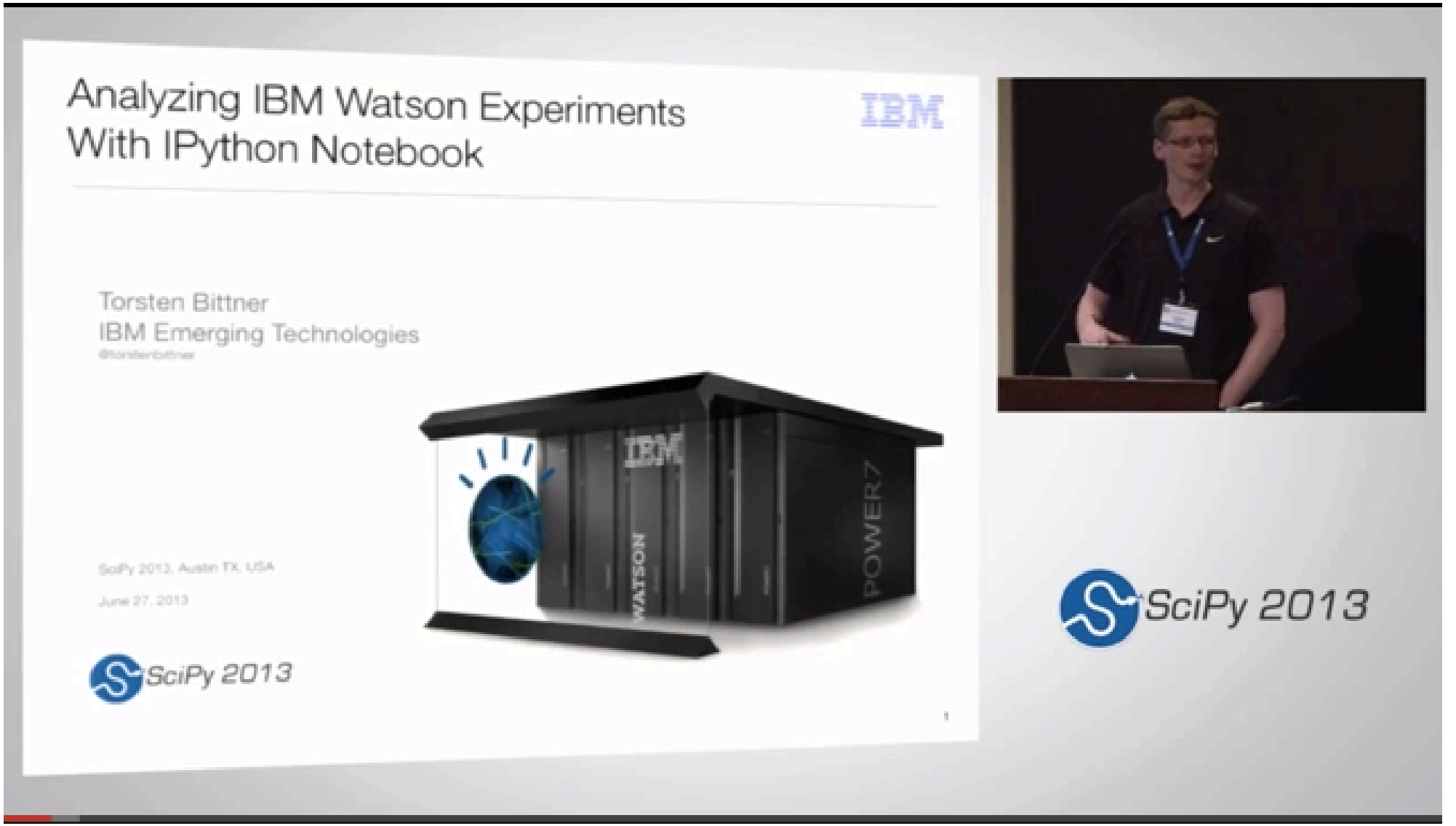
```
Populating the interactive namespace from numpy and matplotlib
```

In [7]:
```
num_bootcampers = 212
r = 2*rand(num_bootcampers)
theta = 2*pi*rand(num_bootcampers)
area = 200*r**2*rand(num_bootcampers)
ax = subplot(111,polar=True)
c = scatter(theta,r,c=theta,s=area,alpha=0.75)
```



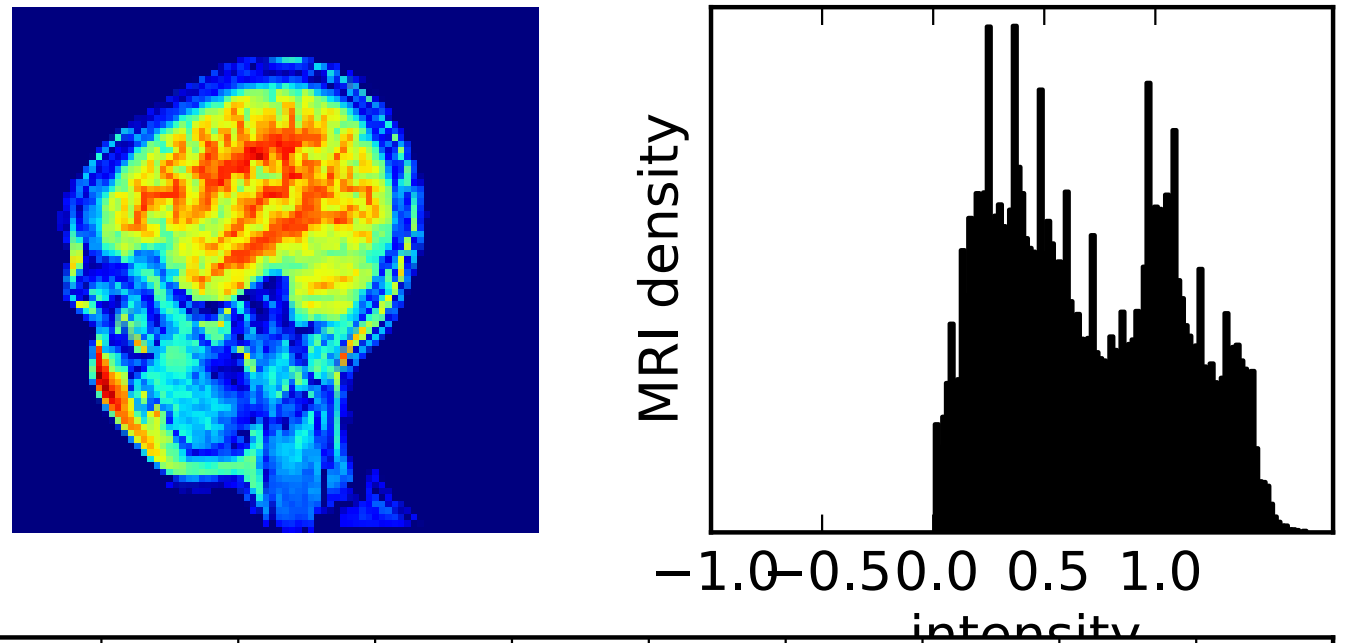
IPython/Juypter notebook:
- interactive
- versionable
- reproducible

