JOHNNY WEI-BING LIN

A Hands-On Introduction to Using Python in the Atmospheric and Oceanic Sciences

HTTP://WWW.JOHNNY-LIN.COM/PYINTRO

2012

Who would *not* want to pay money for this book?: if you do not need a black-and-white paper copy of the book, a color PDF copy with functional hyperlinks, have limited funds, or are interested in such a small portion of the book that it makes no sense to buy the whole thing. The book's web site (http://www.johnny-lin.com/pyintro) has available, for free, PDFs of every chapter as separate files.

Who would want to pay money for this book?: if you want a black-and-white paper copy of the book, a color PDF copy with functional hyperlinks, or you want to help support the author financially. You can buy a black-and-white paper copy of the book at http://www.johnny-lin.com/pyintro/buypaper.shtml and a hyperlink-enabled color PDF copy of the book at http://www.johnny-lin.com/pyintro/buypdf.shtml.

A special appeal to instructors: Instruction at for-profit institutions, as a commercial use, is not covered under the terms of the CC BY-NC-SA, and so instructors at those institutions should not make copies of the book for students beyond copying permitted under Fair Use. Instruction at not-for-profit institutions is not a commercial use, so instructors may legally make copies of this book for the students in their classes, under the terms of the CC BY-NC-SA, so long as no profit is made through the copy and sale (or Fair Use is not exceeded). However, most instruction at not-for-profit institutions still involves payment of tuition: lots of people are getting paid for their contributions. Please consider also paying the author of this book something for his contribution.

Chapter 10

What Next?

Congratulations! You've taken the first step into an amazing and exciting new world. Python, with its modern computer science methods, enormous diversity of packages, and clear syntax, will enable you to write programs you never would have dreamed of before, to investigate science questions that would have been nearly impossible to address using traditional tools. As we wrap up this book and course, your intrepid tour guide of the Python world bids you a fond farewell, but not before suggesting some Python topics to address next, point out some Python packages that will be of interest to AOS users, and provide a list of references that you will find helpful as you continue to build-up your Python chops.

10.1 What Python topics would be good to cover next?

As you've gone through the book, you probably have come up with a personal list of topics you'd like to do more study on. Here is my list of topics for you to study next. See the references listed in Section 10.3 for more on most of these topics.

More NumPy routines and SciPy: We've only touched the surface of the calculations you can make with NumPy, and the SciPy package (imported by the `import scipy` command) offers even more mathematical and scientific packages.

Exception handling: We introduced exception handling in Section 3.16, but the real power and flexibility of exception handling is unleashed when you create your own special exception classes. This requires using inheritance (a topic outside the scope of this book), but enables you to test for and gracefully handle error conditions specific to your application.

Documentation: We briefly discussed the most basic Python element to code documentation (aside from comment lines), the docstring, on p. 62. Besides docstrings, however, a number of packages exist to generate user guides, manuals, and API documentation. Two I like are Epydoc (http://epydoc.sourceforge.net) and Sphinx (http://sphinx.pocoo.org).

Unit testing: Many of the programs AOS users write are quick-and-dirty programs written by one individual with little to no documentation and testing. Unfortunately, as decades of software engineering experience has shown, this results in fragile, buggy code, and a lot of reinventing the wheel. Software testing is a *proven* way of increasing code quality and reliability (e.g., Basili and Selby, 1987). Python makes it easy for us to write unit tests, i.e., tests of small portions of code, through the unittest and pytest packages. The unittest package comes standard with every Python installation; pytest is available at http://pytest.org.

Platform-independent operating system commands: Python has been ported to nearly every operating system imaginable and through it offers the possibility of "write once, run anywhere" code. The os module (imported by `import os`) enables such platform independence by wrapping operating system commands in a Python interface. A submodule of os called path enables platform-independent handling of directory paths (it is imported by the `import os.path` command).

