

# IPython: a very quick overview

Fernando Pérez, Brian Granger, Min Ragan-Kelley  
and 80+ contributors...

UC Berkeley, Cal Poly San Luis Obispo

Oct 15, 2010

# IPython? “just a little afternoon hack” (Nov 2001)

...or “I’d rather not really write my dissertation”

## Getting all the power from interactive computing in Python

- 1 A better Python shell: object introspection, system access, ‘magic’ commands, ...
- 2 An embeddable interpreter: debugging, mix batch/interactive work.
- 3 A flexible component: base environment for systems with Python as the underlying language
- 4 A system for interactive control of distributed/parallel computing systems.

## Ohloh Analysis Summary

-  [Mostly written in Python](#)
-  [Mature, well-established codebase](#)
-  [Increasing year-over-year development activity](#)
-  [Large, active development team](#)
-  [Well-commented source code](#)

Updated 13 Oct 2010 07:10 UTC

## Ratings & Reviews

Community  
Rating

 4.6/5.0

Based on 68 user  
ratings.

Your Rating



Click to rate this  
project.

## Project Cost

This calculator estimates how much it would cost to hire a team to write this project from scratch. [More »](#)

Include

Markup And Code ▾

Codebase

77,003

Effort (est.)

19 Person Years

Avg. Salary

\$  year

**\$ 1,041,279**

## Licenses

Ohloh searches the source code for individual license declarations. These licenses can differ from the project's official license.

|  |           |
|--|-----------|
| <a href="#">BSD Copyright</a>                        | 257 files |
| <a href="#">MIT License</a>                          | 5 files   |
| <a href="#">GNU General Public License 2</a>         | 2 files   |
| <a href="#">GNU Lesser General Public License v3</a> |           |
| <a href="#">Apache License 2.0</a>                   | 1 files   |
| <a href="#">Python 2.4.2 license</a>                 | 1 files   |

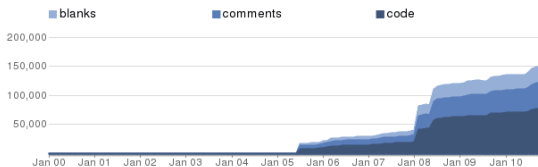
## Languages

Ohloh analyzes the project source code and determines the language of each line of code, excluding comments and blanks.

|                        |     |
|------------------------|-----|
| <a href="#">Python</a> | 92% |
| <a href="#">XML</a>    | 5%  |
| <a href="#">Other</a>  | 3%  |



## Lines of Code



## Lines of Code By Language

|  | Language                     | Code Lines | Comment Lines | Comment Ratio | Blank Lines | Total Lines |
|--|------------------------------|------------|---------------|---------------|-------------|-------------|
|  | <a href="#">Python</a>       | 71,224     | 43,335        | 37.8%         | 27,576      | 142,135     |
|  | <a href="#">XML</a>          | 3,931      | 0             | 0.0%          | 115         | 4,046       |
|  | <a href="#">Emacs Lisp</a>   | 522        | 364           | 41.1%         | 89          | 975         |
|  | <a href="#">CSS</a>          | 498        | 11            | 2.2%          | 91          | 600         |
|  | <a href="#">Perl</a>         | 401        | 440           | 52.3%         | 208         | 1,049       |
|  | <a href="#">Make</a>         | 252        | 49            | 16.3%         | 107         | 408         |
|  | <a href="#">shell script</a> | 143        | 106           | 42.6%         | 63          | 312         |
|  | <a href="#">HTML</a>         | 130        | 0             | 0.0%          | 65          | 195         |
|  | <a href="#">Vim Script</a>   | 124        | 2             | 1.6%          | 17          | 143         |
|  | <a href="#">Objective-C</a>  | 30         | 7             | 18.9%         | 12          | 49          |