IPython/Juypter Notebook is Gaining Major Acceptance



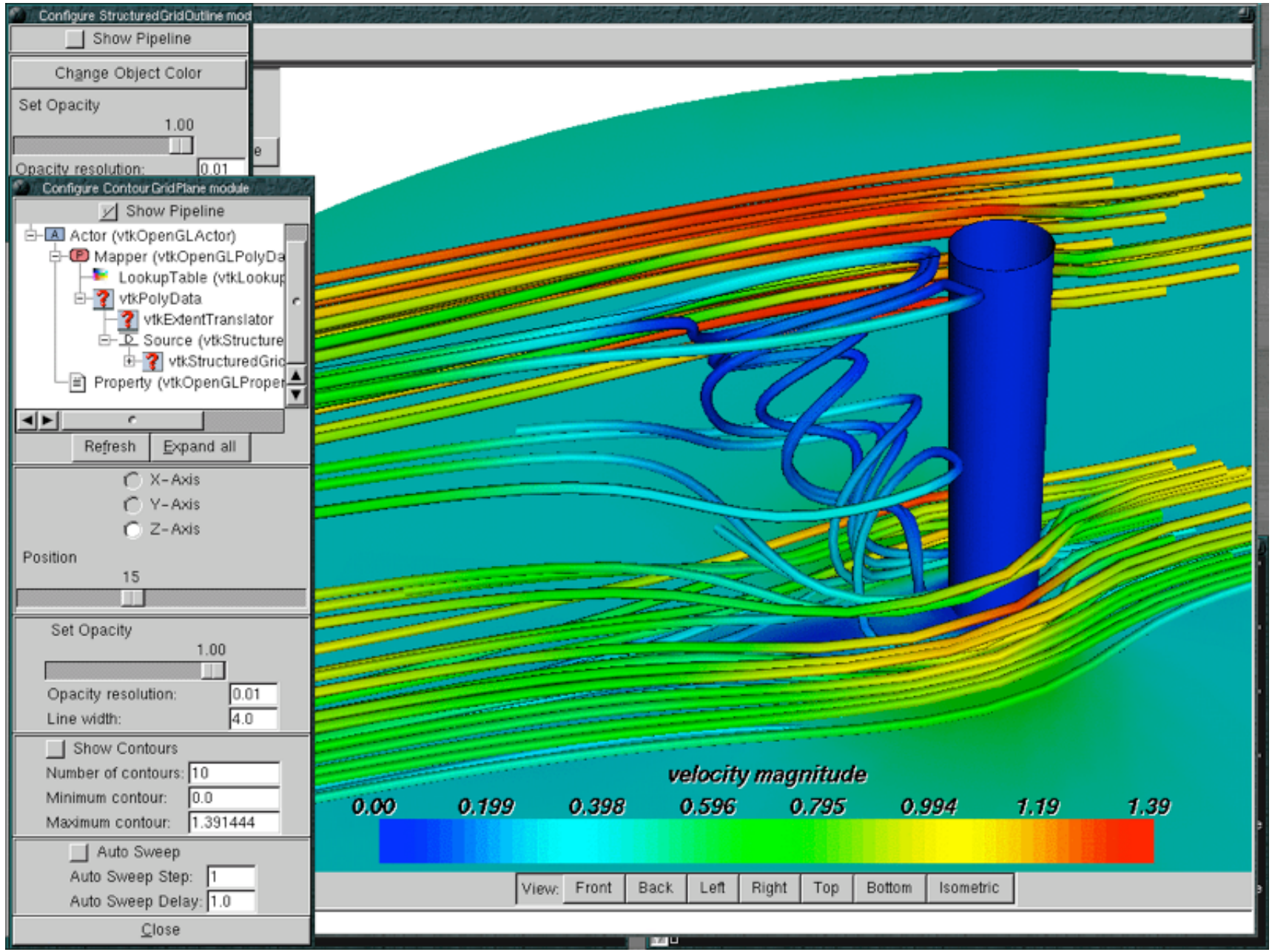
http://www.youtube.com/watch?v=tlontoyWX70

