



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

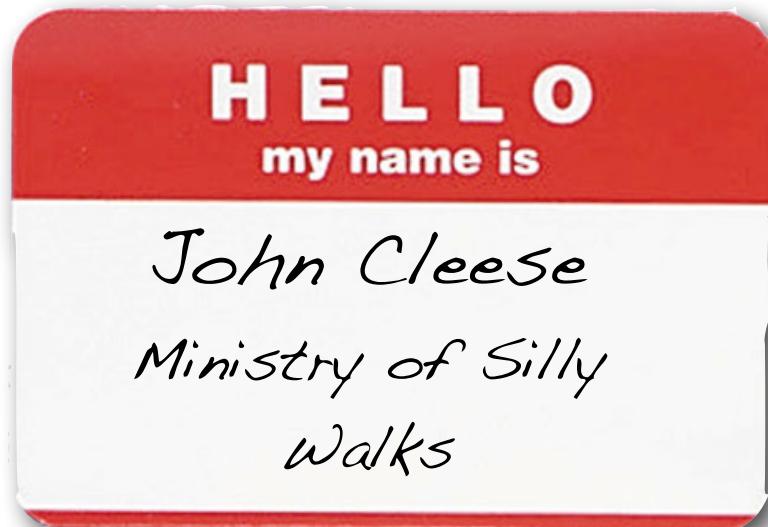
Organization

- 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



profjsb/bootcamp2015

profjsb 15:05 PEOPLE

Welcome to the chat room for the Python Bootcamp 2015 (UC Berkeley/LBNL).

+ Add Details

ACTIVITY

<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?

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

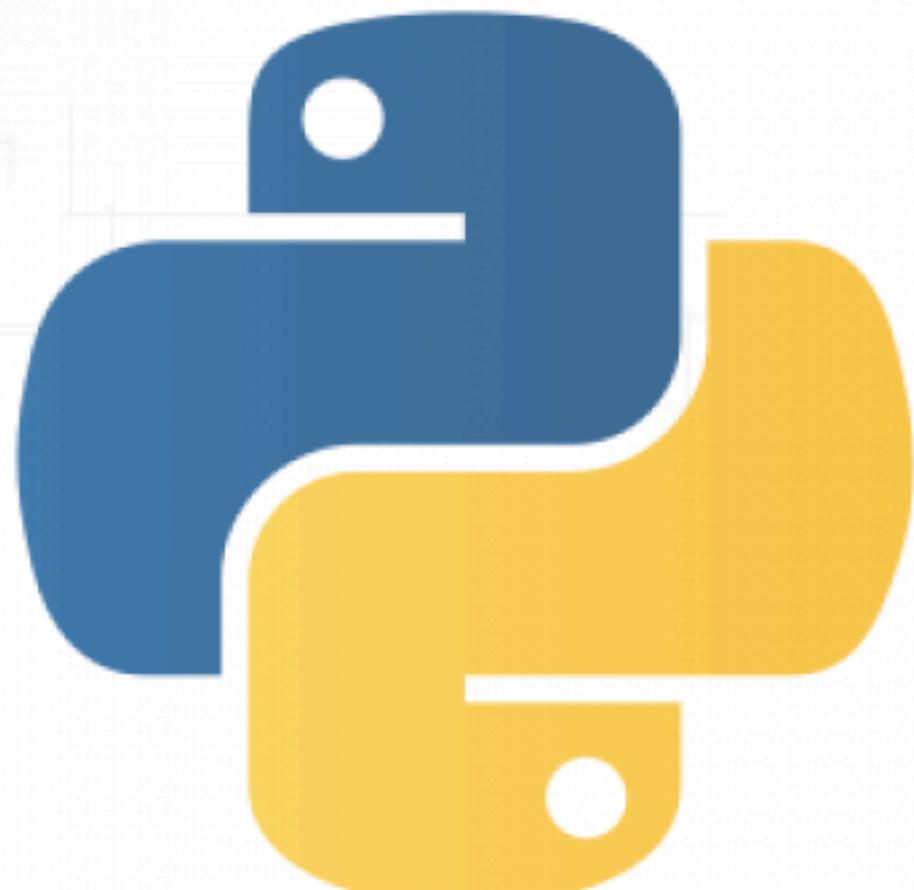
Performance, if you need it

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)

Home ›

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



Google



reddit



Honeywell



You Tube
Broadcast Yourself



USA
United Space Alliance



Quora



PHILIPS



NATIONAL AERONAUTICS AND SPACE ADMINISTRATION

Eventbrite

Verity

GravityZoo



Dropbox

your
grandmother



IP[y]: Notebook

bootcamp fun (unsaved changes)

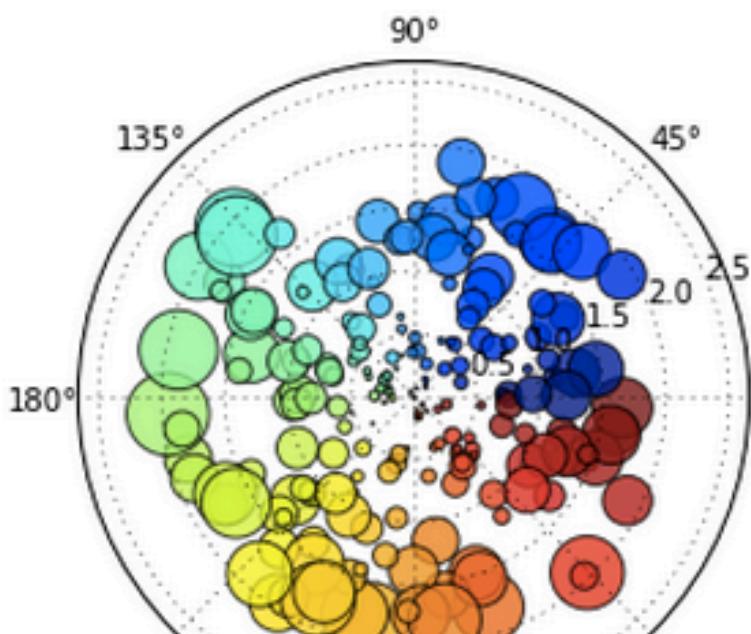
File Edit View Insert Cell Kernel Help



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/Jupyter notebook:
- interactive
- versionable
- reproducible

IPython/Jupyter Notebook is Gaining Major Acceptance

Analyzing IBM Watson Experiments
With IPython Notebook

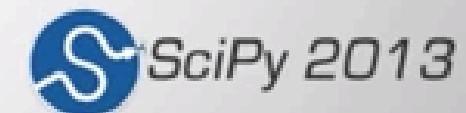


Torsten Bittner
IBM Emerging Technologies
[@torstenbittner](https://twitter.com/torstenbittner)



