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 2

Using the Python Interpreter and Interactive Development Environment

2.1 Getting and installing Python

In Ch. 1, we saw how Python is practically the best thing for AOS users since sliced bread ☺. Now that I've built-up your expectations, I need to describe the first place where things are not all they should be in the Python universe, and that is the level of support for installation and new science users. In contrast with commercial data analysis software companies (e.g., MathWorks, Wolfram Research, etc.), which make installation a piece-of-cake and provide copious amounts of online, print, and live-human support, installation of Python and all the libraries needed by science users can be tricky. And once the install is complete, there is no simple and comprehensive way to find out everything you have and how to use it.

So, if you're a newbie, what do you do? Unfortunately, there are many ways to install Python and the needed scientific libraries. In this section, I will describe three different ways: just choose the one that works best for you. Each has its pluses and minuses. The first two do not require you to be a budding system administrator (though you will need the password of a user with administrator privileges). The last one requires you to build something from source, so that installation method is for those who are willing to experiment a little more. Please pay attention to the footnotes in this section; that's where I'll put the addresses to the web pages you'll want to access.

2.1.1 The easiest way: EPD

EPD is the easiest way to install Python and the needed scientific libraries.

The easiest way to get Python is to install the Enthought Python Distribution (EPD),[1] which bundles Python with over 100 modules and packages in an easy to install package. The full distribution is free to employees and students of academic institutions.[2] Commercial users have to pay money. Once you have EPD installed, you will have both Python and all the libraries you need to do all the examples and exercises in this book; see Enthought's "Getting Started with EPD" instructions for information on running EPD once you have it installed.[3] EPD is available for Mac OS X, Windows, GNU/Linux, and Solaris.

Enthought also offers a stripped down version of EPD, free to everyone (not just those in academia), called EPD Free.[4] EPD Free is enough to do most of the examples and exercises in this book, except for the overlaying of continental boundary maps on matplotlib plots. In this book, I use the Basemap package to plot those boundaries, and Basemap is not included with EPD Free (though Basemap is included with the full EPD).

If you already have EPD Free and you want to install Basemap, you can use Enthought's package manager to do so: just type `enpkg basemap` at the command-line.[5] However, you need to have an EPD subscription in order to use enpkg; that subscription costs money.[6]

2.1.2 The mostly easy way, but for Ubuntu 12.04

Installing using a package manager on Ubuntu 12.04.

With the Ubuntu 12.04 GNU/Linux distribution (and perhaps later versions), the standard **package manager** will enable you to install everything you need to run Python and do all the examples and exercises in this book.[7] Log in as a user with administrator privileges, open a terminal window, and type in the following at the Unix command-line:

[1] http://www.enthought.com/products/epd.php (accessed August 16, 2012).

[2] The academic download is accessed on this page: http://www.enthought.com/products/edudownload.php (accessed August 16, 2012).

[3] http://www.enthought.com/products/epdgetstart.php (accessed August 16, 2012).

[4] http://www.enthought.com/products/epd_free.php (accessed August 16, 2012).

[5] http://www.enthought.com/products/update.php (accessed August 16, 2012).

[6] http://www.enthought.com/products/getepd.php (accessed August 16, 2012).

[7] See http://packages.ubuntu.com/precise/python for a list of Python packages on Ubuntu 12.04 (accessed August 16, 2012).

```
sudo apt-get update
sudo apt-get install python2.7
sudo apt-get install python-matplotlib
sudo apt-get install python-scipy
sudo apt-get install python-scientific
sudo apt-get install python-mpltoolkits.basemap
```

Feel free to substitute in your favorite package manager (I actually use aptitude instead) for apt-get. After these commands are run, you will have Python 2.7 (plus select libraries) installed.

2.1.3 The not as easy way, but it's still free

If you aren't running Ubuntu 12.04, but you are using Mac OS X or another version of GNU/Linux, you can still use a package manager to install Python and most of the needed scientific packages for this book and build the final package needed (Basemap) from source. See the PyAOS articles on installing on a Mac[8] and installing on GNU/Linux[9] for details. (Note these articles address an installation using Python 2.5; the names may have changed for Python 2.7 related packages.)