Visualization



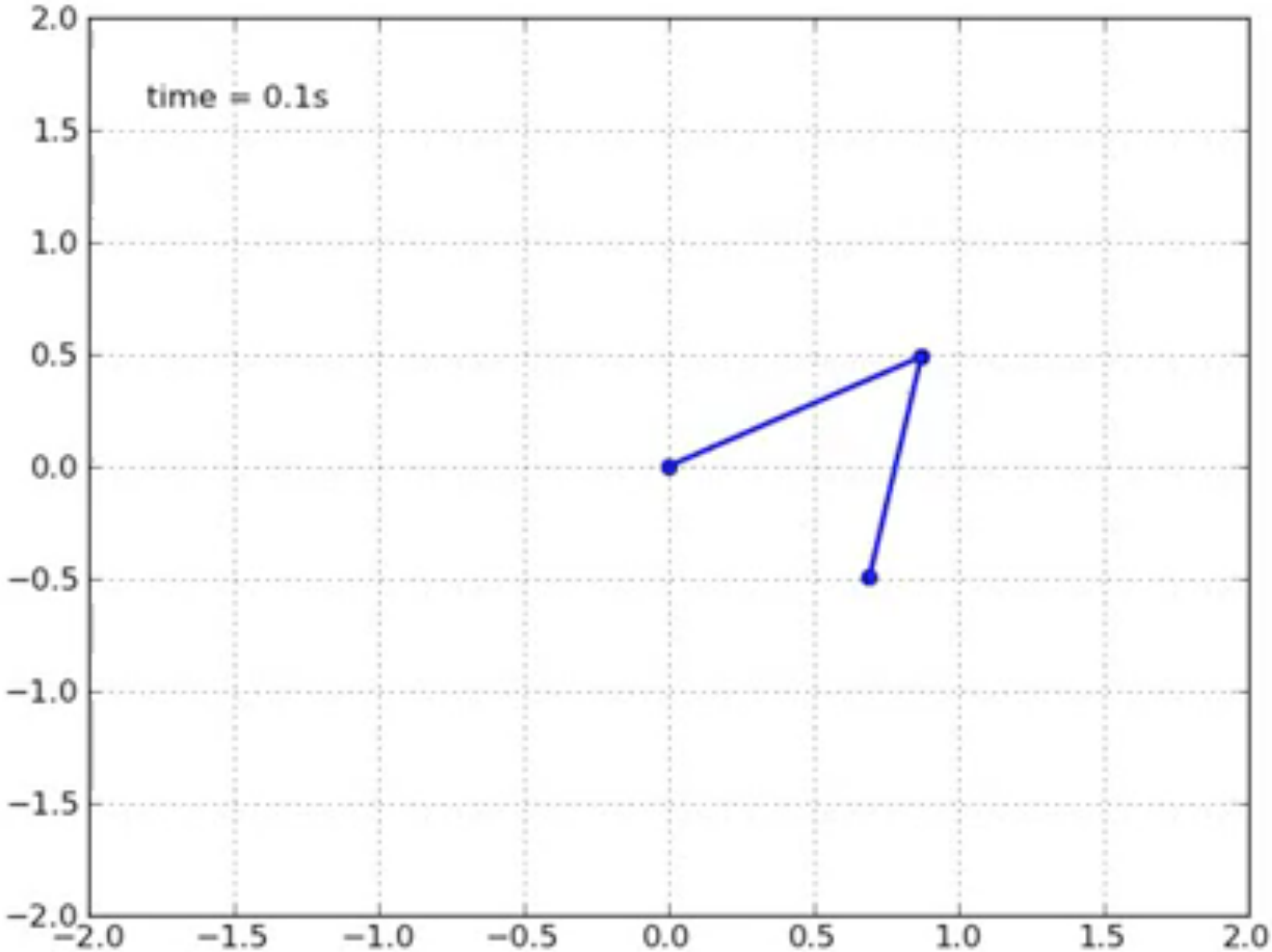
matplotlib

Visualization

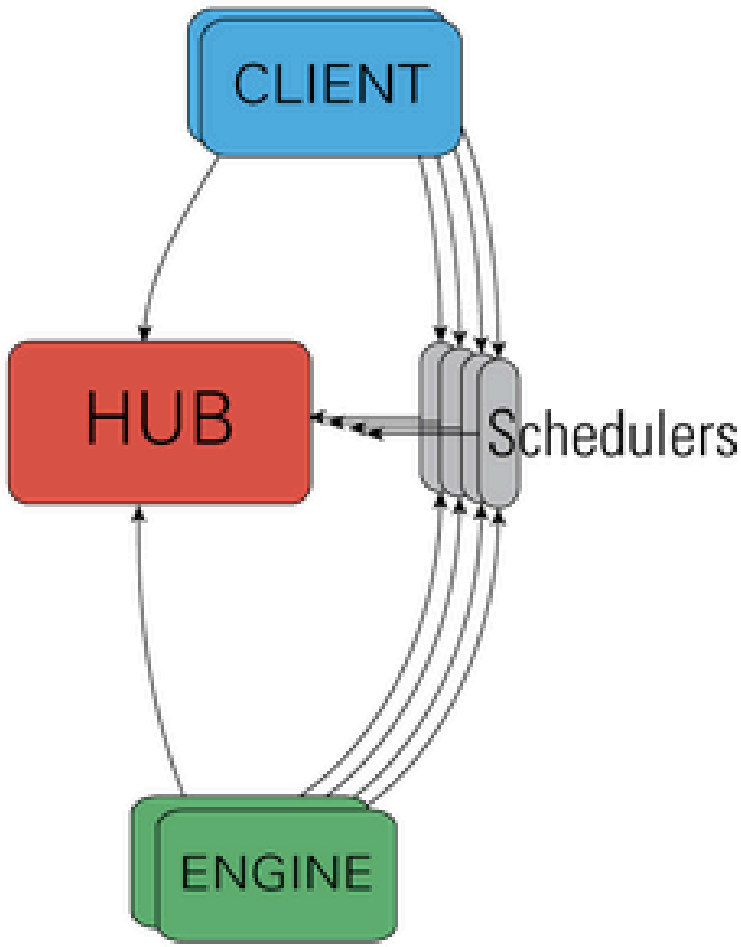


Mayavi

Animation



http://matplotlib.sourceforge.net/examples/animation/double_pendulum_animated.html



Parallelization is now very accessible
(via the IPython notebook)

Tackling Python: Let's Get Started

Asking Questions during Module presentations

- Speak up - Raise your hand

- Post questions to the chat (https://gitter.im/profjsb/bootcamp2015)