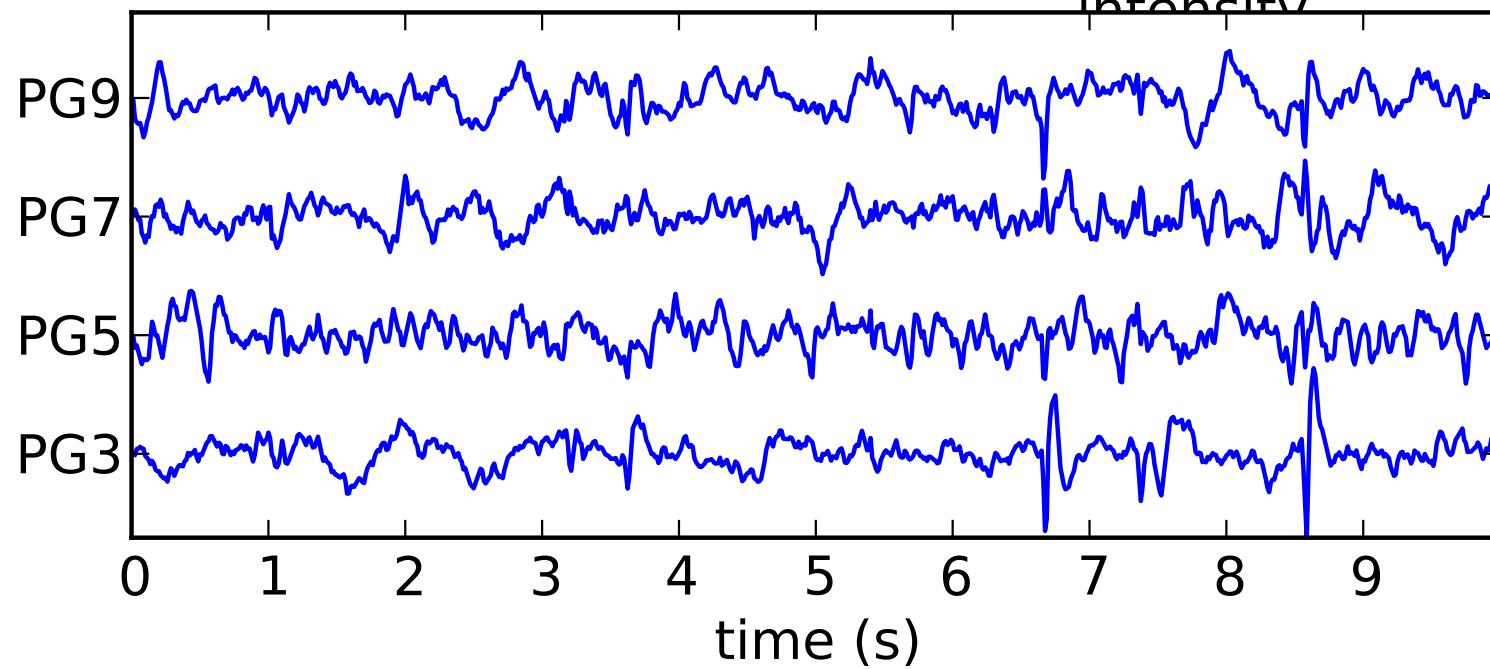
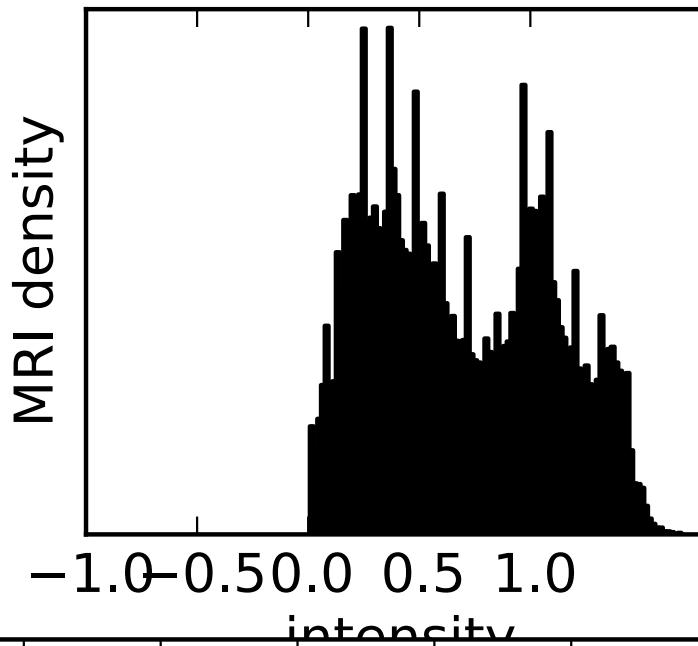
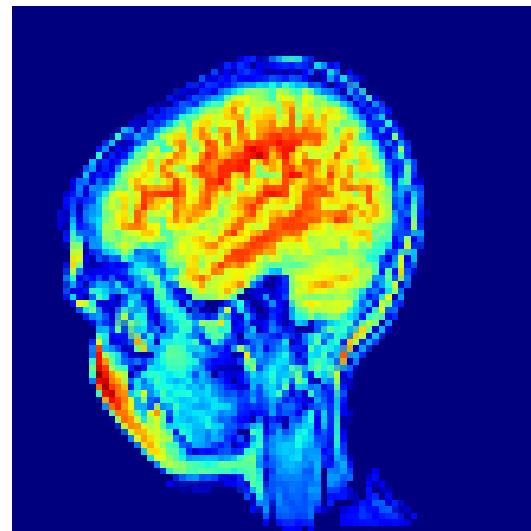
SciPy 2013, Austin TX, USA