# Actively developed: Git/GitHub are amazing

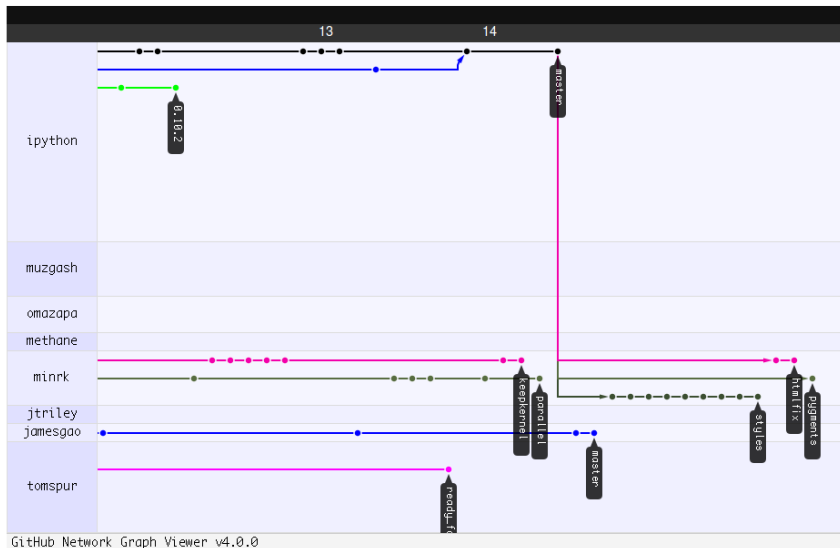
## The python network graph

Keyboard shortcuts available 

All branches in the network using `ipython/ipython` as the reference point. [Read our blog post about how it works.](#)

[Show Help](#)

Last updated: 9 minutes ago



# Projects using IPython

## Scientific

- **PyRAF**: Space Telescope Science Institute
- **CASA**: National Radio Astronomy Observatory.
- **Ganga**: CERN.
- **PyMAD**: neutron spectrometer, Institut Laue Langevin.
- **Sardana**: European Synchrotron Radiation Facility.
- **ASCEND**: engineering modeling (Carnegie Mellon).
- **JModelica**: dynamical systems.
- Denver Aerosol Sources and Health (**DASH**), CU Boulder.
- **PyIMSL Studio**, by Visual Numerics.
- **Trilinos**: Sandia National Lab.
- **Sage**: open source mathematics.
- **Pymerase**: microarray gene expression.

## Web/Other

- **Django** web framework.
- **Turbo Gears** web framework.
- **Pylons** web framework
- **Zope** and **Plone** CMS.
- Axon Shell, BBC **Kamaelia**.
- **Schevo** database.
- **Pitz**: distributed task/bug tracking.
- **iVR** (interactive Virtual Reality).
- **Movable Python** (portable Python environment).
- ...

# Matlab/IDL-like interactive use

```
fperex@longs:/home/fperex - Shell - Konsole
fperex[-]> ipython -pylab
Python 2.4.3 (#2, Apr 27 2006, 14:43:58)
Type "copyright", "credits" or "license" for more information.

IPython 0.7.3.svn -- An enhanced Interactive Python.
?         -> Introduction to IPython's features.
?magic    -> Information about IPython's 'magic' % functions.
help     -> Python's own help system.
object?  -> Details about 'object'. ?object also works, ?? prints more.

Welcome to pylab, a matplotlib-based Python environment.
For more information, type 'help(pylab)'.

In [1]: import math, numpy
In [2]: from scipy.integrate import quad
In [3]: from scipy.special import j0
In [4]: def j0i(x):
...:     """Integral form of J_0(x)"""
...:     def integrand(phi):
...:         return math.cos(x*math.sin(phi))
...:     return (1.0/math.pi)*quad(integrand,0,math.pi)[0]
In [5]: x = numpy.linspace(0,20,200) # sample grid: 200 points between 0 and 20
In [6]: y = j0(x) # sample J0 at all values of x
In [7]: x1 = x[::10] # subsample the original grid every 10th point
In [8]: y1 = map(j0i,x1) # evaluate the integral form at all points in x1
In [9]: # Make a plot with these values (the ; suppresses output)
In [10]: plot(x,y,label=r'$J_0(x)$');
In [11]: plot(x1,y1,'ro',label=r'$J_0\{\text{int}\}(x)$');
In [12]: axhline(0,color='green',label='nolegend');
In [13]: title(r'Verify $J_0(x)=\frac{1}{\pi}\int_0^\pi\cos(x\sin\phi)d\phi$');
In [14]: xlabel('$x$');
In [15]: legend();
In [16]: matshow(numpy.random.random((32,32)))
Out[16]: <matplotlib.figure.Figure instance at 0x4630042c>
```

