Practical Scientific Computing in Python

John D. Hunter
Fernando Pérez

with contributions from

Perry Greenfield
Travis E. Oliphant
Prabhu Ramachandran

Contents

Chapter 1. Why python? 5

6.11. Statistics 91
6.12. Interfacing with Python Imaging Library 91
6.13. Some examples 91

Chapter 7. 3D visualization with MayaVi 95
7.1. Introduction 95
7.2. Getting started 96
7.3. Using MayaVi 98
7.4. Using MayaVi from Python 108
7.5. Scripted examples 113

Chapter 8. 3D visualization with VTK 115
8.1. Hello world in VTK 115
8.4. Working with medical image data 117

Chapter 9. Interfacing with external libraries 121
9.1. weave 121
9.2. swig 130
9.3. f2py 130
9.4. Others 136
9.5. Distributing standalone applications 136

Bibliography 137

CHAPTER 1

Why python?

To be written...

CHAPTER 2

A whirlwind tour of python and the standard library

2.2. Python is a calculator

Aside from my daughter's solar powered cash-register calculator, Python is the only calculator I use. From the python shell, you can type arbitrary arithmetic expressions.

```
>>> 2+2
4
```

```
        FUNCTIONS
            acos(...)
                acos(x)

                Return the arc cosine (measured in radians) of x.

            asin(...)
                asin(x)

                Return the arc sine (measured in radians) of x.
```

And much more which is snipped. Likewise, we can get information on the complex object in the same way

```
>>> s = "Hi Mo6!"
>>> s = """Porky said, "That's all folks!" """
```

You can add strings together to concatenate them

```
# concatenating strings
>>> first = 'John'
>>> last = 'Hunter'
>>> first+last
'JohnHunter'
```

or call string methods to process them: upcase them or downcase them, or replace one character with another

```
# string methods
```

Exercise 2.6. Suppose you have data files named like

```
data/2005/exp0100.dat
data/2005/exp0101.dat
data/2005/exp0102.dat
...
data/2005/exp1000.dat
```

Write the python code that iterates over these files, constructing the filenames as strings in using `os.path.join` to construct the paths in a platform-independent way. *Hint*:ead the help for `os.path.join`!

OK, I promised to torture you a bit more with string interpolation – don't worry, I remembered. The ability to properly format your data when printing it is crucial in scientific endeavors: how many

```
['__add__', '__class__', '__contains__', '__de-
lattr__', '__delitem__', '__del-
slice__', '__doc__', '__eq__', '__ge__', '__getat-
tribute__', '__getitem__', '__get-
```

function that can be overriddei. Below is an example which provides a iormalize keyword argumet. The default argumet is `iormalize=Nione`

```
>>> norm = Normalize(65356) # good for 16 bit images
>>> norm(255)               # call this function
```

but we didn't actually do anything with these files. Here we'll show how to read in the data and do

CHAPTER 3

A tour of IPython

-

```
In [3]: z=x+3
In [4]: print 'y is:',y,'and z is:',z
y is: 5 and z is: 4
```

```
    print "We found bigobject!
```

system:

```
# Try running this code
```

```
# - A
```

```
print 'Main program finished. Bye!'

#
```

```
    def ipshell(): pass

#
```

CHAPTER 4

CHAPTER 5

Introduction to plotting with matplotlib / pylab

5.1. A bird's eye view

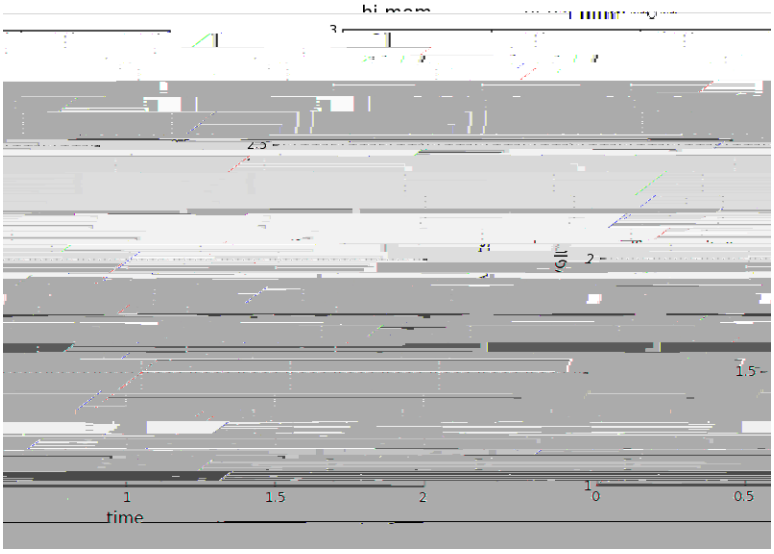

Figure 5.1.1.