June 27, 2013



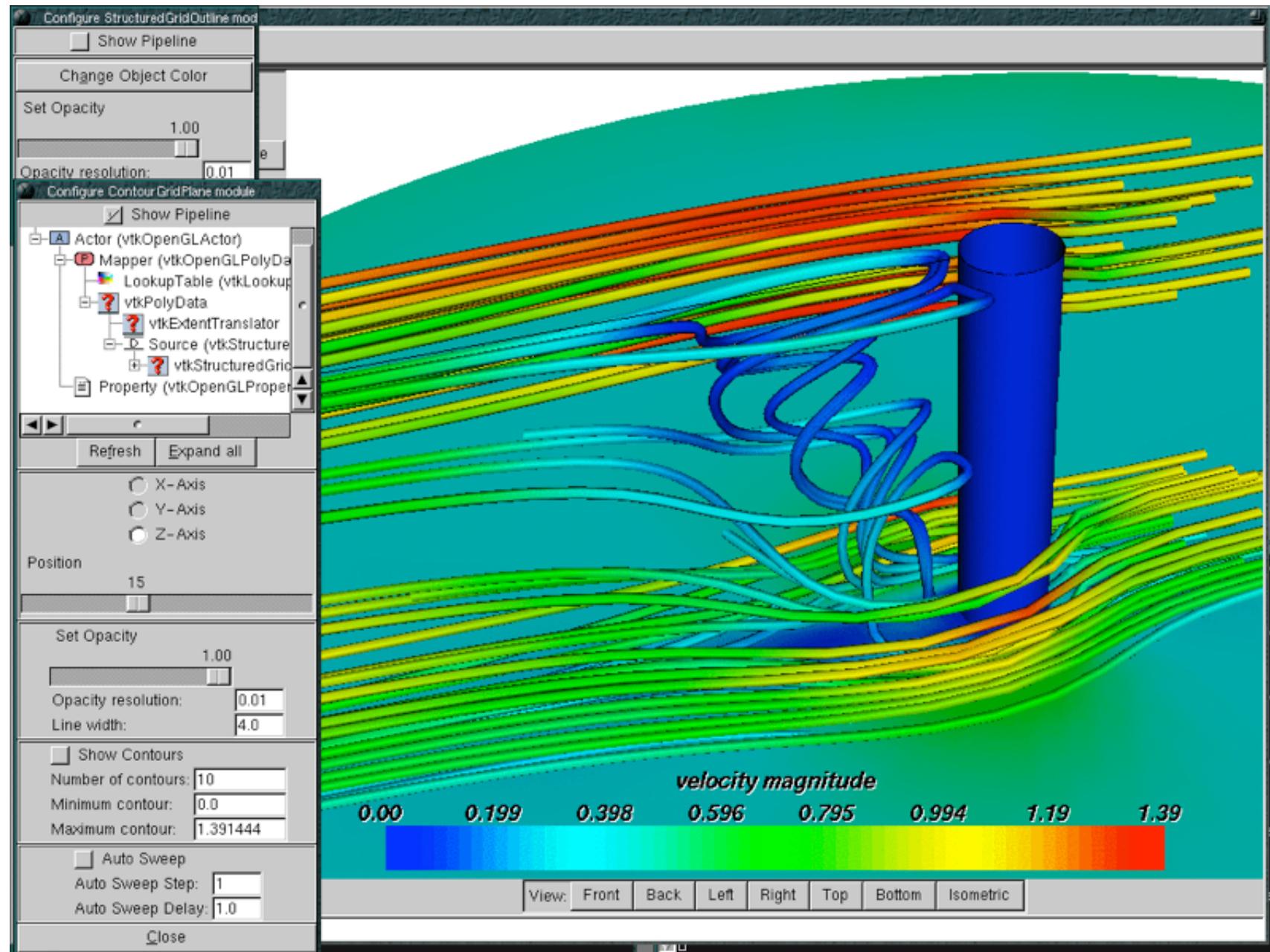
<http://www.youtube.com/watch?v=tlontoyWX70>

