Why Python?

because programming should be easy and fun

Yogesh Wadadekar

NCRA-TIFR

October 2014

What this course is about

- prerequisites

What this course is about

- prerequisites
- please interrupt and ask questions at any time.

What this course is about

- prerequisites
- please interrupt and ask questions at any time.
- programming background of the audience?

What we will teach you in this course

- the core Python language (5 lectures)
- how to use Python for numerical computing (1 lecture)
- how to use Python for making plots (1 lecture)
- how to use Python for astronomy specific computation using astropy. (1 lecture)
- how to use your existing Fortran or C code from within Python (1 lecture).

Why Python?

- A powerful, general purpose programming language, yet easy to learn. Strong, but optional, Object Oriented Programming support
- Very large user and developer community, very extensive and broad library base
- Very extensible with C, C++, or Fortran, portable distribution mechanisms available
- Free; non-restrictive license; open source
- fast becoming the standard scripting language for data analysis
- very powerful array processing capabilities (numpy)
- extensive documentation - Many books and on-line documentation resources available (for the language and its libraries and modules)

NCRA • TIFR

Why python?

- superb database interfaces to all popular databases.
- Clean code (very few non-Âalpha-Ânumerics)
- **forced indentation** (back to old Fortran?)
- concise
- great for large teams
- Plotting is easy (and trivial if you know matlab) using matplotlib
- Support for many widget systems for GUI development

Disadvantages of Python

- More items to install separately (eased by yum, apt-get, pip, easy_install)
- Scientific and numerical libraries not as mature as in Fortran
- but many Fortran libraries are wrapped: e.g. The NAG Library for Python, provides as a set of “Bindings” for use in conjunction with the NAG Library, with full access to the mathematical and statistical routines. Same for IMSL.
- Array indexing convention backwards, compared to Fortran
- Small array performance slower
- No standard GUI run/debug tool e.g. like Eclipse for Java
- Support for many widget systems (angst regarding which to choose)

NCRA • TIFR

Comparison with FORTRAN/C/C++

- huge amount of legacy code
- compilers highly optimized for excellent runtime performance

but...

- FORTRAN not really general purpose
- relatively primitive datatypes
- manual memory management
- slow edit/compile/test cycle

Comparison with IDL/Matlab

- Extremely popular
- Interactive, great visualization, good libraries

but...

- Not really general purpose
- Vendor lock-in
- fairly expensive, source code of core libraries not changeable.

Comparison with Perl

- shares many of Python's strengths

but...

- Write Once, Read Never
- just think and type, that's perl
- I think that "Just think and type, that's python" is more appropriate.

Python usage in optical astronomy

- STScI PyRAF (IRAF) + additional Python only routines
- ESO PyMIDAS (MIDAS)
- STScI PyFITS (access to FITS files)
- Astro-WISE (widefield imaging system)
- Pyephem - solar system ephemeris
- LSST will use Python/C++

Python usage in Radio astronomy

- CasaPy (Casa) - AIPS++, default system for EVLA and ALMA data analysis.
- ParselTongue - call AIPS tasks from Python
- PYGILDAS (GILDAS) - IRAM data analysis software ported to Python
- BoA (Bolometer Analysis Package) for LABOCA on APEX and other bolometers
- APECS (APEX control software)
- KAT-7 CMS is in Python
- Presto - pulsar search and analysis suite; most recent routines in Python

Methodology of a successful programmer

- prepare a full working prototype of any application in Python, without worrrying about speed

Methodology of a successful programmer

- prepare a full working prototype of any application in Python, without worrrying about speed
- *profile* the prototype

Methodology of a successful programmer

- prepare a full working prototype of any application in Python, without worrrying about speed
- *profile* the prototype
- identify the slow parts of the Python code and rewrite them in a compiled language like C/C++ for maximum speed.

Methodology of a successful programmer

- prepare a full working prototype of any application in Python, without worrrying about speed
- *profile* the prototype
- identify the slow parts of the Python code and rewrite them in a compiled language like C/C++ for maximum speed.
- write wrappers to interface the new compiled code with the Python prototype. Repeat until required optimisation is achieved.