Figure 1

Verify  $J_0(x) = \frac{1}{\pi} \int_0^\pi \cos(x \sin \phi) d\phi$

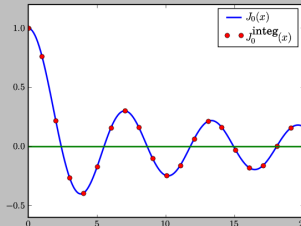
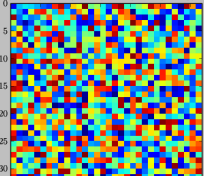


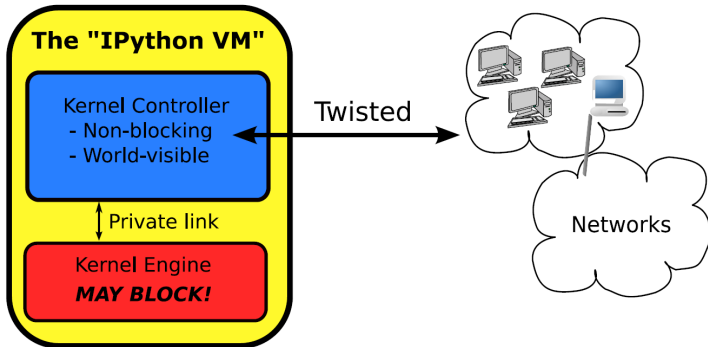
Figure 2



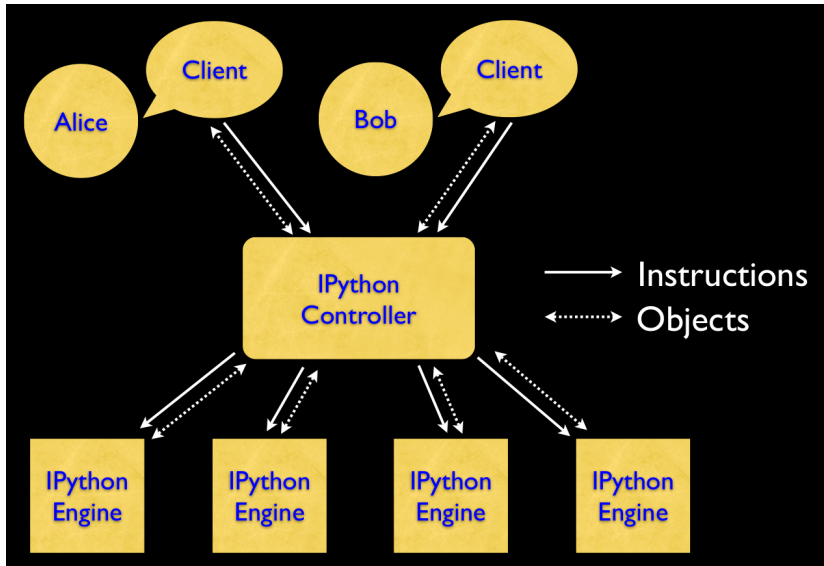
# Parallel work, the story so far...

- Parallel computing: **fully interactive**
  - development, debugging, testing, execution, monitoring,...
- Easy things should be **easy**, difficult things **possible**
- Make parallel computing **collaborative**
- More dynamic model for **load balancing** and **fault tolerance**
- Seamless integration with other tools: plotting/**visualization**, **system shell**.
- Also want to keep the benefits of traditional approaches:
  - Should integrate with threads/MPI if appropriate
  - Should be easy to integrate compiled code and libraries
- Support many types of parallelism

# Network-aware IPython



# Parallel design



# What does IPython offer here?

- **Easy reuse** and distribution of existing serial ('normal') codes.
- High-level abstractions for 'embarrassingly parallel' problems.
  - Direct execution of code over the network: **multiengine** interface.
  - Out-of-the box **task farming** tools: **task** interface.
- Task farming system is "low-latency" (**not** in the Myrinet sense...)
  - can be integrated into more complex codes.
- Implement any approach to parallelism you want:
  - Synchronous or asynchronous execution of code on nodes.
  - Task farming.
  - Traditional Message Passing (MPI).
  - Integrate hybrid codes.