Visualization



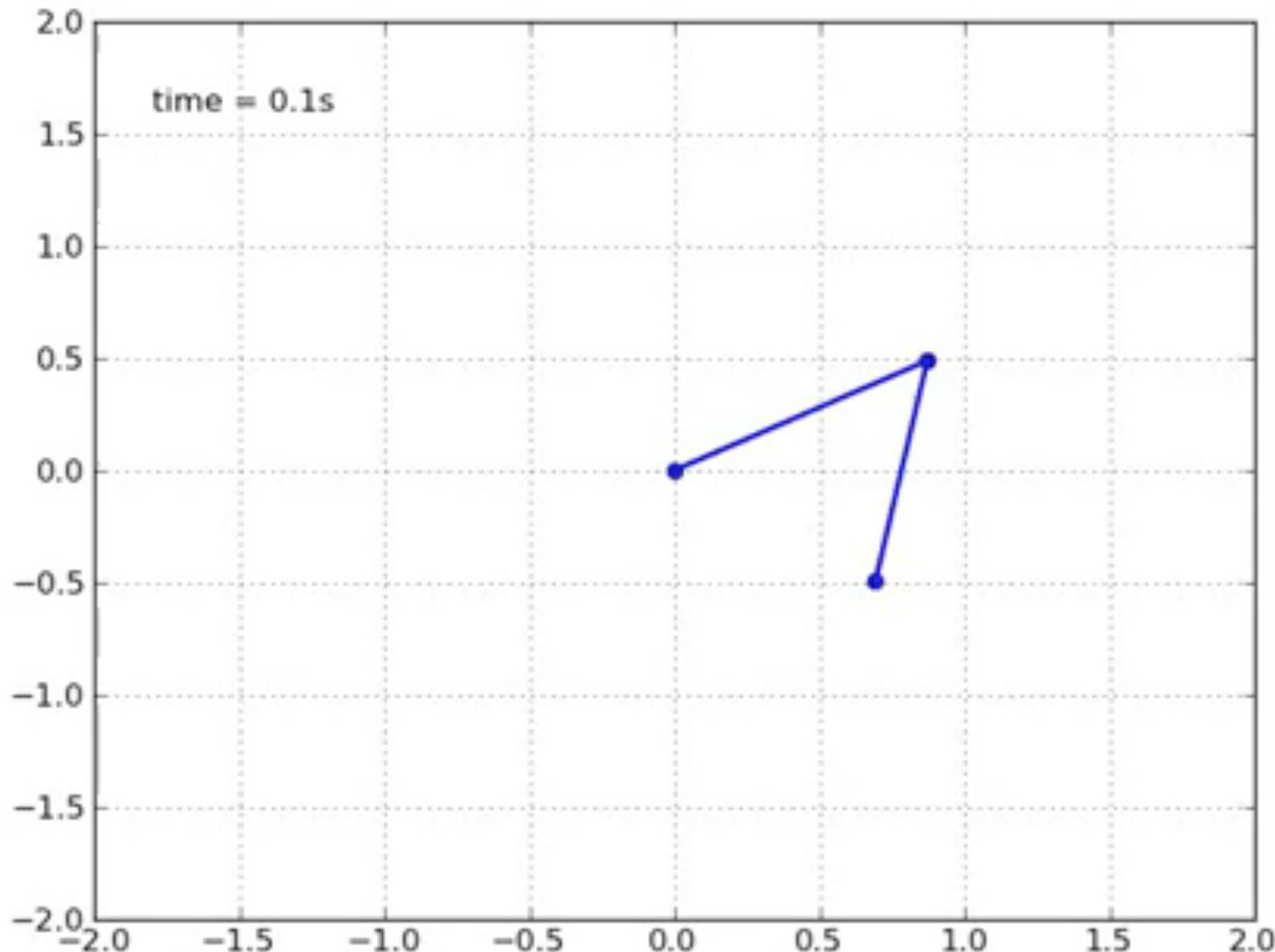
matplotlib

Visualization

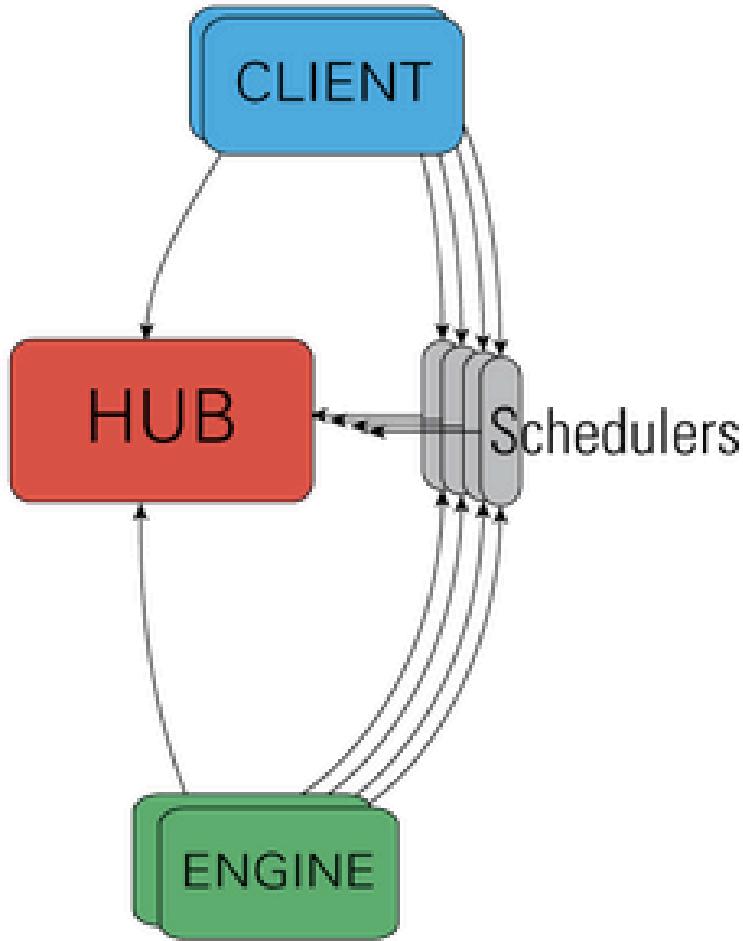


Mayavi

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



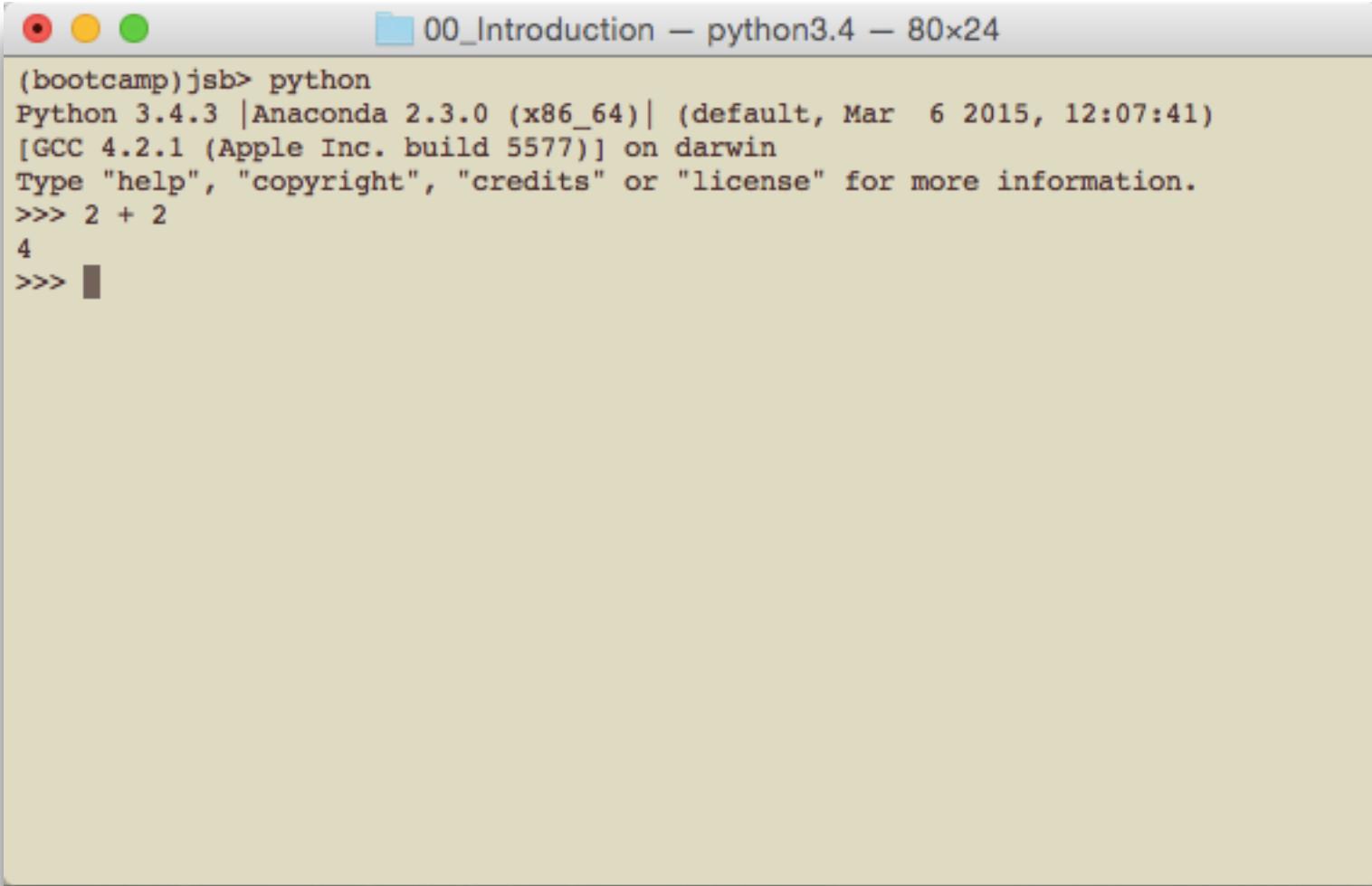
Asking Questions during Module presentations