Methodology of a successful programmer

- prepare a full working prototype of any application in Python, without worrrying about speed
- *profile* the prototype
- identify the slow parts of the Python code and rewrite them in a compiled language like C/C++ for maximum speed.
- write wrappers to interface the new compiled code with the Python prototype. Repeat until required optimisation is achieved.
- If you are talking to devices - write a simulator first.

NCRA • TIFR

Python for scientists and engineers

- full featured, high level programming language
- very easy to learn – National Mission on Education through ICT is sponsoring a large program (crores of rupees) to develop computer education materials in Python for school and college students (http://python.fossee.in/).
- powerful text processing capabilities - many sysadmins are adopting it.
- powerful interfaces to almost any database
- web-friendly language - many frameworks available e.g. Django, Zope, CherryPy, Trac for website CMS, wikis etc.
- CERN's INDICO conference management system is all Python.
- good numerical computation capabilities
- good plotting capabilities

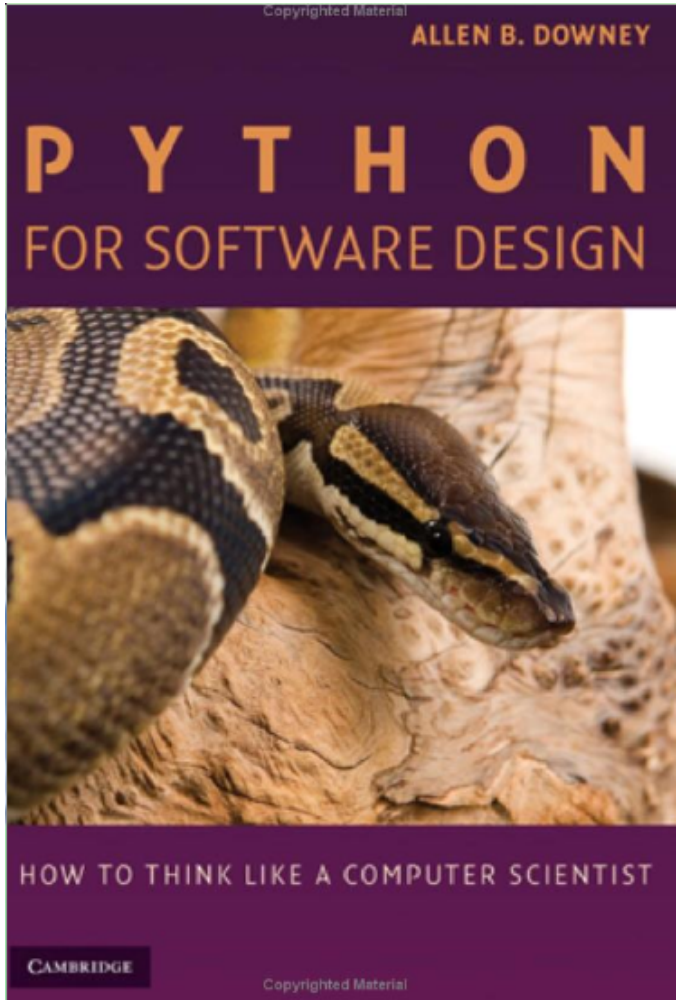
Python extensively used by Google

- The Google build system is written in python. All of Google's corporate code is checked into a repository and the dependency and building of this code is managed by python.
- Packaging. Google has an internal packaging format like RPM. These packages are created using python.
- Binary Data Pusher. This is the area where Alex Martelli is working, on optimizing pushing bits between thousands of servers
- Production servers. All monitoring, restarting and data collection functionality is done with python
- Reporting. Logs are analyzed and reports are generated using Python.
- A few services including code.google.com and google groups run on Python. Most other front ends are in C++ (google.com) and Java (gmail). All web services are built on top of a optimized http server wrapped with SWIG.

