

Why Modern, High-performance Networking Matters for Interactive Computing

Fernando Pérez

Helen Wills Neuroscience Institute, UC Berkeley

<http://fperez.org>

Fernando.Perez@berkeley.edu

Brian Granger, Cal Poly, San Luis Obispo
Evan Patterson, California Institute of Technology

SIAM CSE 2011, Reno, NV

March 2, 2011

IPython: I is for interactive...

In scientific computing,
we typically **don't know what we're doing.**

Scientific computing \Leftrightarrow *Exploratory* computing

IPython: I is for interactive...

In scientific computing,
we typically **don't know what we're doing.**

Scientific computing \Leftrightarrow *Exploratory* computing

IPython: I is for interactive...

In scientific computing,
we typically **don't know what we're doing**.

Scientific computing \Leftrightarrow *Exploratory* computing

Python: an excellent *base* for an interactive scientific system

- Dynamic code evaluation
- No variable declarations
- Powerful introspection
- Very regular object model
- Excellent string processing

IPython:
far beyond Python's interactive shell

Python: an excellent *base* for an interactive scientific system

- **Dynamic** code evaluation
- No variable declarations
- Powerful **introspection**
- Very **regular** object model
- Excellent **string** processing

IPython:
far beyond Python's interactive shell

Python: an excellent *base* for an interactive scientific system

- **Dynamic** code evaluation
- No variable declarations
- Powerful **introspection**
- Very **regular** object model
- Excellent **string** processing

IPython:
far beyond Python's interactive shell

Object info??

```

Type:      module
Base Class: <type 'module'>
String Form: <module 'code' from '/usr/lib/python2.6/code.pyc'>
Namespace: Interactive
File:      /usr/lib/python2.6/code.py
Source:
"""Utilities needed to emulate Python's interactive interpreter.
"""

# Inspired by similar code by Jeff Epler and Fredrik Lundh.

import sys
import traceback
from codeop import CommandCompiler, compile_command

__all__ = ["InteractiveInterpreter", "InteractiveConsole", "interact",
           "compile_command"]

def softspace(file, newvalue):
    oldvalue = 0
    try:
        oldvalue = file.softspace
    except AttributeError:
        pass
    try:
        file.softspace = newvalue

```

```
lines 1-28
```

When things go wrong

```
/bin/bash

In [13]: run ~/scratch/error
reps: 5

-----
ValueError                                Traceback (most recent call last)
/home/fperez/scratch/error.py in <module>()
     70 if __name__ == '__main__':
     71     #explode()

--> 72     main()
     73     g2='another global'

/home/fperez/scratch/error.py in main()
     60     array_num = zeros(size,'d')
     61     for i in xrange(reps):
--> 62         RampNum(array_num, size, 0.0, 1.0)
     63         RNTIME = time.clock()-t0
     64         print 'RampNum time:', RNTIME

/home/fperez/scratch/error.py in RampNum(result, size, start, end)
     43     tmp = zeros(size+1)
     44     step = (end-start)/(size-1-tmp)
--> 45     result[:] = arange(size)*step + start
     46
     47 def main():

ValueError: shape mismatch: objects cannot be broadcast to a single shape

In [14]: □
```

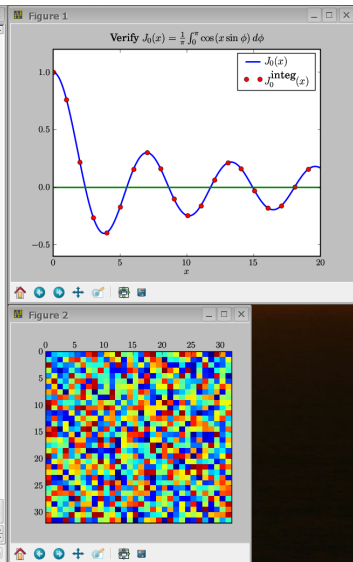
Plotting at the console

```
fperex@longs:/home/fperex - Shell - Konsole
longs[-]> 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 = j0i(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'$J_0^{\text{integ}}(x)$');
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>
```



Embedded widget

The screenshot displays the Mayavi application interface. The main window shows a 3D visualization of a brain MRI scan with a red wireframe overlay. The interface is divided into several panels:

- File Visualize View Tools Help**: The main menu bar.
- Mayavi**: The main application window title.
- TVTK Scene 3**: A panel showing the scene hierarchy, including **ArraySource**, **Colors and legends**, **ImagePlaneWidget**, **ImagePlaneWidget**, and **ScalarCutPlane**.
- Mayavi object editor**: A panel with various settings and sliders, including **Enabled**, **teraction**, **activation**, **o volume**, **terpolate**, **visibility**, **us cursor**, and **oup table**. It also includes a **x_axes** dropdown, a **3** input field, and a **linear** radio button. Sliders for **endle size** (0.001 to 0.5), **ion value** (1), and **on action** (0 to 2) are visible. The **Origin** is set to **F0: 41.804807, F1: -0.5, F2: -0.5**. The **lice factor** is 0.010. **Point1** is **F0: 41.804807, F1: 255.5, F2: -0.5**. **Point2** is **F0: 41.804807, F1: -0.5, F2: 239.5**. Other settings include **Priority: 0.0**, **on action: 0**, **modifier: 0**, **lice index: 42**, and **osition: 41.8048074008**.
- TVTK Scene 3**: The main 3D visualization area showing the brain MRI scan with a red wireframe overlay.
- IPython**: A panel at the bottom right showing the Python console with the following code:

```
In [0]: run brain.py
In [7]: rescale(ima,max=255)
In [8]: imb = ima.astype(np.uint8)
In [9]: view_numpy(imb)
```

Getting all the power from interactive computing in Python

- 1 A better Python shell: object introspection, system access, extensible 'magic' commands, ...
- 2 A flexible, embeddable interpreter:
 - 1 debugging, mix batch/interactive work.
 - 2 build custom systems based on Python with new syntax, etc.
- 3 Data visualization and GUIs: Matplotlib, Mayavi, all GUIs toolkits.
- 4 A rich toolkit: terminal, Qt console, HTTP client.
- 5 High level (and interactive!) parallel computing interfaces.

People who use IPython also use:



Ohloh Analysis Summary

- [Mostly written in Python](#)
- [Mature, well-established codebase](#)
- [Very large, active development team](#)
- [Extremely well-commented source code](#)

Updated 01 Mar 2011 14:14 UTC

Ratings & Reviews

Community Rating



Based on 71 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	<input type="text" value="Markup And Code"/>
Codebase	78,759
Effort (est.)	19 Person Years
Avg. Salary	\$ <input type="text" value="55000"/> year
\$ 1,065,165	

Lines of Code By Language

	Language	Code Lines	Comment Lines	Comment Ratio	Blank Lines	Total Lines
	Python	72,290	44,415	38.1%	28,013	144,718
	XML	3,931	0	0.0%	115	4,046
	HTML	819	0	0.0%	65	884
	Emacs Lisp	522	364	41.1%	89	975
	CSS	499	11	2.2%	91	601
	Perl	401	440	52.3%	208	1,049
	Make	254	49	16.2%	108	411
	shell script	143	106	42.6%	63	312
	Vim Script	124	2	1.6%	17	143
	Objective-C	30	7	18.9%	12	49

This analysis was updated about 18 hours ago. (01 Mar 2011 14:14 UTC)

Some 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).
- ...

IPython: a REPL (Read/Eval/Print Loop)

Core idea: manage a namespace

- Read: take user input.
- Eval: execute code.
- Print: provide output.
- Add support for data transfer...

...and interactive and parallel work start looking very similar.

These steps can happen in multiple processes:

- Read: user environment
- Eval: kernel process
- Print: user environment

IPython: a REPL (Read/Eval/Print Loop)

Core idea: manage a namespace

- Read: take user input.
- Eval: execute code.
- Print: provide output.
- Add support for data transfer...

...and interactive and parallel work start looking very similar.

These steps can happen in multiple processes:

- Read: user environment
- Eval: kernel process
- Print: user environment

IPython: a REPL (Read/Eval/Print Loop)

Core idea: manage a namespace

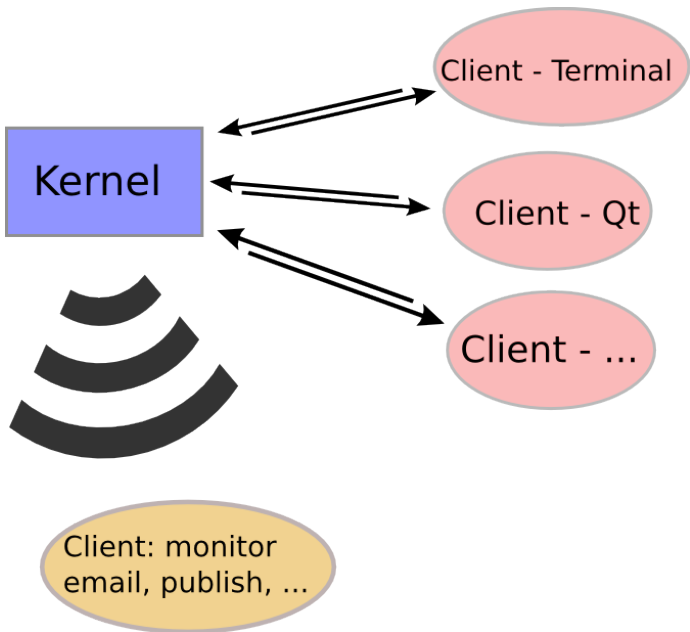
- Read: take user input.
- Eval: execute code.
- Print: provide output.
- Add support for data transfer...