- 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

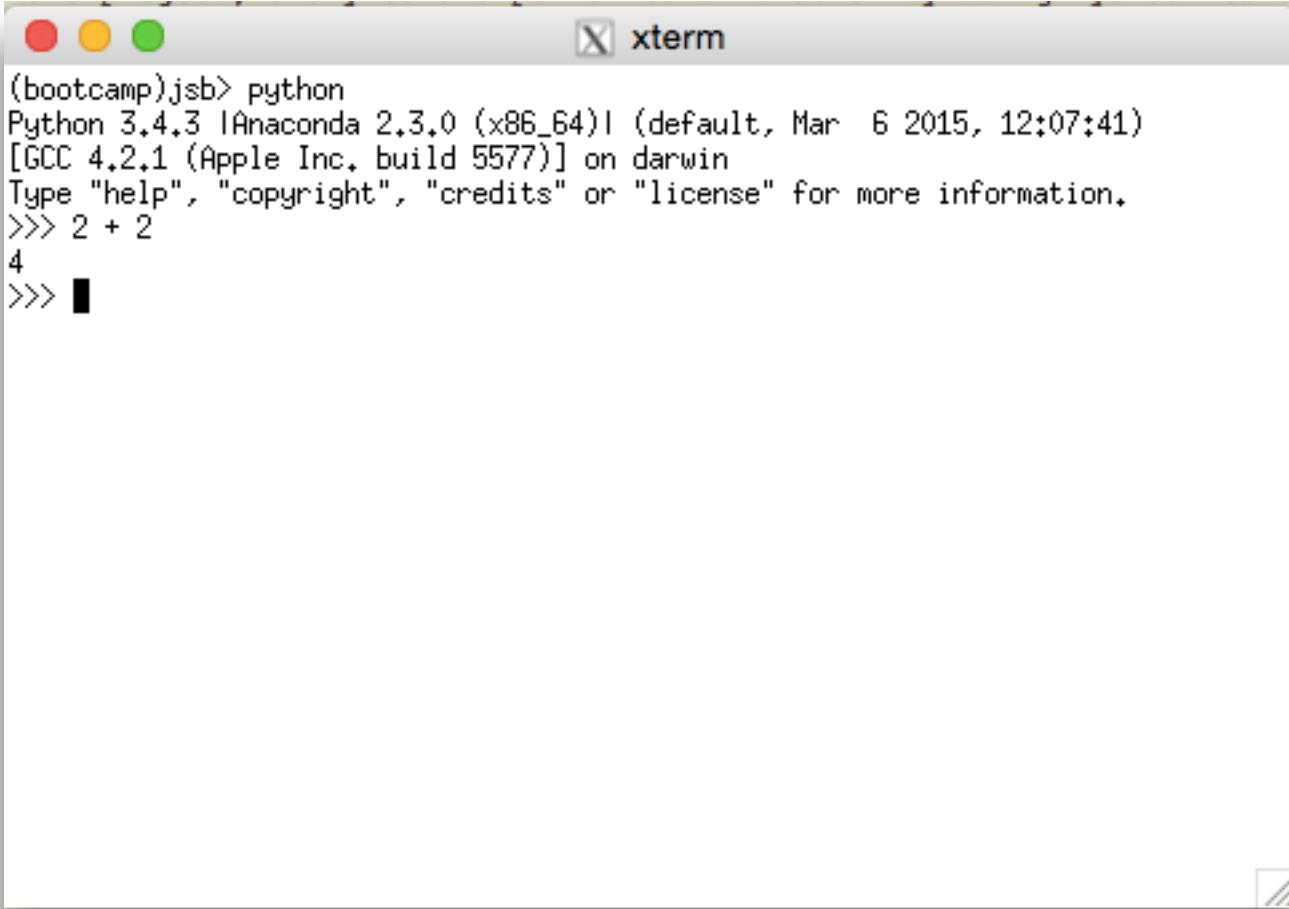


A screenshot of a Mac OS X terminal window titled "00_Introduction – python3.4 – 80x24". The window shows the Python 3.4.3 interpreter running in a Mac OS X environment. The output includes the Python version, build details, help information, and a simple calculation.

```
(bootcamp)jsb> python
Python 3.4.3 |Anaconda 2.3.0 (x86_64)| (default, Mar  6 2015, 12:07:41)
[GCC 4.2.1 (Apple Inc. build 5577)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> 2 + 2
4
>>> █
```

Mac OS X (Terminal)