Getting Help at Any Time

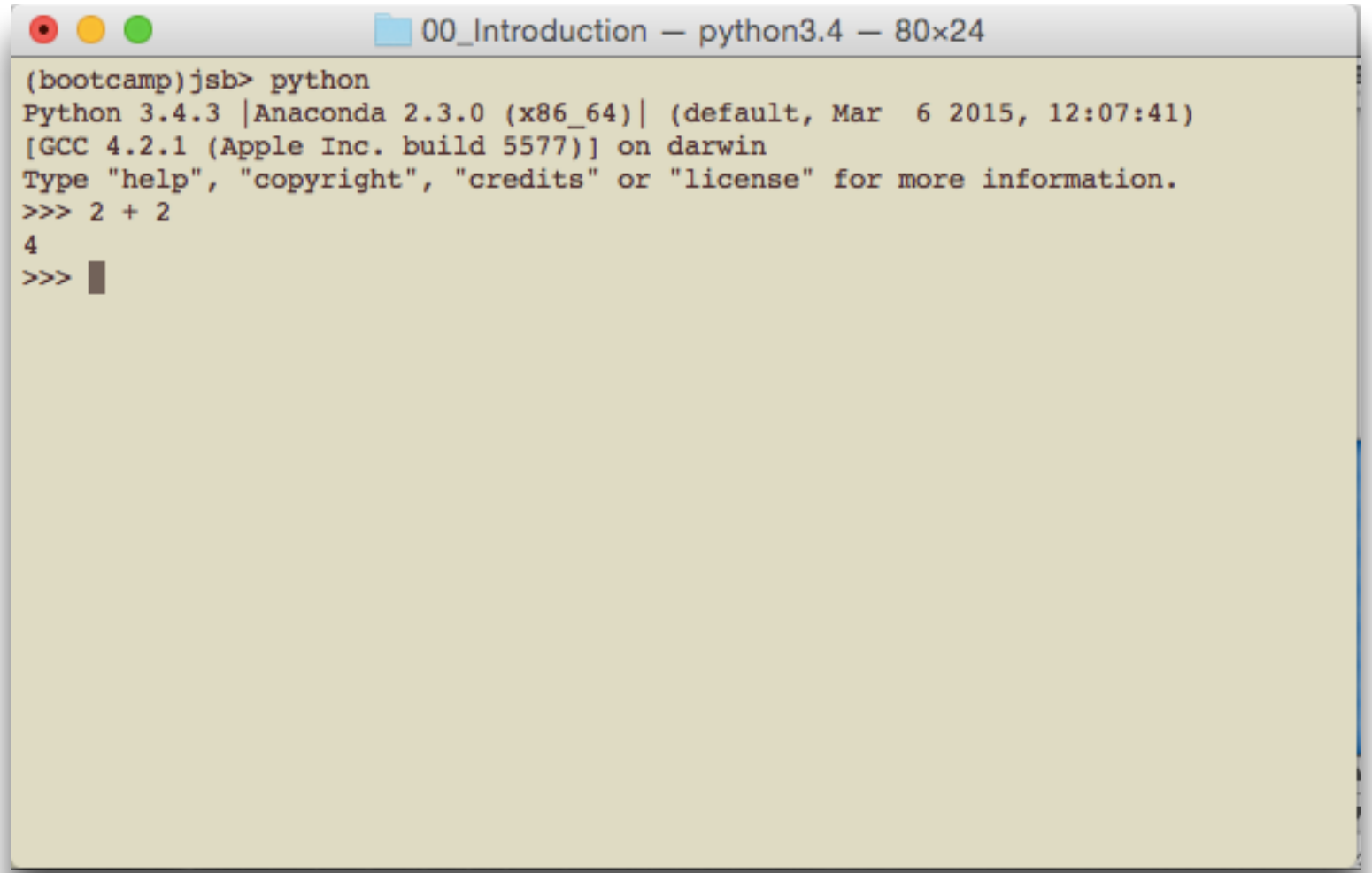
- Raise your hand and make eye contact with a counselor

- Join the chat (https://gitter.im/profjsb/bootcamp2015)