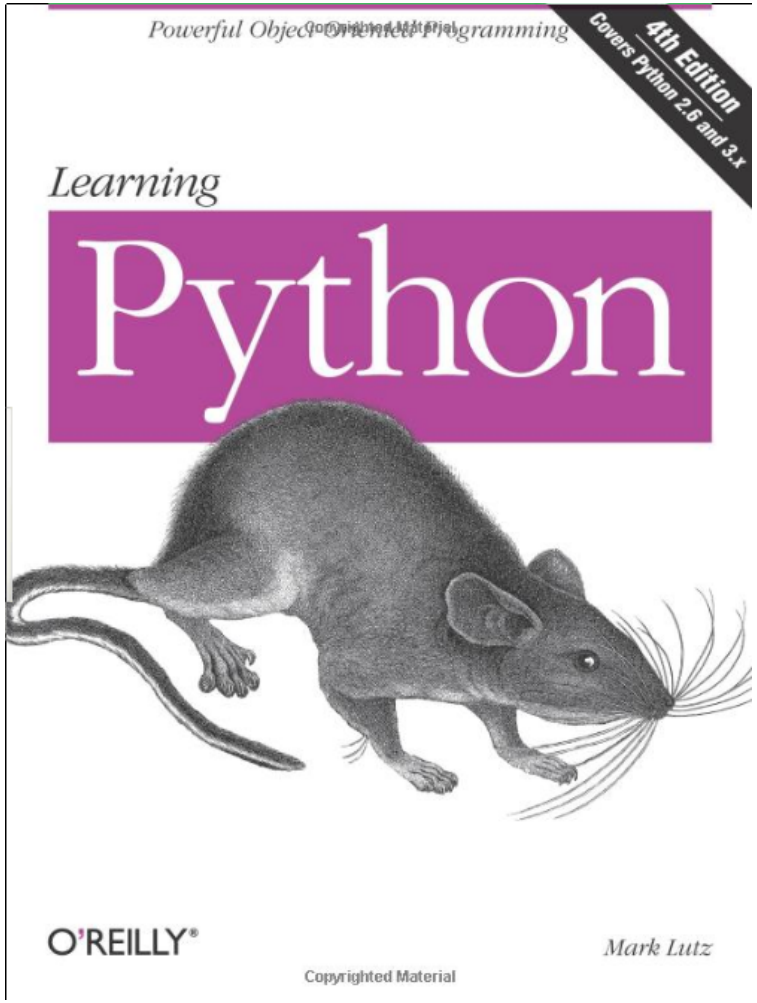
Our course focused on using Python

for data analysis in the science and engineering domain with special emphasis on astronomy.