Firing up the Interpreter

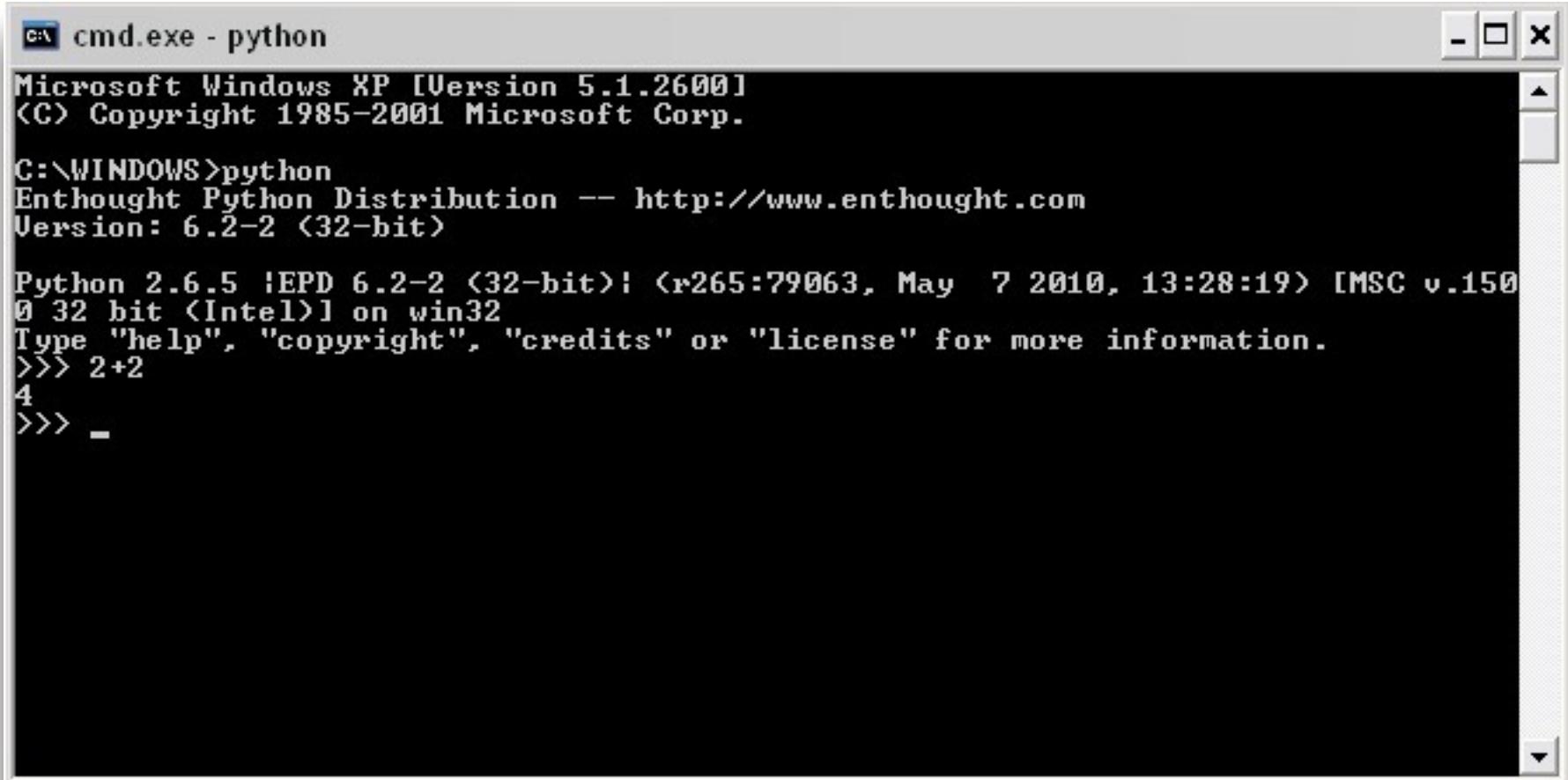


The image shows a screenshot of an xterm window titled "xterm". The window has three colored window controls (red, yellow, green) in the top-left corner. The title bar contains the word "xterm" and a close button. The main area of the window displays the following text:

```
(bootcamp)jsb> python
Python 3.4.3 |Anaconda 2.3.0 (x86_64)| (default, Mar  6 2015, 12:07:41)
[GCC 4.2.1 (Apple Inc. build 5577)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> 2 + 2
4
>>> █
```

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

Firing up the Interpreter



The screenshot shows a Microsoft Windows XP command prompt window titled "cmd.exe - python". The window displays the following text:

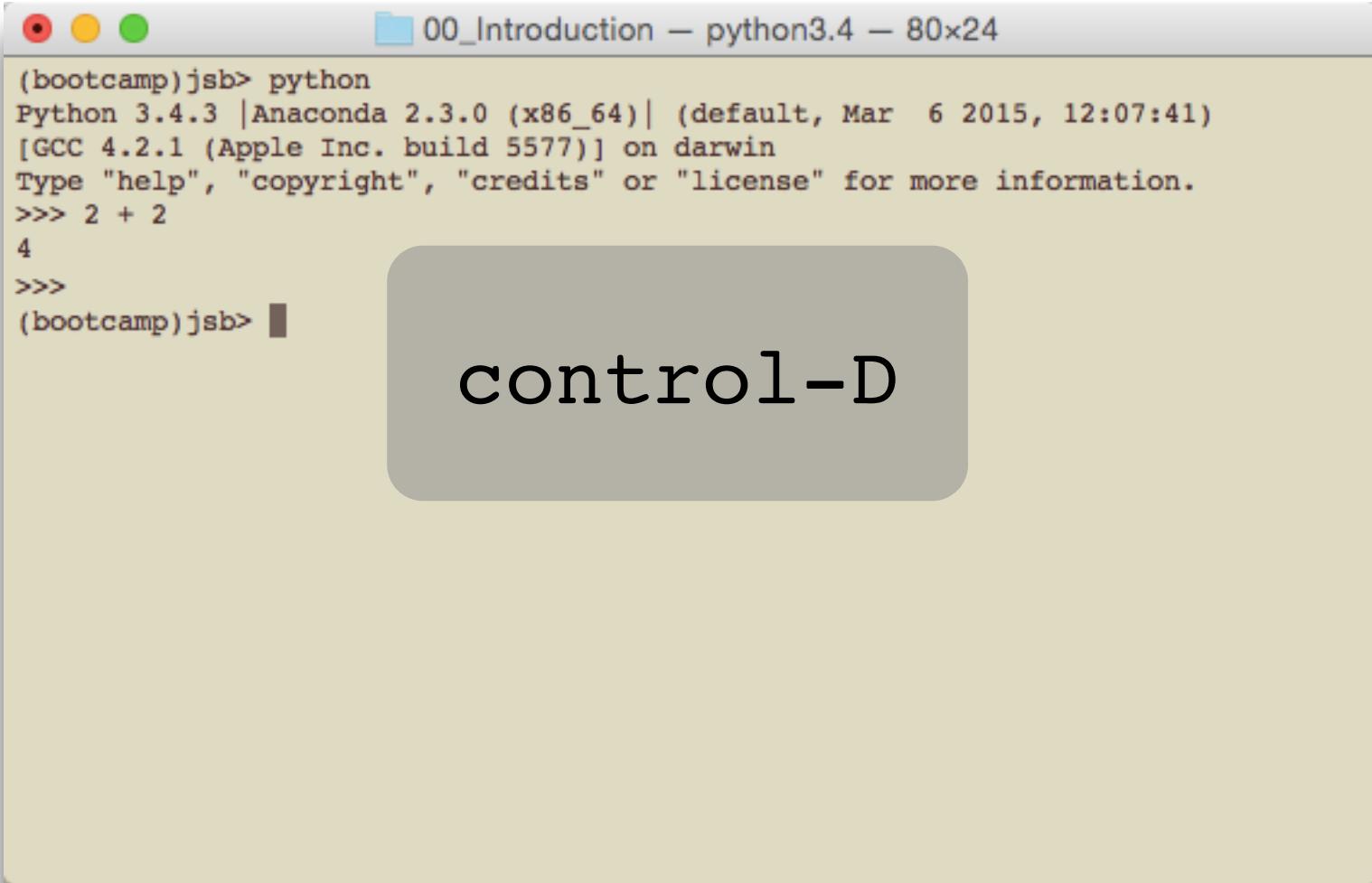
```
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.

C:\WINDOWS>python
Enthought Python Distribution -- http://www.enthought.com
Version: 6.2-2 (32-bit)

Python 2.6.5 |EPD 6.2-2 (32-bit)| (r265:79063, May  7 2010, 13:28:19) [MSC v.150
0 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> 2+2
4
>>> _
```