The point under your mouse when you begin the zoom remains stationary, allowing you to zoom to an

Figure 5.2.2.

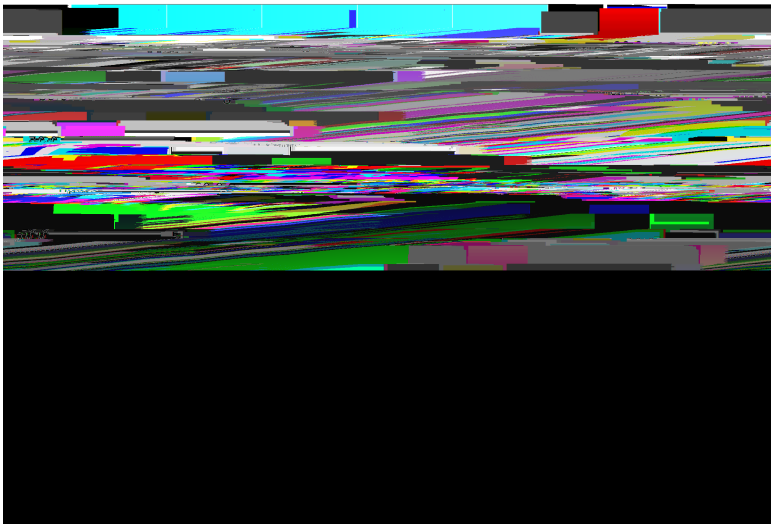

Figure 5-7.2. ter26(olm 0(at)1wi)1on2.491matr mOasR14RcolmOerd dfiRyn283oud71(r)-213owen with

lm 0(at)1wid th16Rwiter26(olm 0(at)1wi)1on2th271(d)1s.T

CHAPTER 6

A tour of SciPy

Chapter contributed by Travis E. Oliphant

6.1. Introduction

SciPy is a collection of mathematical algorithms and convenience functions built on the Numeric extension for Python. It adds significant power to the interactive Python session by exposing the user to high-level commands and classes for the manipulation and visualization of data. With SciPy, an interactive Python session becomes a data-processing and system-prototyping environment rivaling sytems such as Matlab, IDL, Octave, R-Lab, and SciLab.

The additional power of using SciPy within Python, however, is that a powerful programming

```
        [0, 1, 2, 3, 4],
        [0, 1, 2, 3, 4]]])
>>> mgrid[0:5:4j,0:5:4j]
array([[[ 0.    ,  0.    ,  0.    ,  0.    ],
        [ 1.6667,  1.6667,  1.6667,  1.6667],
        [ 3.3333,  3.3333,  3.3333,  3.3333],
        [ 5.    ,  5.    ,  5.    ,  5.    ]],
       [[ 0.    ,  1.6667,  3.3333,  5.    ],
        [ 0.    ,  1.6667,  3.3333,  5.    ],
        [ 0.    ,  1.6667,  3.3333,  5.    ],
        [ 0.    ,  1.6667,  3.3333,  5.    ]]])
```

Having meshed arrays like this is sometimes very useful. However, it is not always needed just to evaluate some N-dimensional function over a grid due to the array-broadcasting rules of Numeric and SciPy. If this is the only purpose for generating a meshgrid, you should instead use the function ogrid

where

$$\mathrm{Si}(x) = \int_0^x \sin\left(\frac{\pi}{2}t^2\right) dt.$$

is the Fresnel sine integral. Note that the numerically-computed integral is within 1.04×10^{-11} of the exact result — well below the reported error bound.

If the function to integrate takes additional parameters, it is possible to add them in the form of `args` and `kwargs` arguments to `quad`:

```
>>> print I(3)
(0.33333333325010883, 2.8604069919261191e-09)
>>> print I(2)
(0.49999999999857514, 1.8855523253868967e-09)
```

6.4.2. Gaussian quadrature (integrate.gauss_quadtol). A few functions are also provided in

where $H(x_0$

For curves in N-dime[sio[al space the function interpolate.splprep allows defining the curve

Figure 6.6.3. Example of two-dimensional spline interpolation.

in the data-set increases). The algorithms relating to B-splines in the signal-processing sub package assume mirror-symmetric boundary conditions. Thus, spline coe□cients are computed based on that assumption, and data-samples can be ompvered exactly from the spline coe□cients by assuming them

on N-dimensional arrays is signal.medfilt. A specialized version that works only for two-dimensional arrays is available as signal.medfilt2d.