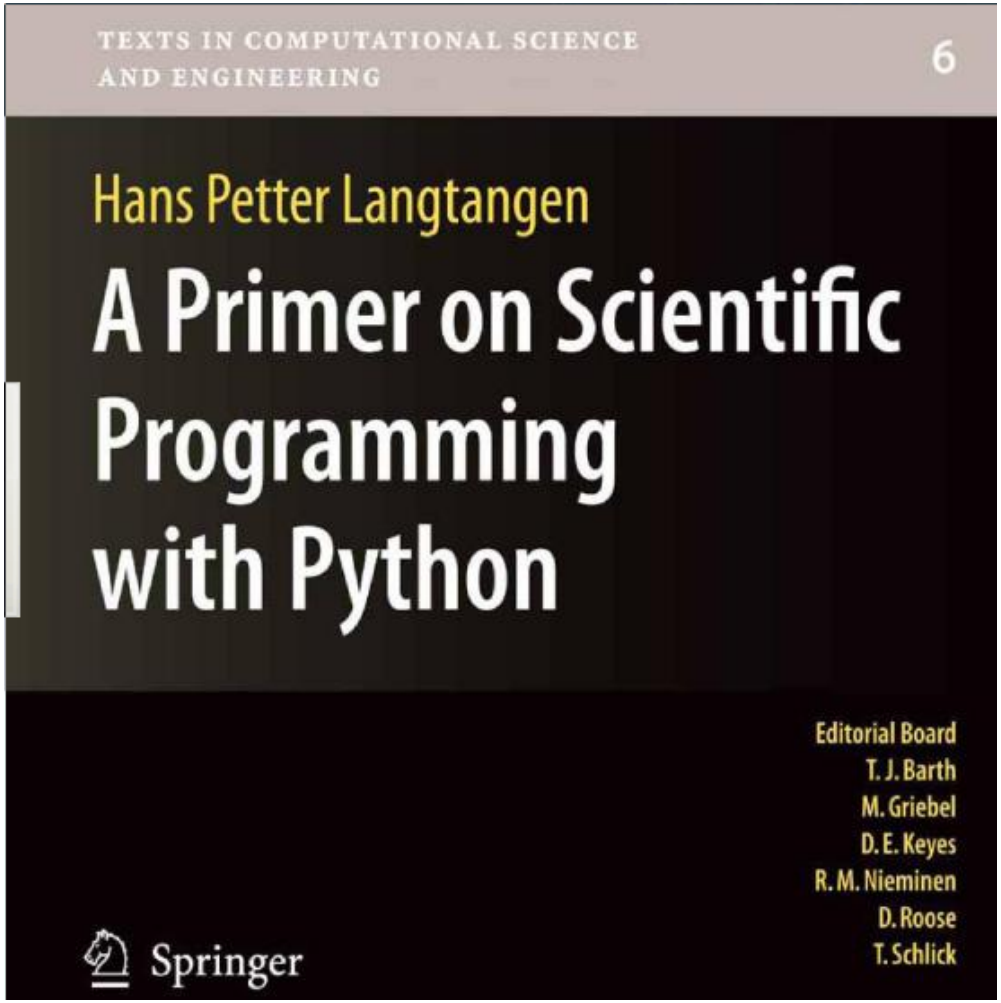
If you are an absolute beginner to programming



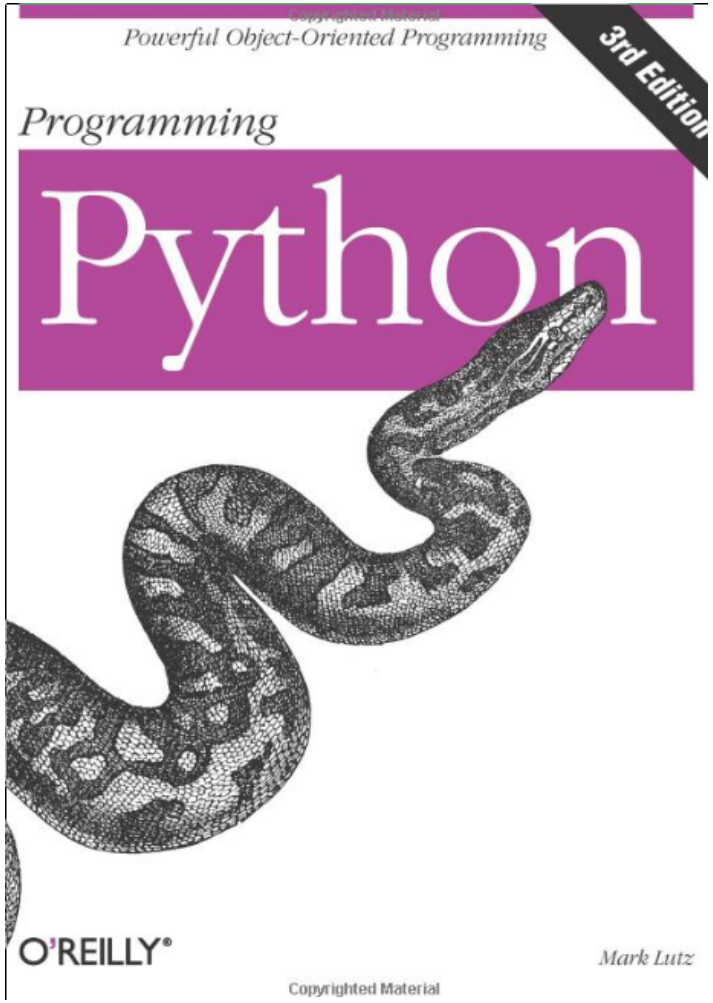
Standard book for beginners - Lutz & Ascher



