Python and Scienti c Computing Notes

John D. Hunter
Fernando P

Contents

Chapter 1. Why python? **5**

Chapter 2. A whirlwind tour of the python?A

CHAPTER 1

Why python?

hello why python

CHAPTER 2

A whirlwind tour of python and the standard library

This is a quick-and-dirty introduction to the python language for the impatient scientist. There are many top notch, comprehensive introductions and tutorials for python. For absolute beginners, there is the *Python Beginner s Guide*.[1] The o cial *Python utori* can be read online[2] or downloaded[3] in a variety of formats. There are over 500 python tutorials collected online.[4]

There are also many excellent books. Targetting newbies is Mark Pilgric's *Di e into Python* which in available in print and for free online[5], though for absolute newbies even this may be too hard [7]. For experienced programmers, David Beasley's *Python Essenti eference* is an excellent introduction to python, but a bit dated since it only covers python2.1 [1]. Likwise Alex Martelli's *Python in Nutshe* is highly regarded and a bit more current { a 2nd edition is in the works[4]. And *he Python Coo oo* is an extremely useful collection of python idioms, tips and tricks [5].

But the typical scientist I encounter wants to solve a speci c problem, eg, to make a certain kind of graph, to numerically integrate an equation, or to t some data to a parametric model, and doesn't have the time or interest to read several books or tutorials to get what they want. d (to na)Tj 418.5897-11.08Td (Guide).T

```
}
```

2.2. Python is a calculator

Aside from mf daughter's solar powered cash-register calculator, Python is the only calculator I use. From the python shell, f ou can type arbitrarf arithmetic expressions.

```
>>> 2+2
4
>>> 2**10
1024
>>> 10/5
2
>>> 2+(24.3 + .9)/.24
107.0
>>> 2/3
0
```

The last line is a standard newbie gotcha { if both the left and right operands are integers, python returns an integer. To do oating point division, make sure at least one of the numbers is a oat

```
>>> 2.0/3
0.6666666666666663
```

The distinction between integer and oating point division is a common source of frustration among newbies and is slated for destruction in the mfthical Python 3000.[6] Since default integer division will be removed in the future, f ou can invoke the time machine with the `from __future__` directives; these directives allow python programmers today to use features that will become standard in future releases but are not included by default because they would break existing code. From future directives should be among the rst lines f ou type in f our python code if f ou are going to use them, otherwise they y(n)TjA035 0 Td 11.8667 0 Td (use)TTj 10.070d (74 0 Td (to)Tj 11.6206 0 Td 8 (da 6.95303 0 Td (ould)Tj 23.

```
>>> 2**200
1606938044258990275541962092341162602522202993782792835301376L
```

but python will blithely compute it and much larger numbers for you as long as you have CPU and memory to handle them. The integer type, if it over ows, will automatically convert to a python `long` **(as indicated by the appended** `L` **in the output above) and has no built-in upper bound on siFe, unlike C/C++ longs.**

Python has built in support for complex numbers. Eg, we can verify $i^2 = -1$

```
>>> x = complex(0,1)
>>> x*x
(-1+0j)
```

To access the real and imaginary parts of a complex number, use the `real` **and** `imag` **attributes**

```
>>> x.real
0.0
>>> x.imag
1.0
```

If you come from other languages like Matlab, the above may be new to you. In matlab, you might do something like this (> > is the standard matlab shell prompt)

```
>> x = 0+j
x =
   0.0000 + 1.0000i
>> real(x)
ans =
     0
>> imag(x)
ans =
     1
```

That is, in Matlab, you use a *function* **to access the real and imaginary parts of the data, but in python these are attributes of the complex object itself. This is a core featpre of python and other object oriented languages: an object carries its data and methods around with it. One might say: \a complex number knows it's real and imaginary parts" or \a complex number knows how to take its conjugate", you don't need external functions for these operations**

```
>>> x.conjugate
<built-in method conjugate of complex object at 0xb6a62368>
>>> x.conjugate()
-1j
```

On the rst line, I just followed along from the example above thiwith `real` **0.0 imagtyp**

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

and for the whole math library

```
>>> help(math)
Help on module math:

NAME
    math

FILE
    /usr/local/lib/python2.3/lib-dynload/math.so

DESCRIPTION
    This module is always available.  It provides access to the
    mathematical functions defined by the C standard.

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 complen object in the same way

```
>>> x =
```

```
>>> last = 'Hunter'
>>> last[0]
'H'
>>> last[1]
'u'
>>> last[-1]
'r'
```

To access substrings, or generically in terms of the sequence protocol, slices, you use a colon to indicate a range

```
# string slicing
>>> last[0:2]
'Hu'
>>> last[2:4]
'nt'
```

As this example shows, python uses \one-past-the-end" indexing when de ning a range; eg, in the range `indmin:indmax`**, the elemen**

in the spirit of

Tuples and lists have the same indexing and slicing rules as each other, and as string discussed above, because both implement the python sequence protocol, ADT.

default value for a function that can be overridden.

and this value can later be reused by other class methods (as it is in `__call__`) and it

CHAPTER 3

- **Dynamic object introspection. You can access do**

5.2. E ective interactive work

IPython has been designed to try to make interactive work as uid and e cient as possible. All

```
In [4]: %cd ..
/home/fperez
In [5]: del cd
```

At any time, your input history remains available. The `%hist` command can show you all previous input, without line numbers if desired (option `-n`) so you can directly copy and paste code either back in IPython or in a text editor. You can also sa

`%run` **also has special flags for timing the execution of your scripts (**`-t`**) and for executing them under the control of either Python's** `pdb` **debugger (**`-d`**) or profiler (**`-p`**). With all of these,** `%run` **can be used as the main**

CHAPTER 6

A tour of scipy

Purpose
Module overview
Some examples

CHAPTER

Figure 7.1.1. **A Cube, brought to you by VTK**

```
# Ready, set, go!
iren.Initialize()
iren.Start()
```

Exer

7.4. WORKING WITH MEDICAL WITH

W

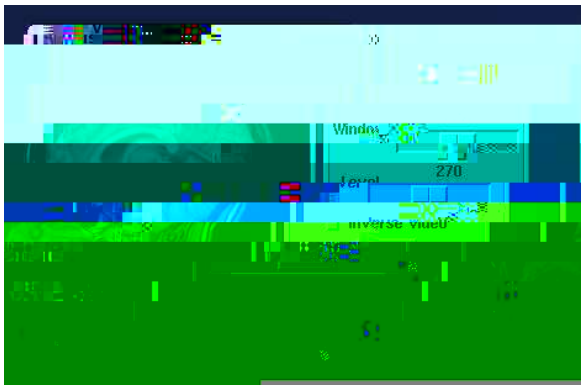

Figure 7.4.1. **A simple slice**

CHAPTER 9

CHAPTER 10

lyx examples

See a [2, 9]