- Send email ucbpythonclass+bootcamp@gmail.com

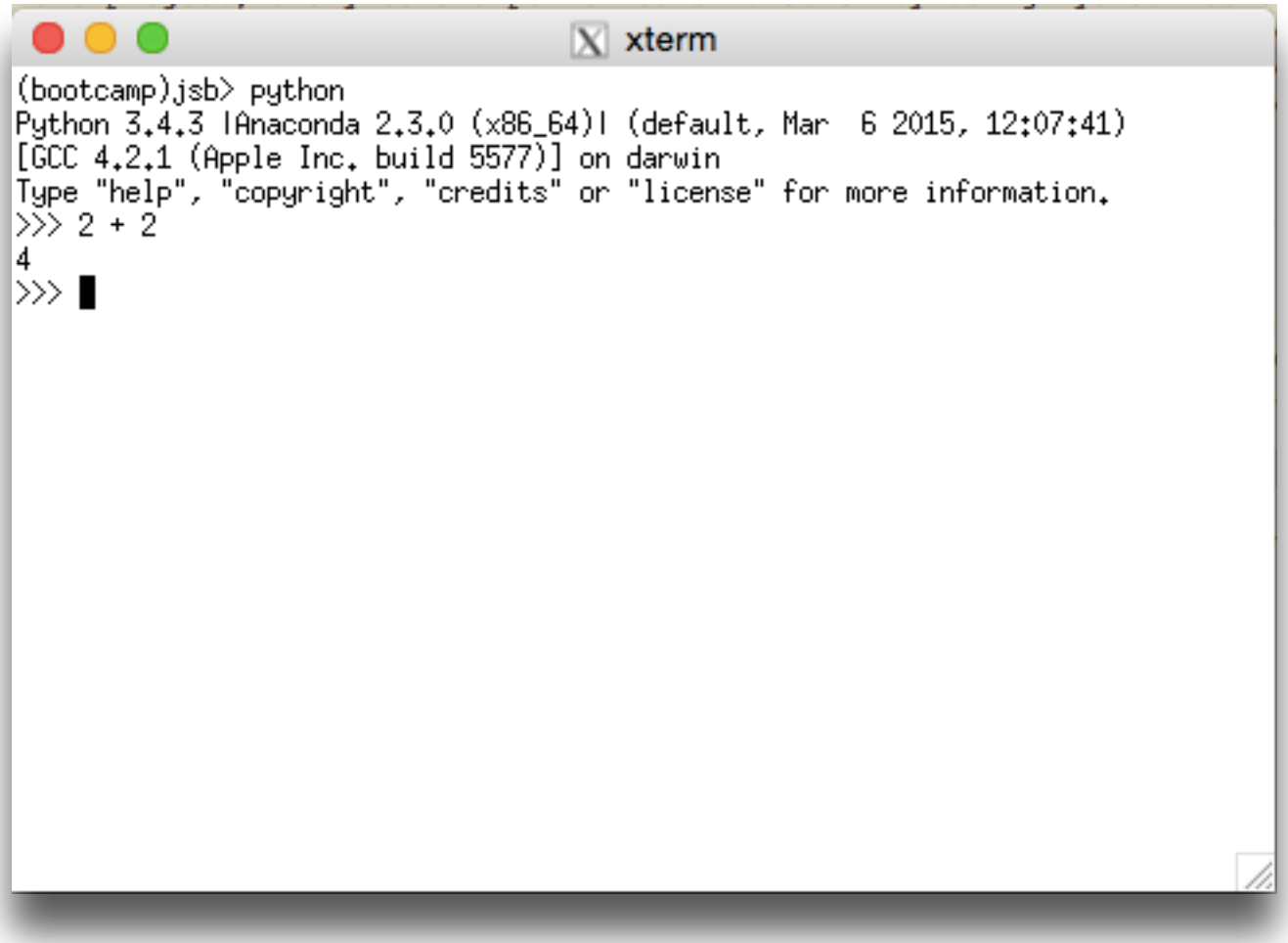
- Twitter Hashtag: #pyboot

Firing up the Interpreter



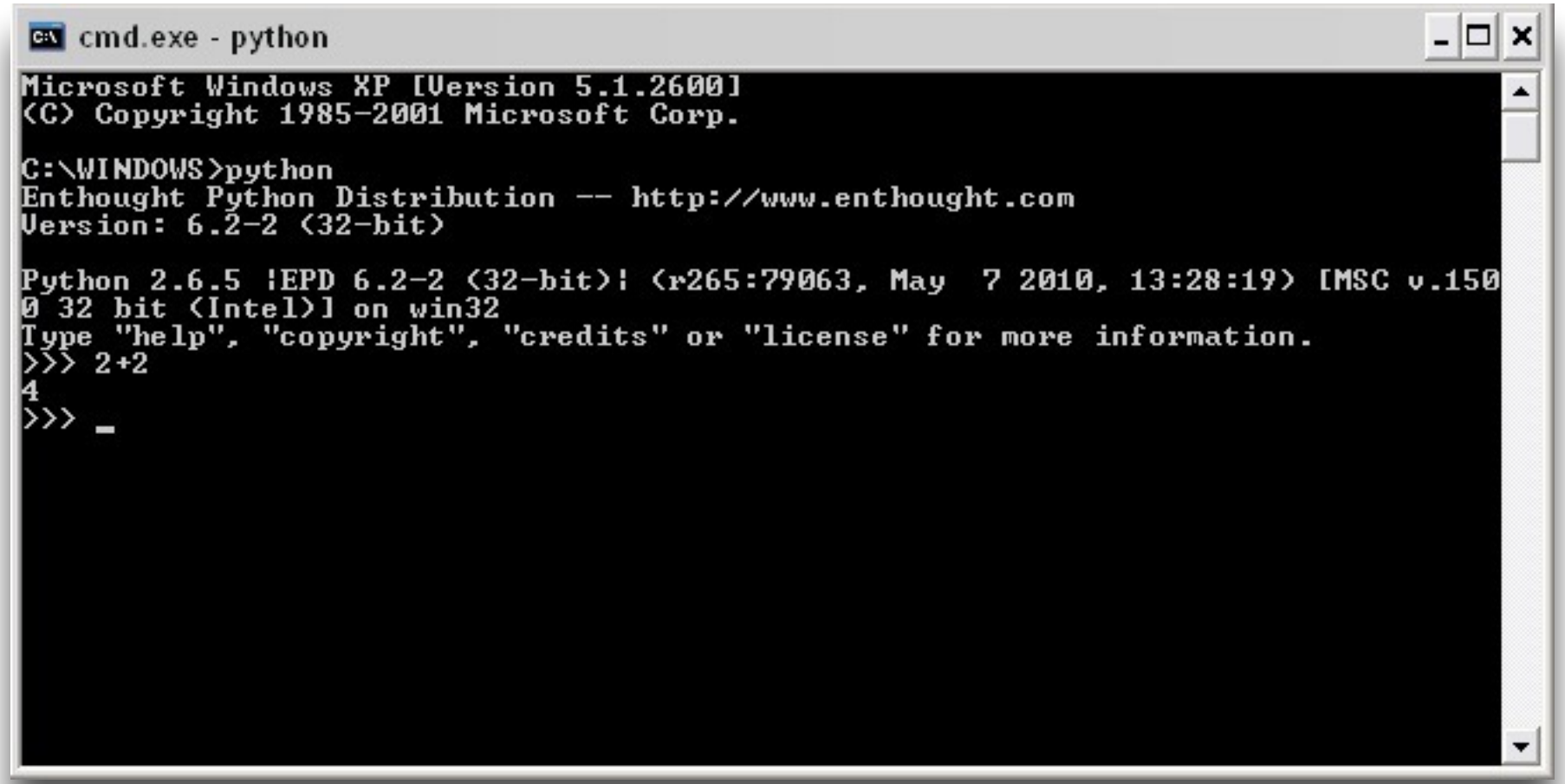
Mac OS X (Terminal)