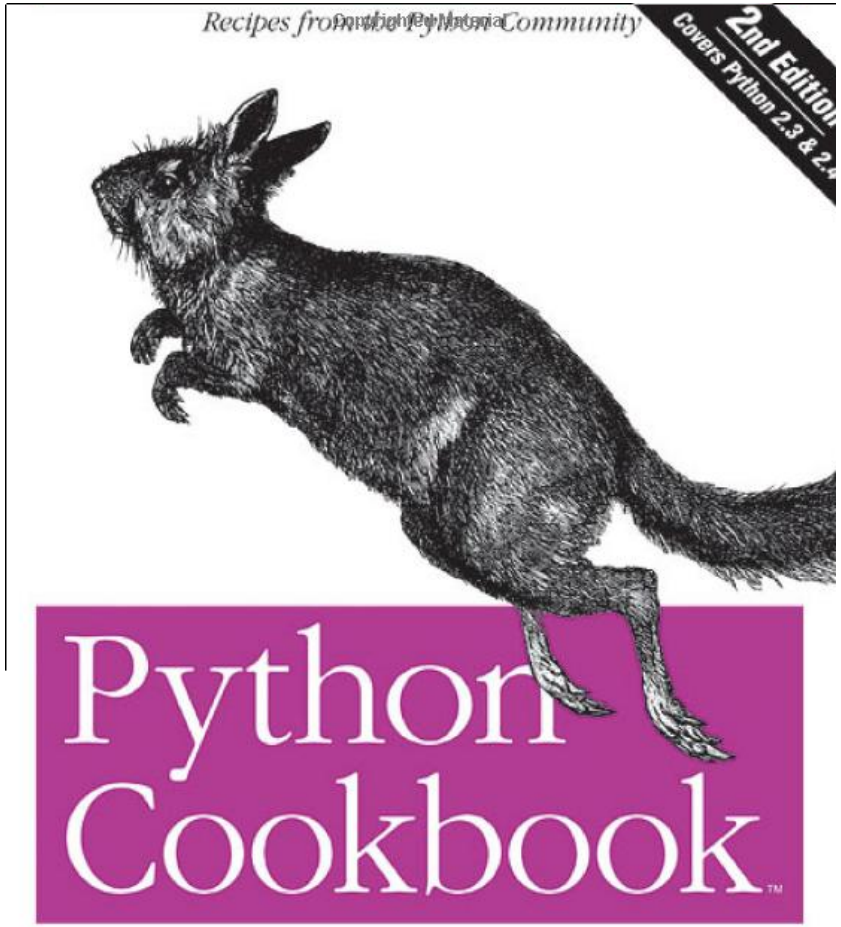
A new primer by Langtangen



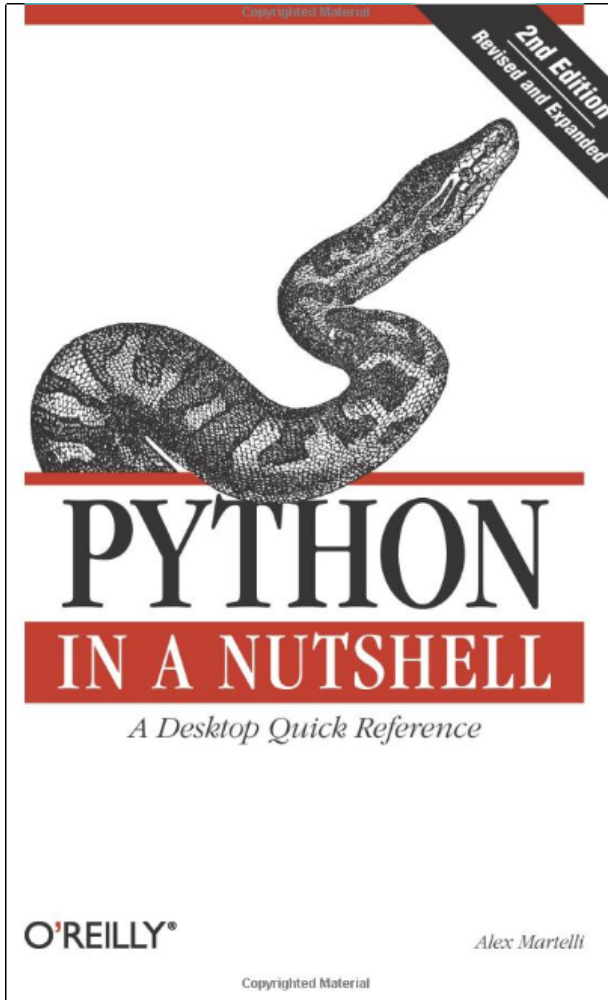
Book for intermediate level



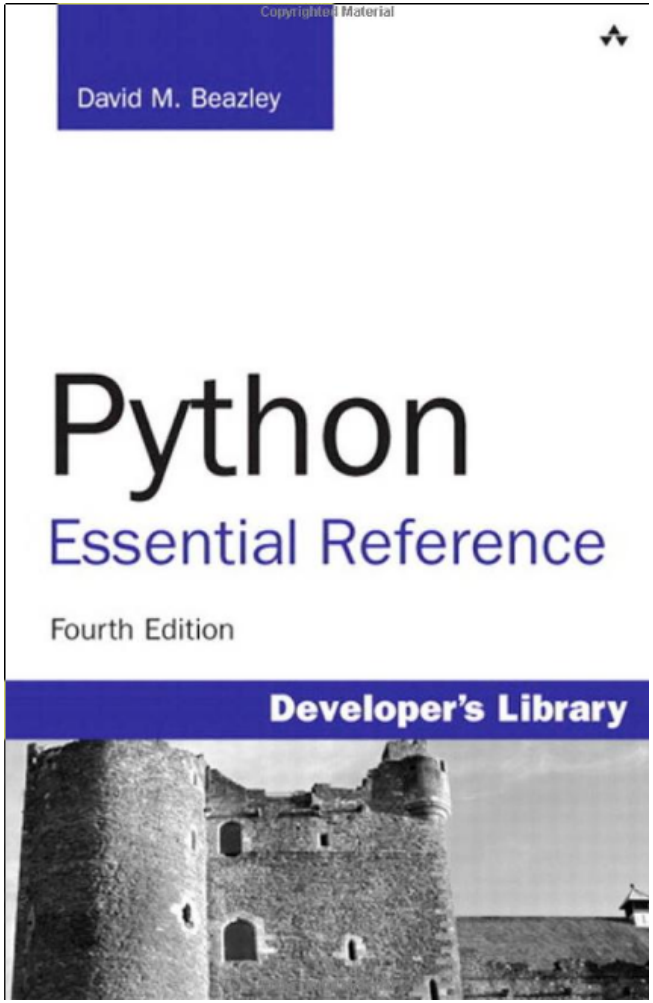
Python Cookbook contains many recipes



A Python quick reference by Martelli

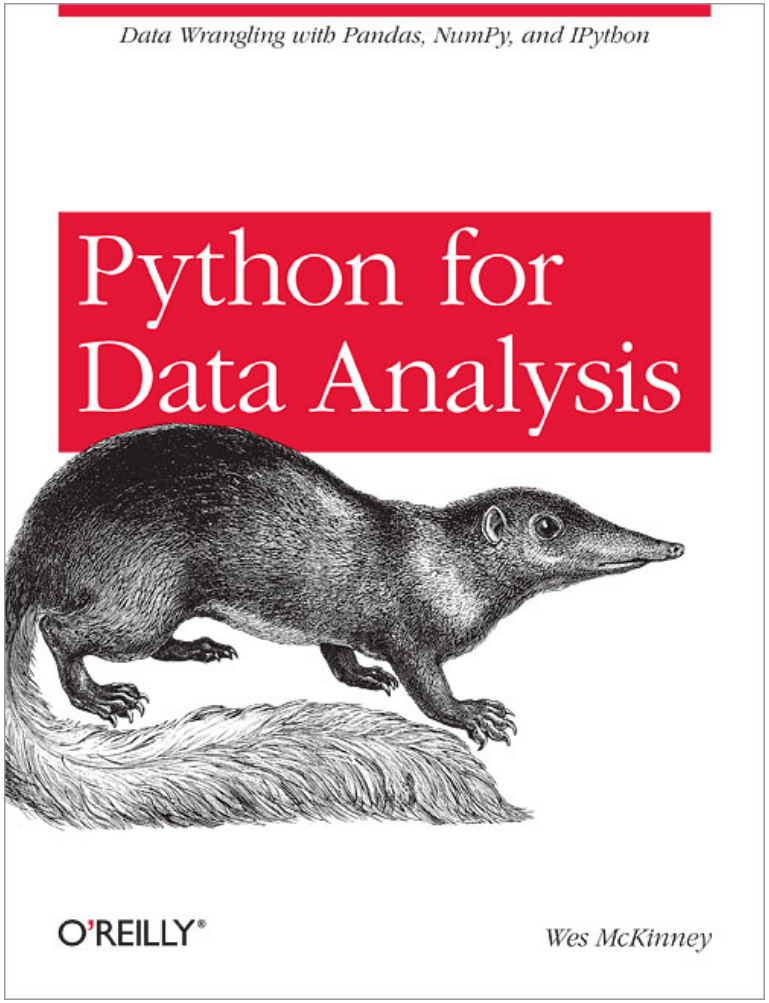


Another reference book by Beazley



Python for Scientific Computing by Langtangen

Python for Data Analysis



Plenty of other books are available

Remember that these books have gone through many editions. Be sure to get the latest one.
Search for “Python programming” on amazon.in throws up 1005 items in the Books department.
Having said this, I have not bought a Python book yet. Many of these books are available in the NCRA library. Also, a lot of excellent documentation is online.

Online Material

- www.python.org

Online Material

- www.python.org
