Johnny Wei-Bing Lin

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

http://www.johnny-lin.com/pyintro

2012

 Printed version: ISBN 978-1-300-07616-2. PDF versions: No ISBNs are assigned.

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.

Regardless of whether or not you paid money for your copy of the book, you are free to use any and all parts of the book under the terms of the CC BY-NC-SA.

Chapter 1

What Is and Why Python?

1.1 Python: The good and the bad

So, what's with all the fuss about Python? If you're reading this book, you might have heard about Python from a co-worker who swears by it, heard a reference to the language in a talk at a conference, or followed a link from a page on scientific computing. When you've asked others about Python, they might have thrown out words like "object-oriented," "interpreted," or "open-source." What does all this mean?

This book answers that question, from the standpoint of a researcher in the atmospheric or oceanic sciences. That doesn't mean, however, there isn't a shorter answer to the question. Here is one summary list of the attributes and features of Python:

- Structure: Python is a **multi-paradigm language**, and can be used for scripting, **procedural programming**, as a fully native object-oriented (OO) language, and as a functional language.

- Interpreted: Python is loosely or **dynamically typed** and interactive. There is no separate compiler but rather commands typed into the interpreter are automatically compiled, linked (as needed) and executed.

- Data structures: Python has a robust built-in set of data types, and users are free to define additional structures.

- Syntax: Easy to read and includes an array syntax that is similar to Matlab, IDL, and Fortran 90 (no loops!).

- Platform independent, open-source, and *free!*

"Python is executable pseudocode."

But what do these features allow you to do? Here is where Python users start waxing enthusiastically. First, because of Python's concise but natural syntax, for both arrays and non-arrays, programs are exceedingly clear and easy to read; as the saying goes, "Python is executable pseudocode. Perl is executable line noise."[1] Second, because the language is interpreted, development is much easier; you do not have to spend extra time with manipulating a compiler and linker. Third, the object-orientation makes code more robust/less brittle, and the built-in set of data structures are very powerful and useful (e.g., dictionaries). Fourth, Python is designed with a built-in namespace management structure which naturally prevents variable and function collisions. In contrast with linking multiple Fortran libraries, where you can easily overwrite a function from one library with a function from another, in Python you have to work at causing such a collision. Finally, Python's open-source pedigree added with a large user and developer base in industry, as well as science—institutions supporting AOS Python include Lawrence Livermore National Laboratory (LLNL)'s Program for Coupled Model Diagnostics and Intercomparison (PCMDI) and National Center for Atmospheric Research (NCAR)'s Computational Information Systems Laboratory (CISL)—means that your programs can take advantage of the tens of thousands of Python **packages** that exist. Multiple visualization packages are available, some numerical libraries, packages that deliver tight interconnects with compiled languages (Fortran via f2py and C via SWIG), memory caching, webservices, graphical user interface (GUI) programming, etc. You are not limited to only what one vendor can provide, or even what only the scientific community can provide!

Python's disadvantages.

To be fair, Python has real disadvantages, including that pure Python code runs much slower than compiled code, there are comparatively few scientific libraries compared to Fortran, and documentation and support for new science users is relatively sparse. There are tools to overcome the speed penalty, the collection of scientific libraries is growing, and science support resources are becoming more robust (which this book hopefully contributes to), but these are real issues. For many, if not most, AOS applications, however, the strengths of Python outweigh the weaknesses.

1.2 Examples of AOS uses for Python

But this all sounds kind of abstract: what are some examples of what Python can do for AOS users? Figure 1.1 shows examples of typical AOS visual-

[1] I don't know who first said this, but you can find one instance of this quote at http://mindview.net/Books/Python/ThinkingInPython.html (accessed March 14, 2012).

ization tasks (skew-T and meteograms), using the PyNGL package, which implements all the primitives (and some of the higher-level functions) of the NCAR Graphics Language (NGL). Figure 1.2 shows screenshots of an application (WxMAP2) written in Python that visualizes and delivers weather maps of numerical weather prediction model results. And Figure 1.3 shows an application written in Python (VisTrails) that manages provenance in the context of geoscientific analysis and visualization: VisTrails enables you to analyze and visualize a dataset while at the same time keeping a record of the operations you did. As these examples show, Python can be used for nearly any analysis and visualization task you would want to do in AOS research and operations.