Environment customization: The sys module gives you access to variables that interact with the Python interpreter, such as the search path for Python modules (which, if you import sys by `import sys` is stored in the module attribute `sys.path`).

Wrapping Fortran routines: As cool as Python is, there is no way we can do our work without Fortran. There are too many lines of legacy code, and sometimes we need the speed of compiled code. Wouldn't it be nice if we could use Fortran routines from within Python? With f2py, this is not only possible but easy-peasy! And, if you have NumPy installed, you already have f2py. See http://www.scipy.org/F2py for more information.

Class inheritance: I mentioned this above when talking about exception handling, but this OOP concept has much more applicability than just in dealing with exceptions. Inheritance enables you to even more easily push functionality to the lowest appropriate level.

Advanced visualization: As I said in the end of Ch. 9, Python does not have only one visualization package but many, each with their own domains of competence. Check out PyNGL, `vcs`, VPython, etc.

10.2 Some packages of interest to AOS users

There is no possible way for me to list all the packages available for Python (as of August 2012, PyPI listed over 23,000 packages),[1] nor even all the packages of possible interest to AOS users. Here I mention just a few, organized by tasks AOS users conduct.

Data analysis: UV-CDAT is a veritable Swiss Army knife for climate data analysis. It includes specialized routines to deal with dates and times, spatial domains and regridding, climate model history files, OpenDAP, visualization, etc.; see http://uv-cdat.llnl.gov. UV-CDAT is an outgrowth of an older application CDAT; the CDAT pages have more documentation and are at http://www2-pcmdi.llnl.gov/cdat. (A lighter version that only has the core CDAT packages, Cdat-lite, is also available: see http://proj.badc.rl.ac.uk/cedaservices/wiki/CdatLite.) Finally, pandas is an interface on top of NumPy that enables you to reference subarrays using string labels (like dictionaries) and easily deal with missing values: see http://pandas.pydata.org.

PyGrADS provides a Python interface to the gridded data analysis and visualization system GrADS (http://opengrads.org/wiki/index.php?title=Python_Interface_to_GrADS).[2] PyNIO provides Python bindings to file i/o routines for formats of interest to AOS users; see http://www.pyngl.ucar.edu.

Visualization: PyNGL provides Python bindings to the NCAR Graphics Language; see http://www.pyngl.ucar.edu. PyGrADS, mentioned earlier, also helps AOS users visualize their data.

Mathematical and scientific functions: As I mentioned in Section 10.1, SciPy provides numerous mathematical and scientific routines (http://www.scipy.org). SAGE is another set of mathematical and scientific libraries (http://www.sagemath.org). Finally, RPy gives a Python interface to the powerful statistical language R (http://rpy.sourceforge.net).

GIS: ArcGIS scripting can be done in Python; see http://www.esri.com/software/arcgis. Other packages using Python that enable GIS manipulation include: PyKML (http://pypi.python.org/pypi/pykml), OpenClimateGIS (https://github.com/tylere/OpenClimateGIS), and GDAL (http://trac.osgeo.org/gdal/wiki/GdalOgrInPython).

Webservices: Python has a number of packages that enable you to do webservices. One of the most comprehensive is the Twisted package (http://twistedmatrix.com/trac). CherryPy is another, more accessible, package (http://cherrypy.org).

[1]http://pypi.python.org/pypi (accessed August 17, 2012).
[2]PyGrADS is part of the OpenGrADS project.

The PyAOS website maintains a list of packages of interest to AOS users. See: http://pyaos.johnny-lin.com/?page_id=20.

10.3 Additional references

Throughout the book, especially in this chapter, I've referenced a variety of resources, most of them online. In this section, I list places to look for more general help in using Python or resources that address specific topics that don't fit anywhere else. The Bibliography gives the full citation for books mentioned in this section.

Transitioning from Matlab/IDL to Python: Thankfully, you can find a number of equivalence sheets online for Matlab to Python and IDL to Python.[3] Of course, the languages aren't one-to-one identical to Python, but these sheets can still help with the transition.