...and interactive and parallel work start looking very similar.

These steps can happen in multiple processes:

- Read: user environment
- Eval: kernel process
- Print: user environment

More complex interactive uses?



A messaging protocol

Direct communication

- Execute code ('eval')
- Object information
- Complete
- History
- Connect

Broadcasting

- Functional execution:
 - Python inputs
 - Python outputs
 - Python errors
- Side effects:
 - Streams (stdout, stderr, etc)
 - Display data: plots, other payloads

The socket library that acts as a concurrency framework

- Pure C++ library.
- Python bindings in [Cython](#) (Brian Granger, Min RK). Python 2.5-3.2.
- 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](#).
- Open source (LGPL), actively developed.

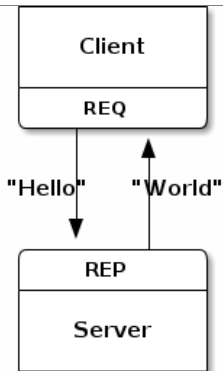


Figure 1 – Request-Reply

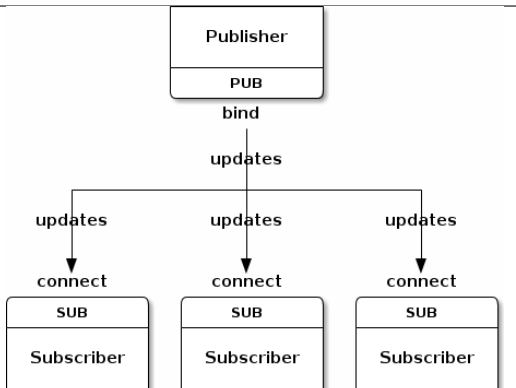
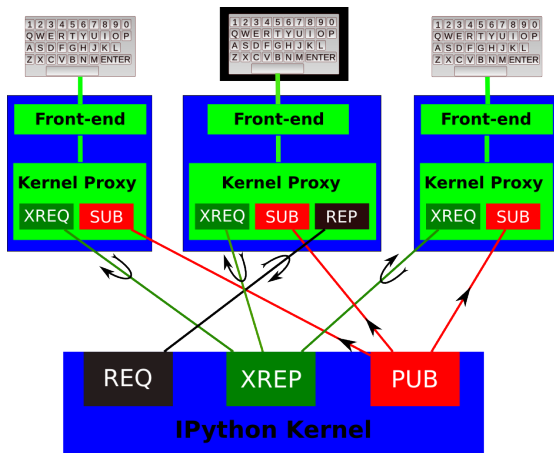


Figure 4 – Publish-Subscribe

Image credit: official ØMQ documentation

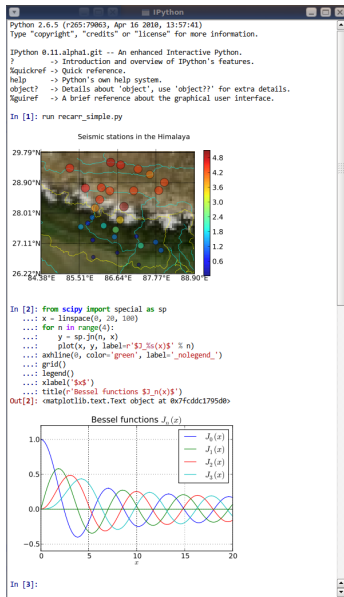
Interactive IPython on ØMQ



- - Kernel raw_input
- - Requests to kernel
- - Kernel output broadcast
- ↻ - Request/Reply direction

Back to the clients: a 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

Forward \emptyset MQ to HTTP: a web frontend!