When Meteorologists Play With Their Food

As neat as these examples are, however, the greatest advantage of Python to AOS work (in my opinion) is how it enables one to have a truly unified workflow: analysis, visualization, and workflow management are all (potentially) integrated together. Figure 1.4 illustrates this potential. As shown in the figure, the problem encountered by the Northeast Regional Climate Center (NRCC) was how to unify the many different components of the Applied Climate Information System: data ingest, distribution, storage, analysis, web services (tasks and formats in black). The traditional solution would be to cobble together a crazy mix of shell scripts, compiled code, makefiles, Matlab/IDL scripts, and a web server. (And when you think of it, most AOS workflows are like that; a crazy mix of tools that talk to each other through

Python enables a unified workflow.

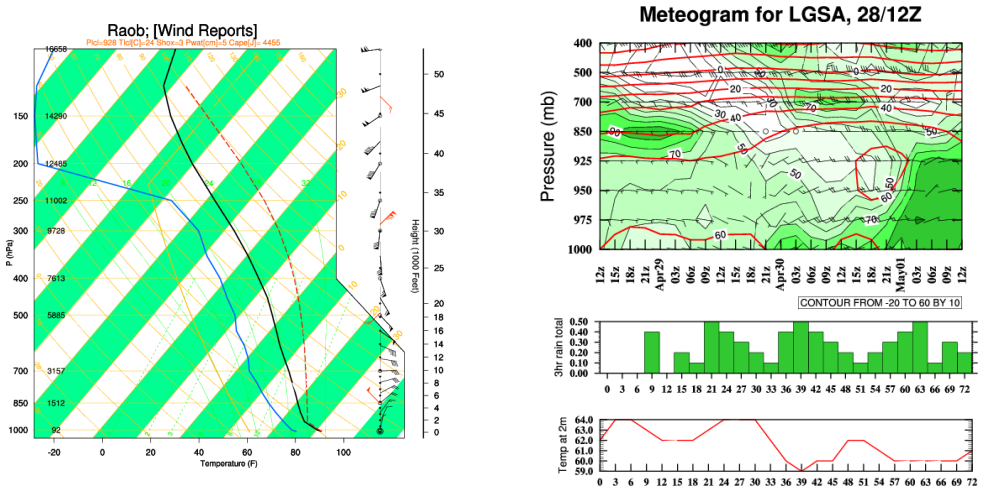

Figure 1.1: Visualization of a skew-T plot and a meteogram using PyNGL. These plots are taken from the PyNGL website http://www.pyngl.ucar.edu. See http://www.pyngl.ucar.edu/Examples/gallery.shtml for the source code to make the plots. Plots are copyright, University Corporation for Atmospheric Research. Graphics were generated with PyNGL, developed at NCAR and sponsored by NSF. Used by permission.

text pipes or files.) NRCC's solution: Do it all in Python (package names in red), and the resulting single environment of shared state created a more powerful, flexible, and maintainable system than would otherwise have been possible.

So, this is why I'm so excited about Python and why I wrote this book! Python is a flexible, powerful, open, and free language whose native object-oriented structure permits more robust programs to be written. The result is a language that enables better atmospheric and oceanic sciences to be done more easily at less cost, both in time and money. The bottom line is that Python enables me to do my science more easily and reliably; how cool is that?