6.9.4. Shifting.

6.9.5. Sample frequencies.

6.9.6. Hilbert transform.

6.9.7. Tilbert transform.

6.10. Linear Algebra

6.10.2.4. *Computing norms.* Matrix and vector norms can also be computed with SciPy. A wide

The following example Fig. 6.10.1 demonstrate the use of linalg.lstsq and linalg.pinv for solving a data-fitting problem. The data shown below were generated using the model:

$$y_i = c_1 e^{-x_i}$$

```
>>> print Vh
Matrix([[-0.2722, -0.6804, -0.6804],
        [-0.    , -0.7071,  0.7071],
```

il1bulu(t)s81(t)1(ra81(te)1(s)-390(t)1hu)1erhn281uerhu380duasii:BRA

```
import scipy as S
```

CHAPTER 7

3D visualization with MayaVi

Chapter contributed by Prabhu Ramachandran

7.1. Introduction

MayaVi is a scientific data visualizer. It is written in Python

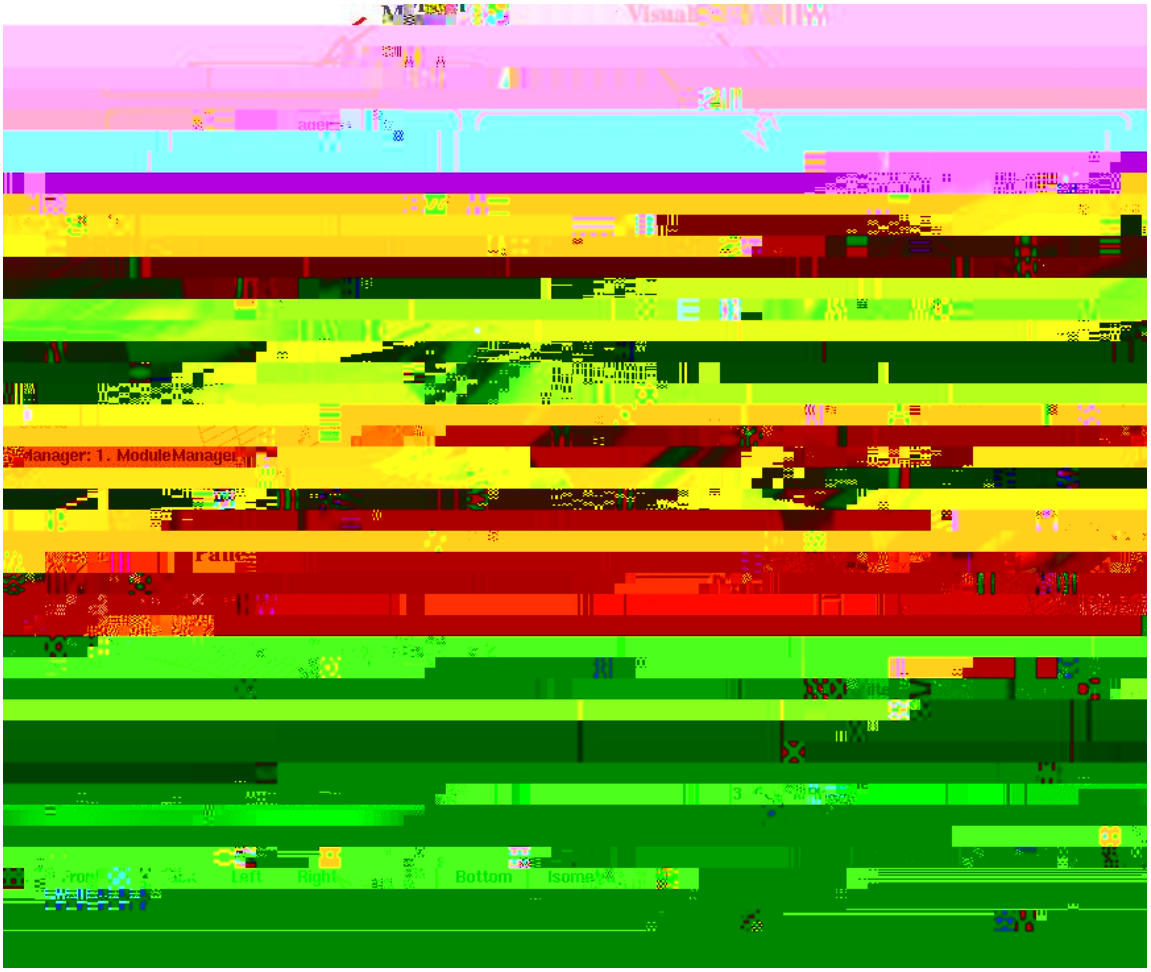

Figure 7.2.1. MayaVi window

Vi()1(u)1(alizatio)1(n)1(:)]T

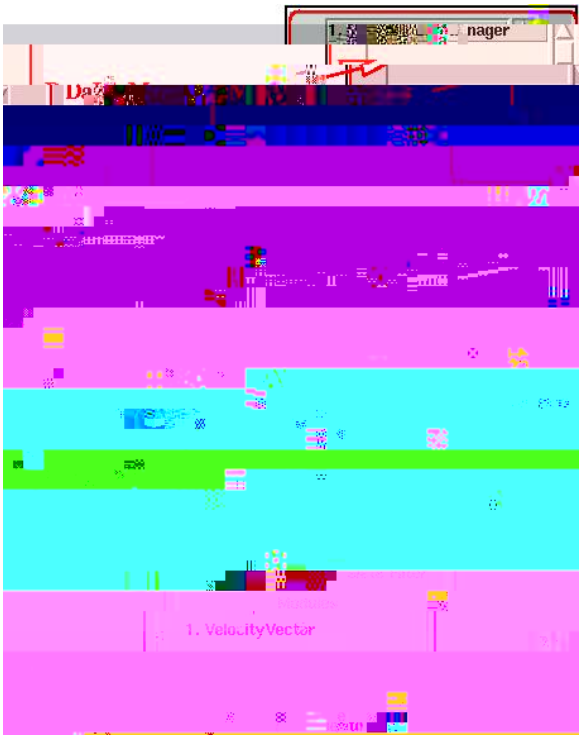

Figure 7.3.1. MayaVi control panel

MayaVi supports PLOT3D file format with binary structured grid data. The other PLOT3D datasets will not work due to limitations in VTK's vtkPLOT3DReader. Simple support for multi-block data is also incorporated.

EnSight data:

MayaVi supports EnSight data. EnSight6 and EnSightGold formats are supported. Only single parts are supported at this time.

In addition to this MayaVi allows one to import VRML2 files and 3D Studio files. Texturing is not yet

Only one scalar attribute and one vector attribute is supported at a given time. Each *DataVizManager*