Windows

Firing up the Interpreter

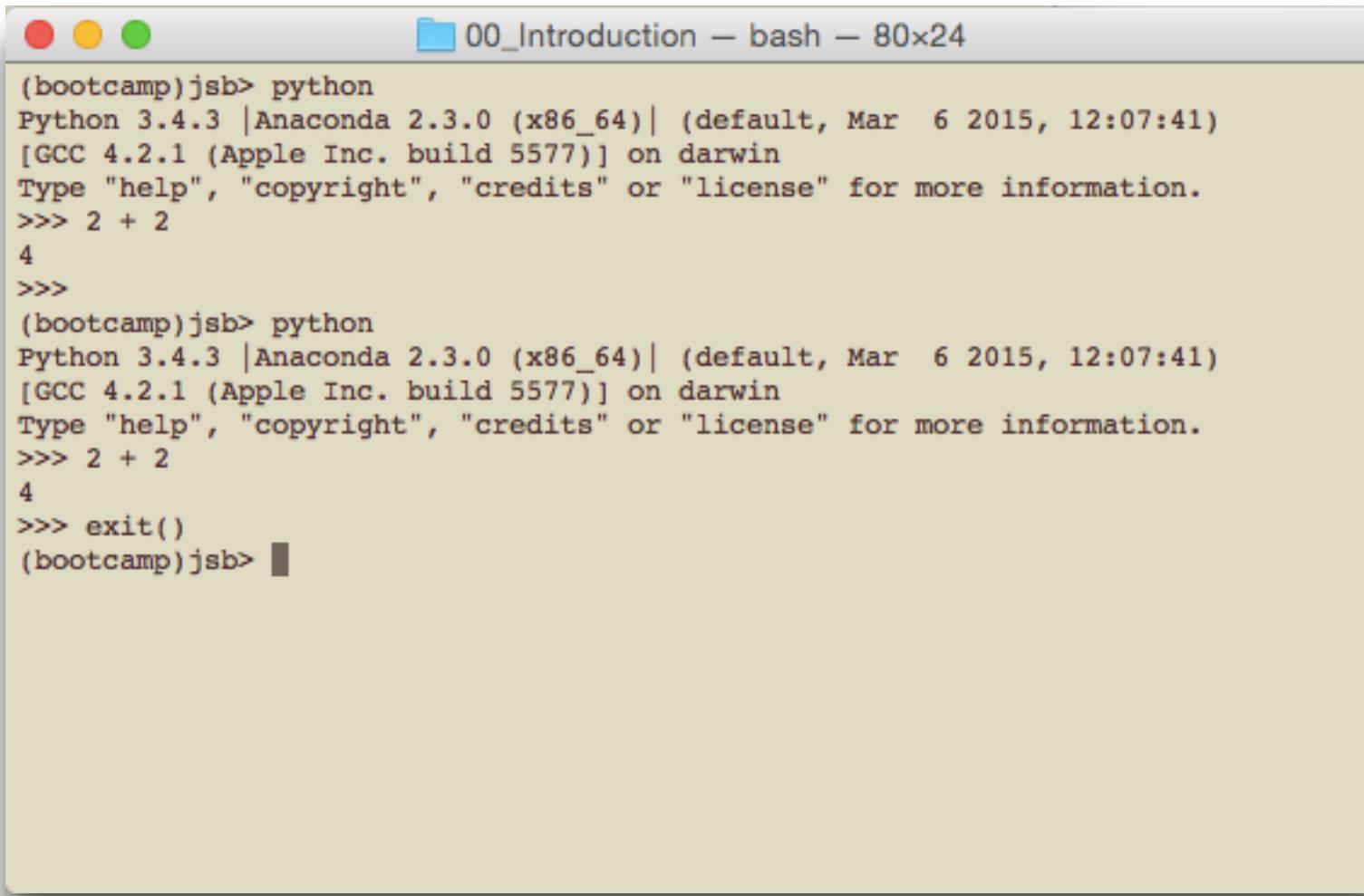


```
(bootcamp)jsb> python
Python 3.4.3 |Anaconda 2.3.0 (x86_64)| (default, Mar  6 2015, 12:07:41)
[GCC 4.2.1 (Apple Inc. build 5577)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> 2 + 2
4
>>>
(bootcamp)jsb>
```

control-D

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

Firing up the Interpreter

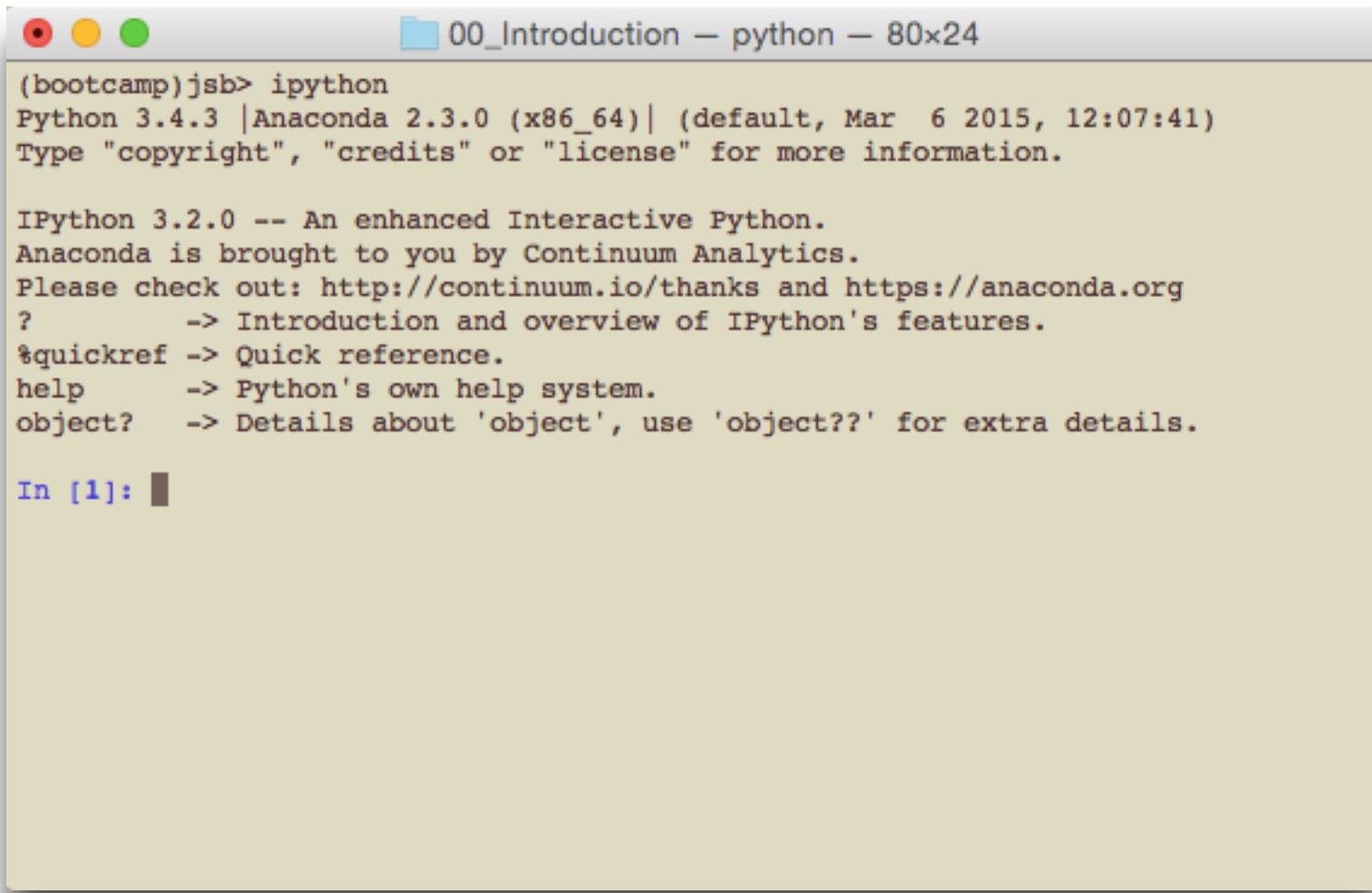


The screenshot shows a terminal window titled "00_Introduction – bash – 80x24". The window contains two separate sessions of the Python 3.4.3 interpreter. In the first session, the user types "2 + 2" and gets the output "4". In the second session, the user types "2 + 2" again and gets the output "4". Finally, the user types "exit()" to exit the interpreter. The terminal has a light beige background and a dark gray border.

```
(bootcamp)jsb> python
Python 3.4.3 |Anaconda 2.3.0 (x86_64)| (default, Mar  6 2015, 12:07:41)
[GCC 4.2.1 (Apple Inc. build 5577)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> 2 + 2
4
>>>
(bootcamp)jsb> python
Python 3.4.3 |Anaconda 2.3.0 (x86_64)| (default, Mar  6 2015, 12:07:41)
[GCC 4.2.1 (Apple Inc. build 5577)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> 2 + 2
4
>>> exit()
(bootcamp)jsb>
```