- Start with the Python tutorial - http://docs.python.org/tutorial/ we will cover all of it and more in this course.

Online Material

- www.python.org
- Start with the Python tutorial - http://docs.python.org/tutorial/ we will cover all of it and more in this course.
- SciPy conferences - http://conference.scipy.org - lots of interesting talks (some with video versions)

Online Material

- www.python.org
- Start with the Python tutorial - http://docs.python.org/tutorial/ we will cover all of it and more in this course.
- SciPy conferences - http://conference.scipy.org - lots of interesting talks (some with video versions)
- Also check out the excellent Python programming courses on coursera and eDx.
- http://python.fossee.in also has a number of simple tutorials.

Python on MS Windows

Full distributions provided by Enthought and ActiveState. Enthought distribution (EPDFree) is better for scientific/technical computing.

Python on Mac

First install Xcode and then EPDFree

Python on Linux

Almost always, Python will be already installed. Type `python` in a terminal to check it out.
On Redhat like distributions, the installer – anaconda – is written in Python.

GUIs for python development

IDLE is one that is distributed with Python.
emacs is a very good IDE if you willing to learn how to use it or know it already.
Numerous other free and commercial IDEs are available. A comprehensive list is available at
http://wiki.python.org/moin/PythonEditors
I like Pycharm (www.jetbrains.com/pycharm/)

Python Version 2 or 3?

Python 3 is newer but Python 2 has more existing third party software. For this reason, in this course, I will use Python 2. But almost everthing we use should be usable in Python 3, as is.