This function reloads all the currently loaded Python modules. This is *very* useful while debugging a new feature for some module that one is creating. One may also see funny behavior for already instantiated objects.

7.3.7.4. *Help Menu.* The **Help** menu provides the following menu items.

About:
Displays a few details about MayaVi.

Users Guide:
Opens a web browser and displays this MayaVi users guide.

Home page:
Opens a web browser and displays the MayaVi home page.

choose a particular module or choose filtered data. The module will then generate text lables for the data in the chosen module and display it. The module provides many configuration

a list of matching classes is returned. Clicking on a class will pop up a window with the particular

CHAPTER 8

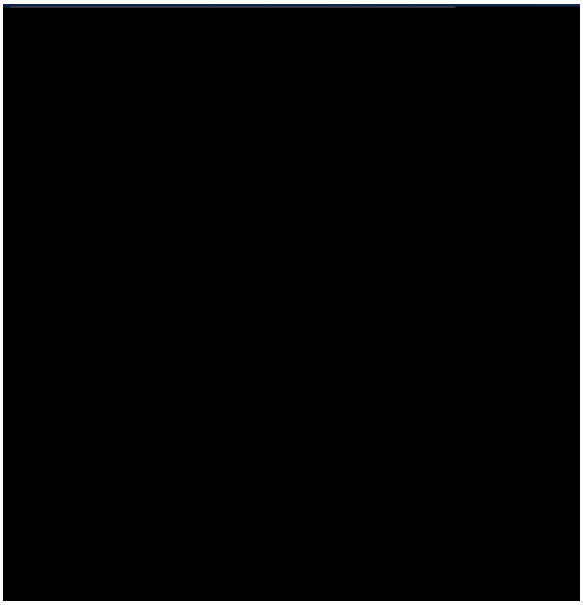

Figure 8.1.1. A Cube, brought to you by VTK

```
import vtk
```

CHAPTER 9

Interfacing with external libraries

9.1. weave

Below is a listing of examples of weave use. This needs a lot of cleaning, as some of this code is very old and doesn't actually run with current weave.

```
#!/usr/bin/env python
"""Simple examples of weave use.

Code meant to be used for learning/testing, not production.
```

```
"""
    inline(code,['num','mat
```

0a218.24274.676682.142emQg0600.40.80.2k00.40.80.2kQ001a274.676682.142emBTQF838.966Tf275.30642Td[(')]TJET0001218.24280.326682.142em
'

```
//a(2,2).imag
```

```
    print 'time C',tC
```

```
c       - array sizes can be au,o-de,ermined
Cf2py   in,eger in,en,(hide),depend(x):: nn=len(x)
```

```
setup(name = name,
      description  = description,
      author       = author,
```