While Python works fine on Windows, for the rest of this book, I will assume that you are using a Unix system (e.g., Mac OS X, GNU/Linux, etc.). For many operating systems, the default distribution of Python and associated applications are located in */usr/bin*; if your system is different, please substitute accordingly anytime you see */usr/bin* in this book.

2.2 Getting and installing the course files

Throughout this book, you will find reference to files from a directory called *course_files*. This directory of files is not part of a Python distribution but is instead a set of files I have created for this book. This directory of files is available online at the book's website; see p. viii for details on accessing the site and files.

2.3 The Python interpreter

Python is an interpreted language, meaning that you just type in a command in Python, press Enter, and Python will execute that command right then and

[8]http://pyaos.johnny-lin.com/?p=190 (accessed August 16, 2012).
[9]http://pyaos.johnny-lin.com/?p=76 (accessed August 16, 2012).

there. This is similar to the behavior of Matlab, IDL, and Mathematica, and the environment where this all occurs in Python is called the **interpreter**. Let's try an example:

Example 1 (My first Python interpreter command):

Start out by opening a **terminal window**. Everything you do will be in that window.

Questions to ask if you can't start the Python interpreter.

- Start the Python interpreter by typing `python` at the Unix command-line. You should get something that looks like Figure 2.1. If this doesn't happen, here are some possible fixes:
  - If your environment path is not set up correctly, you may have to type in the full path name to your Python binary. One common name is `/usr/bin/python`.
  - On some other installations, you may have to type in the version of Python you want, e.g., `python2.5`.
  - If you are using Mac OS X and you installed Python using EPD, you may have to type something like:

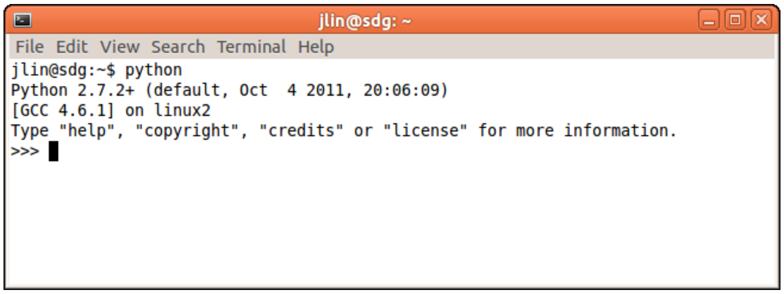
    ```
    /Library/Frameworks/Python.Framework/Versions/
    7.3/bin/python
    ```

    instead of just `python` (or you may want to add the EPD Python path to your shell environment's *PATH* variable).
  - If you are using Mac OS X and you installed Python using the Fink package manager, you may have to type something like `/sw/bin/python2.5`.
- When you see the >>> prompt, type:

  ```
  print "hello world!"
  ```

  and press Enter.
- The interpreter immediately executes the command, printing the string `hello world!` to screen.
- To exit the interpreter and return to the Unix command-line, type Ctrl-d.

Press Ctrl-d to exit the interpreter.



```
jlin@sdg:~$ python
Python 2.7.2+ (default, Oct  4 2011, 20:06:09)
[GCC 4.6.1] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> 
```

Figure 2.1: Starting the Python interpreter in a terminal window.

The Python interpreter has two very helpful interpreter commands:

- `help(x)`: This shows online help for the command *x*.
- `dir()`: This shows (approximately) all the variables and functions defined in the current scope.

The `help` and `dir` commands.

We'll talk more about these commands as we go along. For now, just keep them in mind.

Usually, you will write code in a file and ask Python to execute the code, rather than doing everything interactively in the interpreter. There are a number of excellent and sophisticated programming (or development) environments available for Python, such as IPython,[10] Python(x,y),[11] and Spyder.[12] For our purposes, we'll go basic and consider two very simple ways of setting up a programming environment.

Other development environments for Python.

2.4 The command-line environment

In this example, we set up a programming environment using multiple terminal windows.

Example 2 (A simple command-line programming environment):

- Open two terminal windows on your desktop.
- In one window, use a text editor (e.g., vim, emacs, gedit) to open a file *foo.py*.