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

Firing up the Interpreter



A screenshot of a terminal window titled "00_Introduction – python – 80x24". The window shows the startup of the Python 3.4.3 interpreter using ipython. The output includes the Python version, Anaconda details, IPython version, and help command documentation. The text "In [1]:" is visible at the bottom left.

```
(bootcamp)jsb> ipython
Python 3.4.3 |Anaconda 2.3.0 (x86_64)| (default, Mar  6 2015, 12:07:41)
Type "copyright", "credits" or "license" for more information.

IPython 3.2.0 -- An enhanced Interactive Python.
Anaconda is brought to you by Continuum Analytics.
Please check out: http://continuum.io/thanks and https://anaconda.org
?          -> Introduction and overview of IPython's features.
%quickref -> Quick reference.
help       -> Python's own help system.
object?    -> Details about 'object', use 'object??' for extra details.

In [1]:
```

ipython

Editing Python Files



```
def union (self,other):  
    other = str(other)  
    hash = self.hash  
    for i in range(min(len(self.hash),len(other))):  
        if self.hash[i] != other[i]:  
            hash = self.hash[:i] + ("1" * (self.depth*2)) + other[i:]  
            break  
    return type(self)(hash,self.bound,self.depth)  
  
__add__ = union  
  
class Geohash (Geostring):  
    BASE_32 = "0123456789bcdefghjklmnpqrstuvwxyz"  
  
    def bitstring (cls,coord,bound=defbound,depth=defdepth):  
        bits = Geostring.bitstring(coord,bound,depth)  
        hash = ""  
        for i in range(0,len(bits),5):  
            m = sum([int(n)<<(4-j) for j,n in enumerate(bits[i:i+5])])  
            hash += cls.BASE_32[m]  
        return hash  
    bitstring = classmethod(bitstring)  
  
    def bbox (self,prefix=None):  
        if not prefix: prefix=len(self.hash)  
        bits = [[n>>(4-i)&1 for i in range(5)]  
                for n in map(self.BASE_32.find, self.hash)]  
        bits = reduce(lambda x,y:x+y, bits, [])  
        return self._to_bbox(bits)
```

Line: 188 Column: 44 | Python | Soft Tabs: 4 | def bitstring (cls,coord,bound=d



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, ...

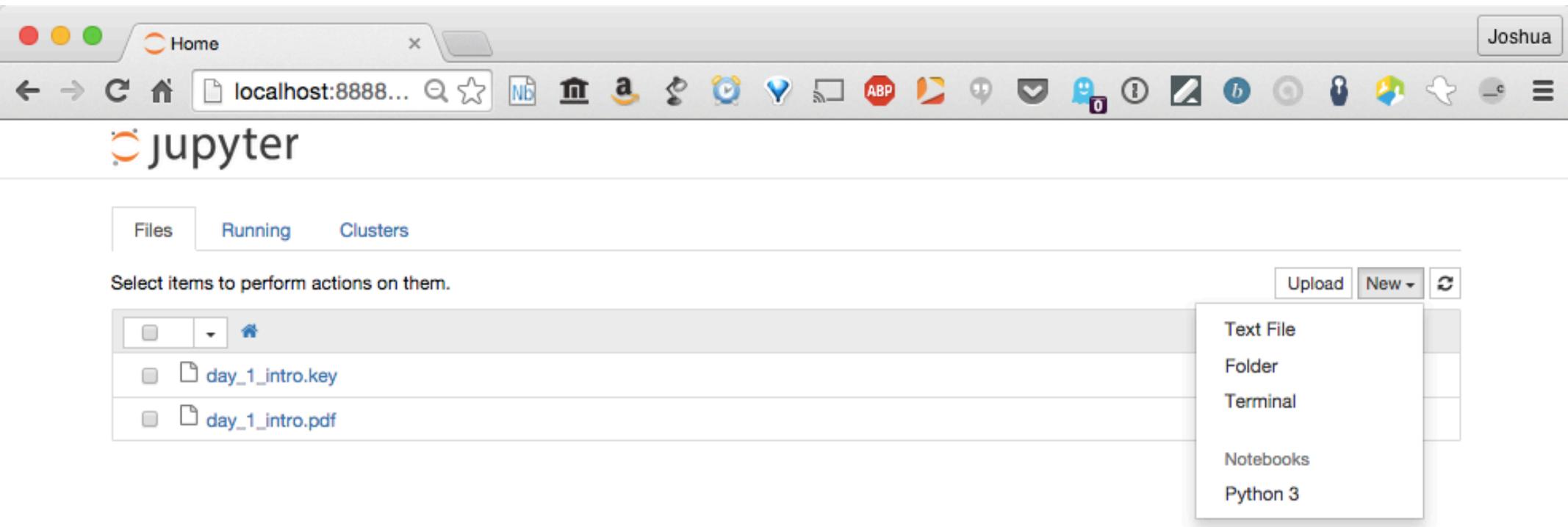
<http://www.sublimetext.com/2>

<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/Lecture  
[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 confirmation)
```

in your browser...



or ... try.jupyter.org

The screenshot shows the Jupyter Notebook interface running on try.jupyter.org. At the top, there's a browser header with tabs, icons, and a user profile. Below it is the Jupyter logo and navigation links for 'Files', 'Running', and 'Clusters'. A sidebar on the left lists 'communities', 'datasets', 'featured', and several 'Welcome' notebooks for Julia, R, Haskell, Python, and Gadfly. On the right, there's a toolbar with 'Upload' and 'New' buttons, and a dropdown menu showing options like 'Text File', 'Folder', 'Terminal', and various kernel choices (Bash, Haskell, Julia 0.3.2, Python 2, Python 3, R, Ruby 2.1.5). A context menu is open over a cell titled 'hello.py', which contains the following code:

```
1 # this is a text file
2 print(2+2)
```

This is a screenshot of a Jupyter Notebook cell. The title bar says 'jupyter hello.py' and indicates it was modified 'a few seconds ago'. The cell contains a single line of Python code:

```
1 # this is a text file
2 print(2+2)
```