Hello World program

```
$ python -c 'print ("Hello World")'
Hello World
```

Starting Python

simply type *python* at the command prompt

```
$ python
Python 2.7.6 (default, Mar 22 2014, 22:59:56)
[GCC 4.8.2] on linux2
Type "help", "copyright", "credits" or "license" for
more information.
>>>
```

Ipython - an enhanced python shell

simply type *ipython* at the command prompt. If it is not installed `sudo apt-get install ipython`

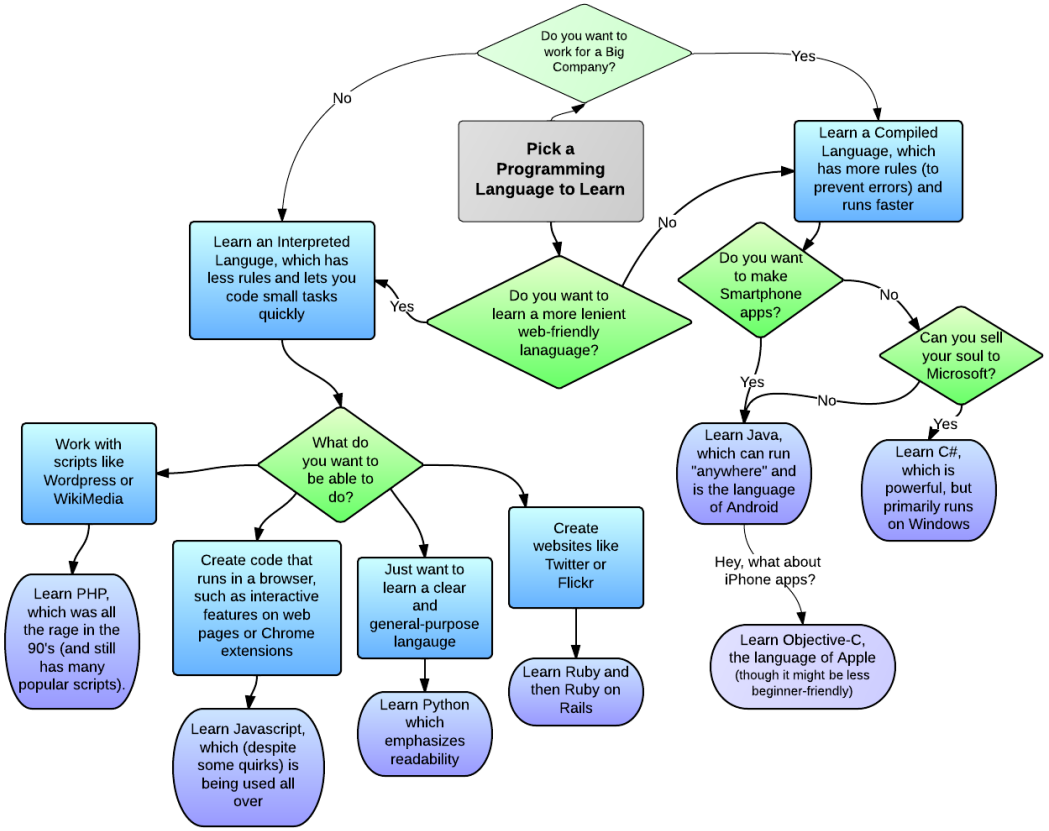
```
IPython 1.2.1 - An enhanced Interactive Python.
?  -> Introduction and overview of IPython's
features.
%quickref -> Quick reference.
help -> Python's own help system.
object?  -> Details about 'object'.  ?object also
works, ??  prints more.
In [1]:
```

Ipython is the shell for astronomy analysis packages - casapy and Pyraf.

Ipython notebook

Demo

Programming language flowchart



Source: zappable.com

The Zen of Python, by Tim Peters *import this*

Beautiful is better than ugly.
Explicit is better than implicit.
Simple is better than complex.
Complex is better than complicated.
Flat is better than nested.
Sparse is better than dense.
Readability counts.
Special cases aren't special enough to break the rules.
Although practicality beats purity.
Errors should never pass silently.
Unless explicitly silenced.
In the face of ambiguity, refuse the temptation to guess.
There should be one– and preferably only one –obvious way to do it.
Although that way may not be obvious at first unless you're Dutch.
Now is better than never.

The Zen of Python, by Tim Peters *import this*

Although never is often better than *right* now.
If the implementation is hard to explain, it's a bad idea.
If the implementation is easy to explain, it may be a good idea.
Namespaces are one honking great idea – let's do more of those!

That Python feeling