Firing up the Interpreter



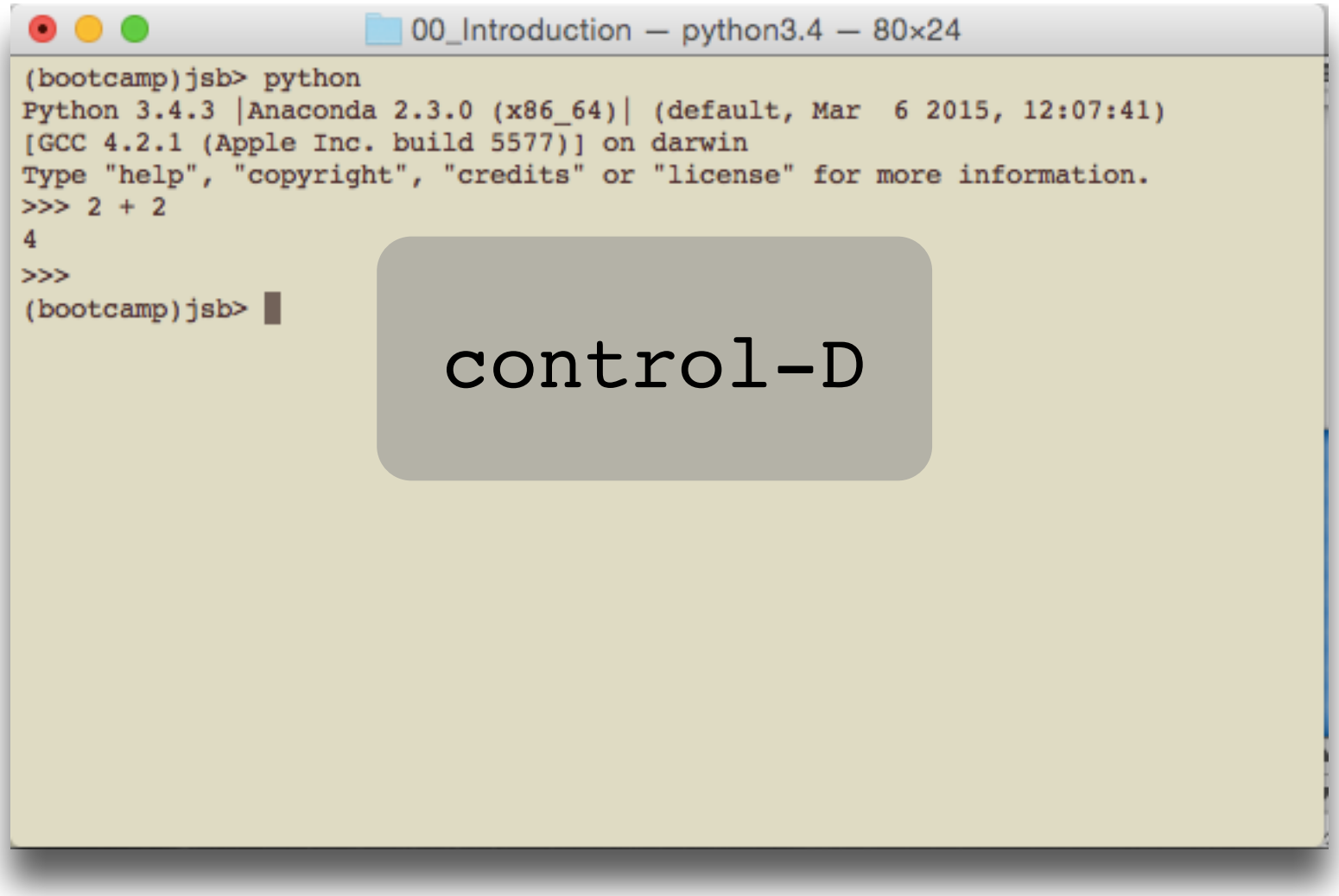
Linux/UNIX/Mac OS X (X11/Xterm)