James Gao, UC Berkeley

AJAX test

localhost:8080/notebook

```
In [16]: t = linspace(-pi, pi, 1024)
s = sin(10*t) / t
plot(t, s, linewidth=1.0)

xlabel('time (s)')
ylabel('voltage (mV)')
title('About as simple as it gets, folks')
grid(True)
```

About as simple as it gets, folks

voltage (mV)

time (s)

History

```
[16]: t = linspace(-pi, pi, 1024)
s = sin(10*t) / t
plot(t, s, linewidth=1.0)

xlabel('time (s)')
ylabel('voltage (mV)')
title('About as simple as it gets, folks')
grid(True)
```

Variables

Namespace display not implemented

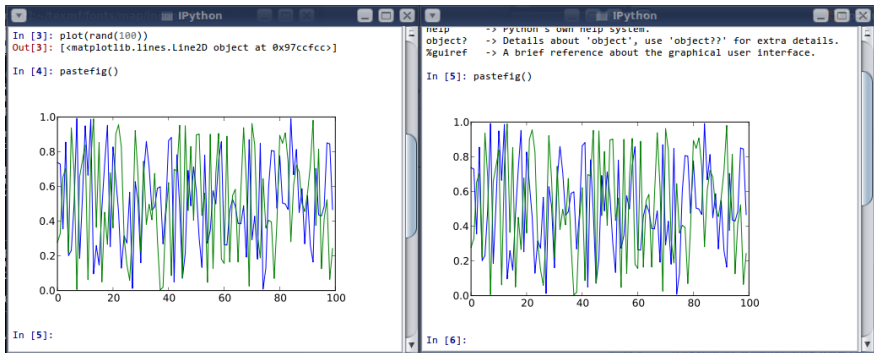
In [17]: |

idle

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

A short demo

(time permitting)

To wrap up: why should you care?

- **Interactive user**: interfaces adapted to each environment
 - Terminal. For those days when you can only SSH into your cluster...
 - Web.
 - Rich Qt client...
- **Scientific programmer**: embed an IPython kernel into your code
 - Monitor it over the network,
 - Inspect it interactively, ...
- IPython as **your interactive shell: custom magics.**
 - `%pyomo [options] <model.py> [<model.dat>]`
- Build **custom interactive environments**
 - even beyond the Python language - Sage
- **Civilized parallelism - next talk.**

Just one idea

Reuse a common set of abstractions to manipulate a namespace, define the protocols to do it and use a networking layer that makes it possible.

Support for IPython

- Enthought <http://www.enthought.com>.
- Bivio Software <http://www.bivio.biz>
- Tech-X corporation <http://txcorp.com>
- Ohio Super Computing Center and Defense High Performance Computing Modernization Program (HPCMP) - José Unpingco, Ohio State.
- NIH (NiPy project) - Grant 5R01MH081909-02 from National Institute Of Mental Health, IRG: ZRG1. Mark D'Esposito, UC Berkeley.
- Google (Summer of Code).
- Microsoft - Wenming Ye and Shahrokh Mortazavi.
- NSF - William Stein, U. Washington.
- DOD/AFOSR - John Verboncoeur, UC Berkeley/U. Michigan.

(Incomplete) Cast of Characters

- **Brian Granger** - Physics, Cal State San Luis Obispo
- **Min Ragan-Kelley** - UC Berkeley
- **Robert Kern** - Enthought
- **Jörgen Stenarson** - Sweden.
- **Evan Patterson** - Physics, Caltech
- **Thomas Kluyver** - Plant biology, U. Sheffield
- Stéfán van der Walt - Applied Math, U. Stellenbosch, South Africa
- John Hunter - TradeLink Securities, Chicago.
- Prabhu Ramachandran - Aerospace Engineering, IIT Bombay.
- Satra Ghosh- MIT Neuroscience
- Gaël Varoquaux - Neurospin (Orsay, France)
- Ville Vainio - CS, Tampere University of Technology, Finland
- Barry Wark - Neuroscience, U. Washington.
- Ondrej Certik - Physics, U Nevada Reno
- Laurent Dufrécho - France
- Darren Dale - Cornell
- Justin Riley - MIT
- James Gao - UC Berkeley
- Mark Voorhies - UC San Francisco
- Thomas Spura - Fedora project
- **Many more!** (~60 commit authors)

Upcoming conferences

- US Scipy Conference: 2001-today.
 - **SciPy 2011: Austin, TX. July 11-16..**
- EuroScipy: since 2008.
 - **EuroScipy 2011: Paris, August 25-28.**
- Scipy India: since 2009.
 - **Scipy India 2011: December.**
- Scipy Japan 2011
 - **Being planned...**
- **Many workshops (general and discipline specific)**

Thank You!

<http://ipython.scipy.org>

<http://github.com/ipython>

<http://fperez.org>

Fernando.Perez@berkeley.edu