Tutorials: There are a bunch of great Python tutorials, both online as well as in print. Perhaps the place to start is the standard Python Tutorial (http://docs.python.org/tutorial), though it is written more for a computer science audience rather than an AOS audience. Michael Williams's Handbook of the Physics Computing Course (http://pentangle.net/python/handbook) is a nice tutorial. Though it does not cover i/o and is geared for an audience of physicists, it is accessible to new Python users of all kinds. We've also listed a number of tutorials at PyAOS, both those that address Python for general science users (http://pyaos.johnny-lin.com/?page_id=215) and AOS-specific tutorials (http://pyaos.johnny-lin.com/?page_id=217).

Reference manuals: I really like *Python in a Nutshell* (Martelli, 2006). It's perhaps best known for providing a handy (but long) list of functions and options, but its explanations of key Python concepts, while terse, nonetheless are clear and illuminating. However, *Nutshell* is not a good resource for newbies; it assumes a formidable level of prior knowledge about computer science concepts.

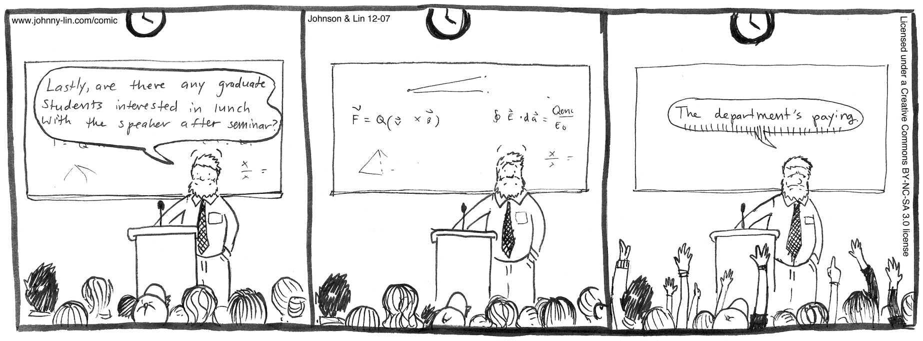
The Python Language Reference (http://docs.python.org/reference) is the definitive guide to Python syntax, but I think its language and explanations are even farther removed from the world of non-computer scientists. I've found Martelli's descriptions and organization to be more helpful and understandable, as a result. The Python Standard Library (http://docs.python.org/library) documentation describes all the built-in functions and modules that

[3]For Matlab to Python, see http://www.scipy.org/NumPy_for_Matlab_Users. For IDL to Python, see https://www.cfa.harvard.edu/~jbattat/computer/python/science/idl-numpy.html. Both are accessed August 17, 2012.

come with Python. The os, sys, and unittest modules mentioned earlier are described here.

Finally, the NumPy and SciPy online documentation (http://docs.scipy.org/doc) is a must-bookmark site. Keep the NumPy list of functions in easy reach: http://www.scipy.org/Numpy_Functions_by_Category.

Other texts: It seems like everyday brings a new crop of books on Python, geared generally for data analysts and scientists. I haven't had the chance to look through most of these resources, so my recommendation to you would be to search a phrase like "python scientific programming" in your favorite bookseller's search engine.


www.johnny-lin.com/comic
Lastly, are there any graduate students interested in lunch with the speaker after seminar?
Johnson & Lin 12-07
The department's paying.
Licensed under a Creative Commons BY-NC-SA 3.0 license

10.4 A parting invitation

Come join PyAOS!

Throughout this book, I've mentioned the PyAOS website. As we end our time together, let me extend an invitation to you, gentle reader, to come and join PyAOS. Through our website, blog, and mailing list, we aim to support atmospheric and oceanic science users of Python: to help new users learn the language and experienced users to share with one other the cutting-edge work going on with the language. We're online at: http://pyaos.johnny-lin.com. Hope to see you there, soon! Lunch is on us ☺!