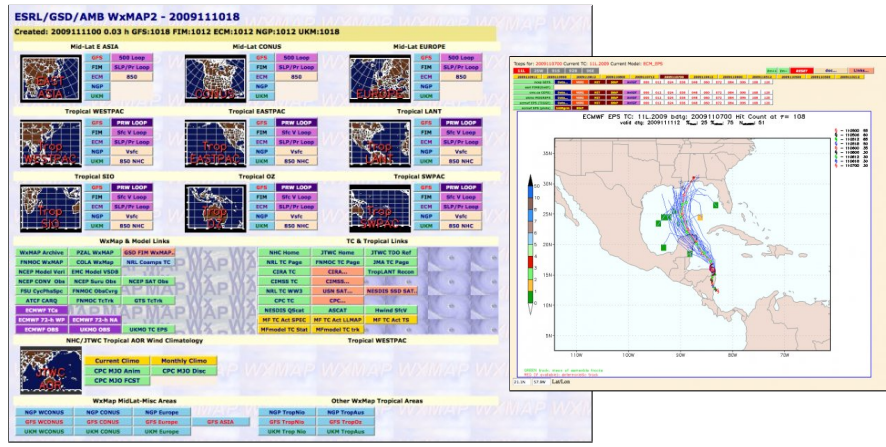

Figure 1.2: Screenshots taken from the WxMAP2 package web site. See: http://sourceforge.net/projects/wxmap2. These screenshots are by Michael Fiorino (NOAA Earth System Research Laboratory, Boulder, CO.) and are used by permission.

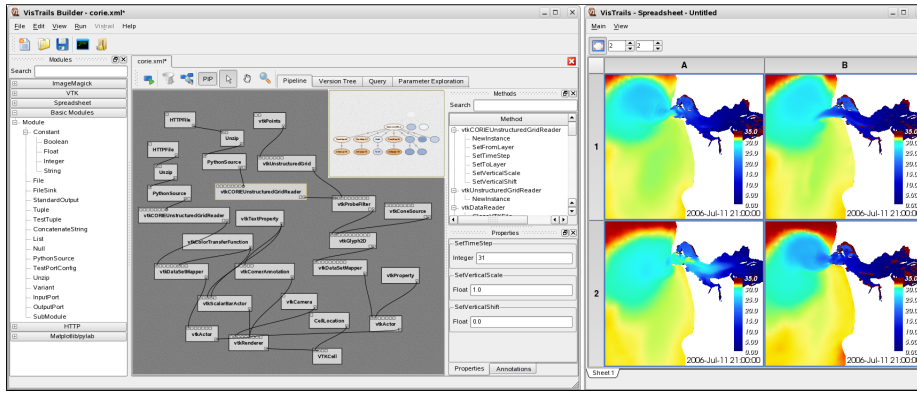

Figure 1.3: Session of the VisTrails visualization and data workflow and provenance management system; salinity data in the Columbia River estuary is graphed. See: http://www.vistrails.org/index.php?title=File:Corie_example.png&oldid=616. The screenshot is by Steven Callahan and is used by permission.

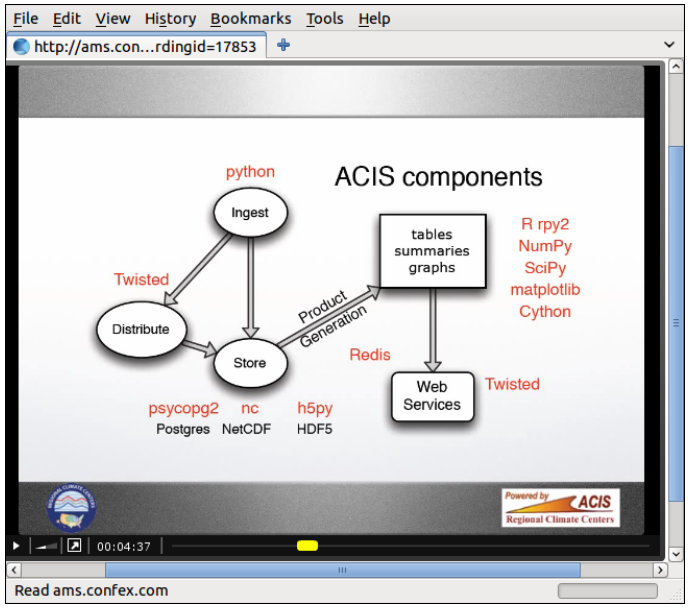

Figure 1.4: Image from: AMS talk by William Noon, Northeast Regional Climate Center, Ithaca, NY, http://ams.confex.com/ams/91Annual/flvgateway.cgi/id/17853?recordingid=17853. Used by permission.