# IPVision: visual distributed computing

Michel Sanner, Jose Unpingco, Ananth Devulapalli [Ohio Supercomputing Center/OSU]

The image displays the IPVision software interface, which is used for visual distributed computing. The main window is titled "Vision" and shows a data flow graph for a "beamPat5" process. The graph includes several components: a "range" block with "from" and "stop" values, a "DivArts" block with a 3D vector visualization and numerical values for x, y, and z, a "Function run" block with an "import" statement, a "Log10" block, and a "Multiply" block. The graph is connected to a "Polar" window showing a 2D visualization of beam patterns, which are represented as concentric circles and a central spot on a color scale from blue to red. Below the main window, there are two terminal windows. The first terminal window shows the command "ipcluster -n 4" and its output, including starting controller and engines, and instructions for interactive use. The second terminal window shows the command "ipcluster" and its output, including a list of results and the execution of a function.

**beamPat5**

range  
from: [0, 0]  
to: [100, 100]  
step: [1, 1]

DivArts  
-fold vector  
x: [0.47244]  
y: [0.70702]  
z: [0.52624]  
normalize and set

Function run  
import [from beamPattern import]

Log10  
operator [log10]

Multiply  
operator [multiply]

**Polar**

100  
50  
0  
-50  
-100  
-200 -150 -100 -50 0 50 100 150 200

**fperez@bic128:~\$ ipcluster -n 4**  
Starting controller: Controller PID: 12832  
Starting engines: Engines PIDs: [12834, 12835, 12836, 12837]  
Log files: /home/fperez/.ipython/log/ipcluster-12832-\*

Your cluster is up and running.

For interactive use, you can make a MultiEngineClient with:

```
from IPython.kernel import client  
sec = client.MultiEngineClient()
```

You can then cleanly stop the cluster from IPython using:

```
sec.kill(controller=True)
```

You can also hit Ctrl-C to stop it, or use from the cmd line:

```
ipcluster
```

**fperez@bic128:~/research/ipyvision/beam - Vision - Konsole**

```
>>> <Results List>  
(0) In [3]: from beamPattern import run  
(1) In [3]: from beamPattern import run  
(2) In [3]: from beamPattern import run  
(3) In [3]: from beamPattern import run  
  
L scattering el [-90, -84, -78, -72, -66, -60, -54, -48, -42, -36, -30, -24, -18, -12, -6, 0, 6, 12, 18, 24, 30, 36, 42, 48, 54, 60, 66, 72, 78, 84]  
[None, None, None, None]  
pull [[-90, -84, -78, -72, -66, -60, -54, -48], [-42, -36, -30, -24, -18, -12, -6, 0], [6, 12, 18, 24, 30, 36, 42], [48, 54, 60, 66, 72, 78, 84]]  
EXECUTING2 result = run(el, El=el, Az=Az, locdata=locdata, locfile=locfile )  
<Results List>  
(1) In [4]: result = run(el, El=el, Az=Az, locdata=locdata, locfile=locfile )  
(2) In [4]: result = run(el, El=el, Az=Az, locdata=locdata, locfile=locfile )  
(3) In [4]: result = run(el, El=el, Az=Az, locdata=locdata, locfile=locfile )  
(4) In [4]: result = run(el, El=el, Az=Az, locdata=locdata, locfile=locfile )
```

# Some technical notes

- Networking: [Twisted](#)
  - High-level interfaces: no need to learn Twisted.
- RPC: Twisted's [foolscap](#)
- [Security](#): foolscap supports SSL ([pyOpenSSL](#)) and a capabilities model.
  - [Review/improvements welcome, we're not security experts!](#)
- MPI support is there, use [mpi4py](#) bindings.
- Integration with queuing systems, better process control coming...

## “Sockets done right”

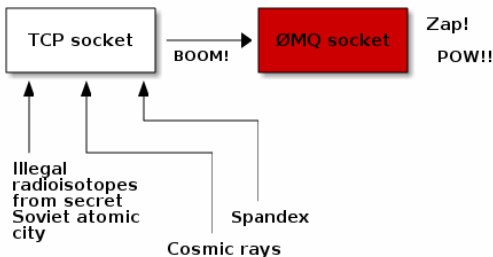
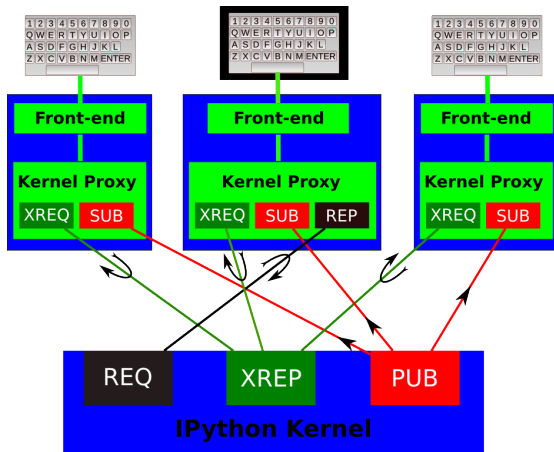


Figure 1 - A terrible accident...