Firing up the Interpreter



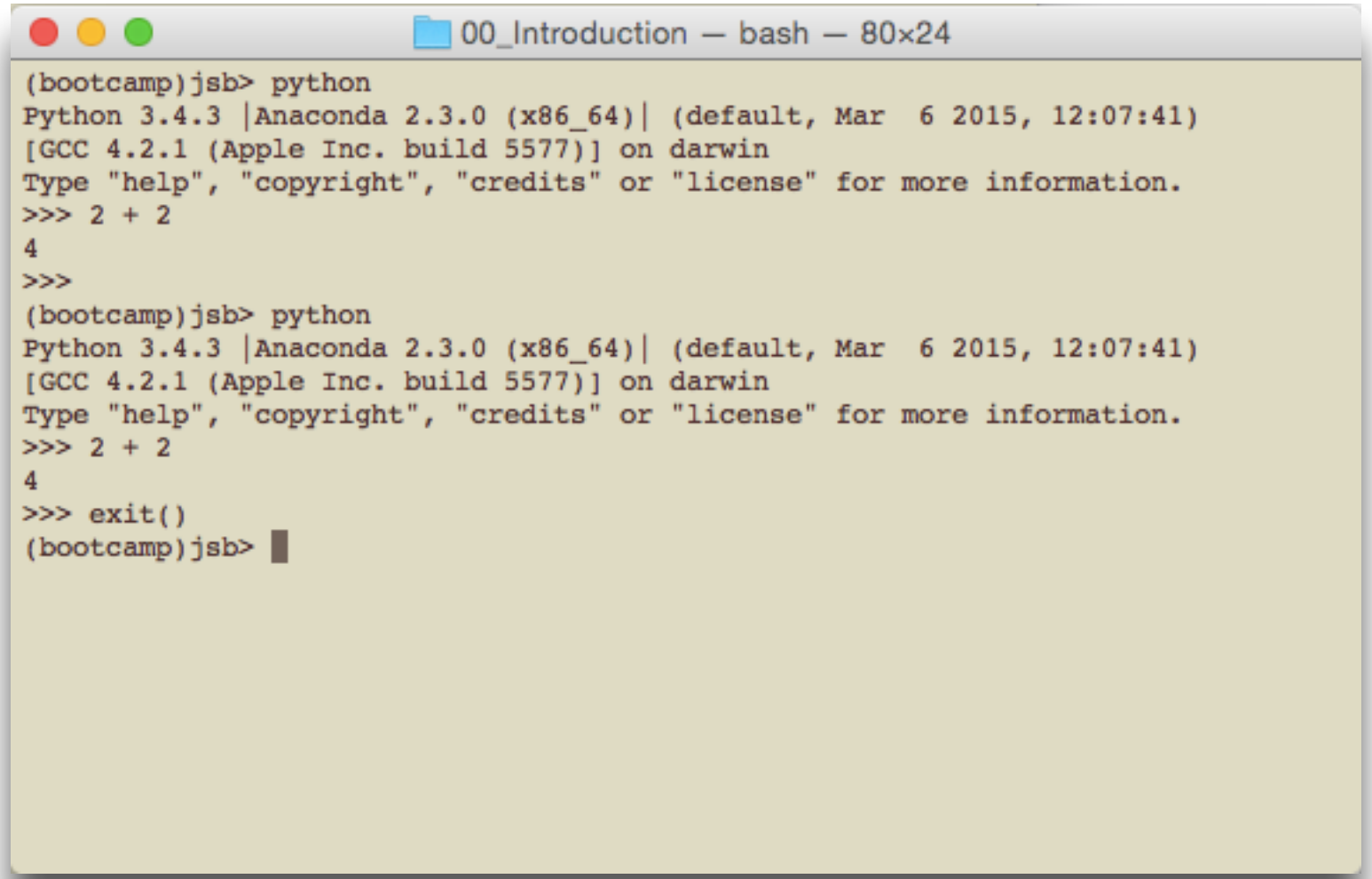
Windows

Firing up the Interpreter



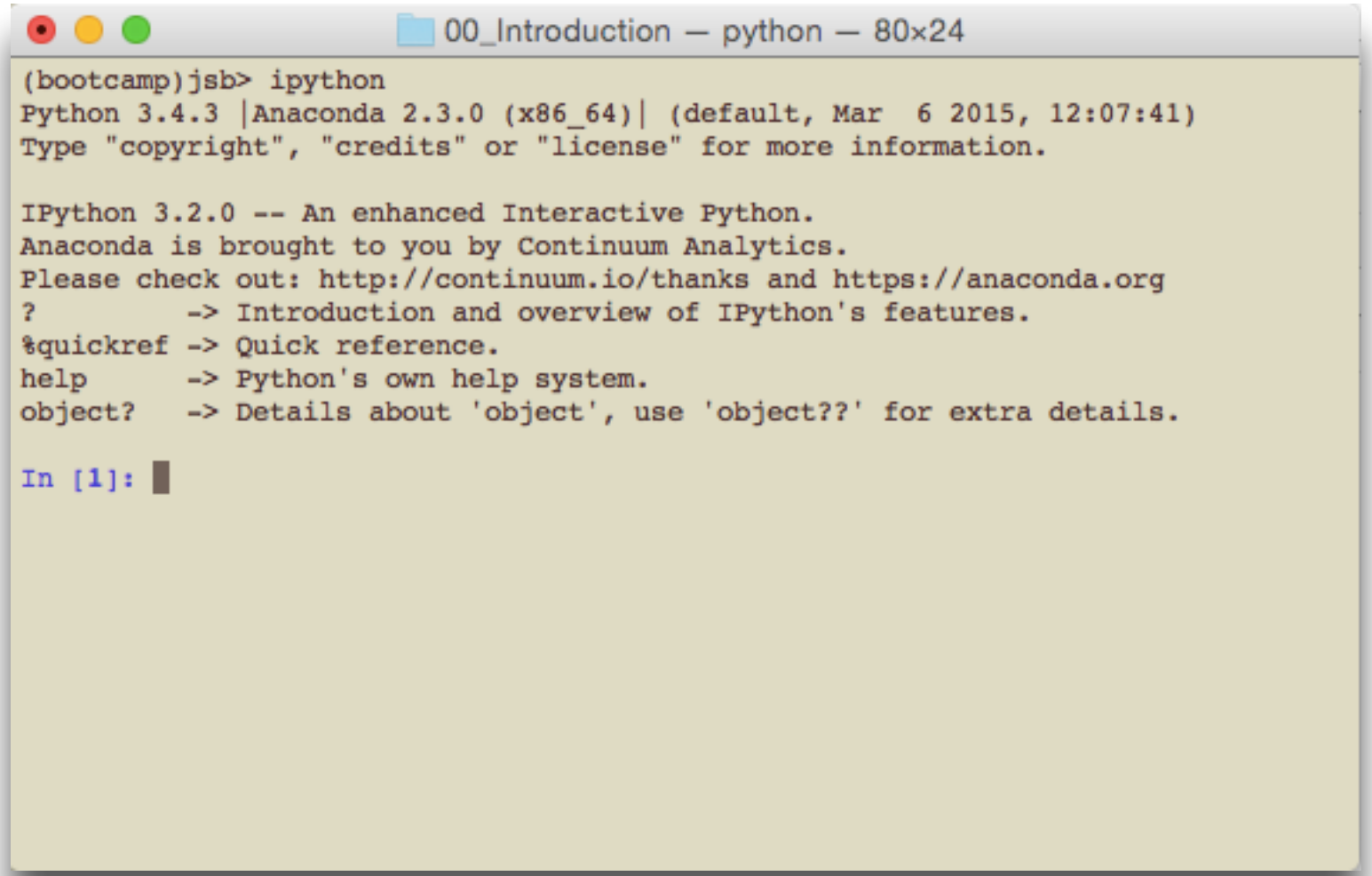
to exit: either `control-D` or `exit()`

Firing up the Interpreter



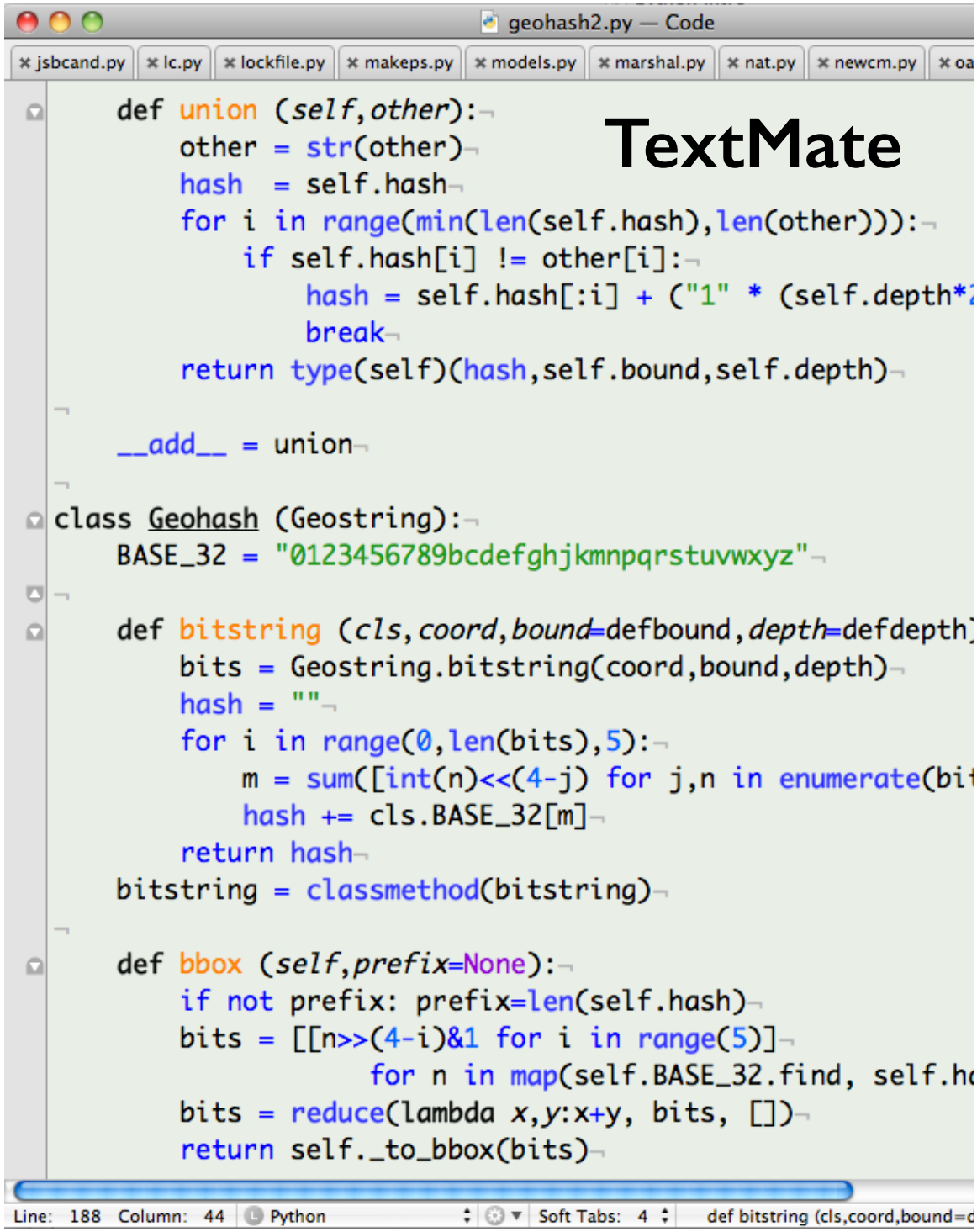
to exit: either `control-D` or `exit()`

Firing up the Interpreter



`ipython`

Editing Python Files



usually we name python files with a **.py** suffix

- snazzy GUI-based editors:

Sublime Text, TextWrangler (Mac); NotePad++, SublimeText (Windows); KWrite, Scribes, eggy (linux)

- old/powerful editors:

vim, emacs, nano, ...

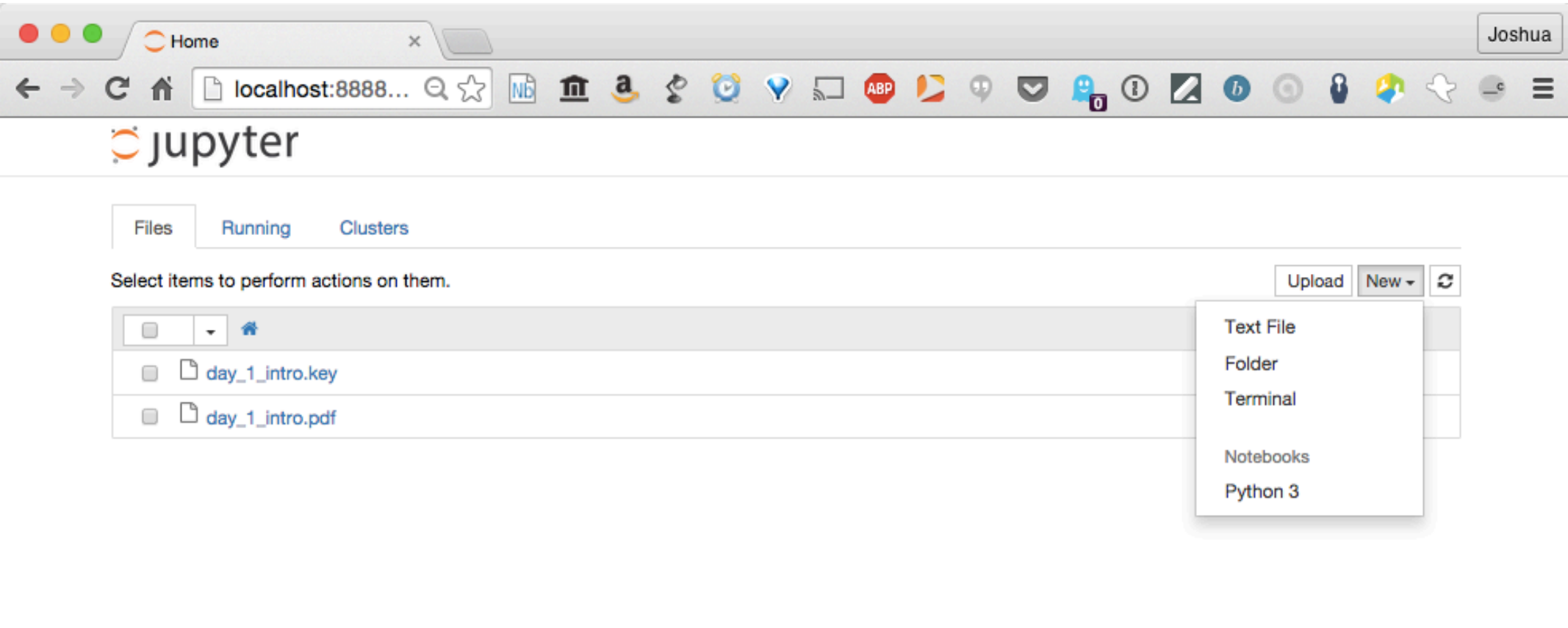
http://www.sublimetext.com/2

http://wiki.python.org/moin/PythonEditors

type: ipython notebook

```
00_Introduction — python — 137×19
(bootcamp)jsb> ipython notebook
[I 18:30:08.551 NotebookApp] Using MathJax from CDN: https://cdn.mathjax.org/mathjax/latest/MathJax.js
[I 18:30:08.631 NotebookApp] Serving notebooks from local directory: /Users/jbloom/Classes/python-bootcamp/Lectur
[I 18:30:08.631 NotebookApp] 0 active kernels
[I 18:30:08.631 NotebookApp] The IPython Notebook is running at: http://localhost:8888/
[I 18:30:08.631 NotebookApp] Use Control-C to stop this server and shut down all kernels (twice to skip confirmat
```

in your browser…



or … try.jupyter.org