- C/C++ library
- Python bindings in Cython (Brian Granger, Min RK)
- Python bindings run messaging in native threads - **no GIL**
- Abstractions are at the **message delivery** level, not the raw-bytes level.
- Socket types encapsulate **messaging patterns**

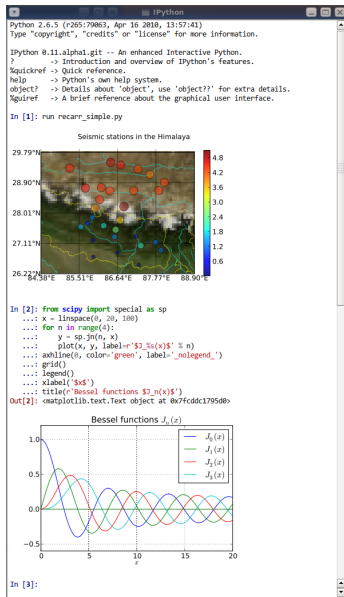
# Interactive IPython on ØMQ



- - Kernel raw\_input
- - Requests to kernel
- - Kernel output broadcast
- ↻ - Request/Reply direction

# Rich Qt Console

Enthought: sponsorship, Evan Patterson.



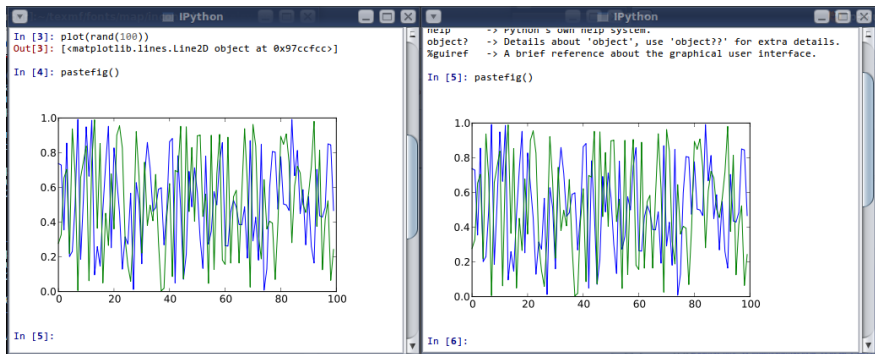
## Feels like a console, runs like a GUI

- Inline and floating images
- Syntax highlighting, full multiline editing
- Session saving
  - HTML (with PNG or SVG)
  - PDF/printing
- Help viewer
- %magics, !system access, IPython...
- Detach/reattach support

# Collaborative interactive computing

'Reverse parallelism', or Google docs for interactive computing

Multiple users of one process instead of many processes for one user



These could be two different hosts on separate networks

# Where to next?

- **ZeroMQ-based architecture**: very solid design, polish work to do...
- **Release 0.11**: December 2010
  - 0.10.2 was out on Tuesday Oct 12.
- **Clients**:
  - Continue improving **Qt console**: lots and lots of ideas.
  - Rich **Qt “notebook” client**: prototype exists (Google SoC 2010).
  - Collaborative **web client**: prototype already started at UC Berkeley (two days ago). AJAX/HTML5.
- **Parallel architecture on ZeroMQ** (later today - see Min's talk)
- **Python3 support**: experimental branch already on GitHub.

New ideas from today???

# Where to next?

- **ZeroMQ-based architecture**: very solid design, polish work to do...
- **Release 0.11**: December 2010
  - 0.10.2 was out on Tuesday Oct 12.
- **Clients**:
  - Continue improving **Qt console**: lots and lots of ideas.
  - Rich **Qt “notebook” client**: prototype exists (Google SoC 2010).
  - Collaborative **web client**: prototype already started at UC Berkeley (two days ago). AJAX/HTML5.
- **Parallel architecture on ZeroMQ** (later today - see Min's talk)
- **Python3 support**: experimental branch already on GitHub.

**New ideas from today???**