[10]http://ipython.scipy.org (accessed August 16, 2012).
[11]http://code.google.com/p/pythonxy (accessed August 16, 2012).
[12]http://packages.python.org/spyder (accessed August 16, 2012).

- In that file, type: `print "hello world!"`
- Save the file.
- In the other window, type: `python -i foo.py`
- Python will execute the commands in your file and leave you in the interpreter, where you can type in more commands, if you wish.

Note, to automatically exit to the operating system after executing the Python commands in your file, do not include the `-i` option.

The `-i` option; interpreter command history.

An aside regarding command history: On many Unix systems, the up- and down-arrow keys allow you to scroll through the history of the commands typed into the interpreter, with the up-arrow scrolling into the past and the down-arrow scrolling back towards the present. This feature, however, is a function of how readline is implemented in the operating system, so this may or may not work for you.

2.5 The IDLE environment

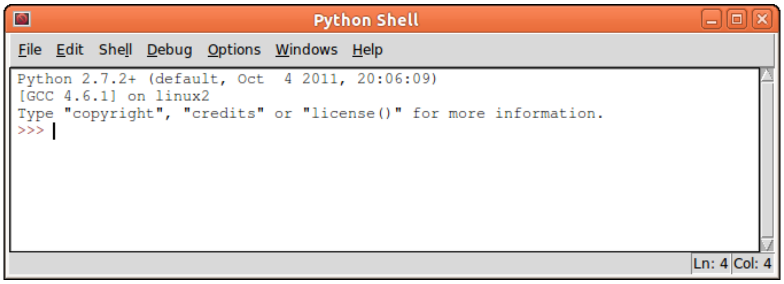
Python comes with its own interactive **development environment** called IDLE. IDLE is actually written in Python, using the Tk GUI widgets system.

Example 3 (Using IDLE):

- To start IDLE, at the command-line, type: `idle &`
  - The ampersand puts the process in the background.
  - If you are using Mac OS X, you may have to first start up X11.
  - Sometimes you have to specify the version number of IDLE, e.g., `idle2.5`. On Ubuntu, it might be called `idle-python2.7`, or something similar.

IDLE has a source code window and a shell window.

- A Python Shell window will automatically open up. (You should get something like what is shown in Figure 2.2, though yours will probably be a little larger.) This window contains a Python interpreter (shown by the >>> prompt), and so whenever in this book I talk about typing something into the interpreter, if it's short, you can type it in here.



```
Python 2.7.2+ (default, Oct  4 2011, 20:06:09)
[GCC 4.6.1] on linux2
Type "copyright", "credits" or "license()" for more information.
>>> 
```

Figure 2.2: The Python shell in IDLE, just started.

- In the menu bar of the Python Shell window, pick File → New Window. (This is a window for a code file. Whenever in this book I talk about typing something into the interpreter, and it's long, type it in here, save it, and run it as described in the next few bullet points.)

- In that window, type: `print "hello world!"`

- Save the file as *foo.py* using File → Save.

- In that same window (your code window), pick Run → Run Module.

- Python will execute, in the Python Shell window, the commands you typed in your code file.

An aside regarding command history: In IDLE, the up- and down-arrows generally do not work to access command history. Instead, place the mouse on the line of the Python Shell you want to execute again, click on that line (i.e., click on the left mouse button; do not select or highlight the line), then press Return, and the line will be duplicated at the current interpreter cursor. If you press Return again, that line will be executed.

Command history in IDLE.

2.6 Exercises using Python programming environments

Exercises are your opportunity to implement what you've learned by trying out the examples; usually, exercises are permutations of topics you've seen in the examples. Try the following exercises to help you get used to the command-line and IDLE environments. Exercise 2 also introduces using Python as a simple calculator.

▷ **Exercise 1 (Printing some words to the screen):**

The `print` command.

- In your first terminal window, open a file and type in a series of `print` commands, operating on strings, one on each line. For instance:

```
print "My name is Johnny Lin."
print "I think Python is neat."
print "I wish I had used it for my Ph.D."
```

- Save the file.
- In your second terminal window, run the file. Did you get what you expected?
- Change one of the lines and add another line printing additional text. Save the file and re-run it.

▷ **Exercise 2 (Using Python as a simple calculator):**

- In your first terminal window, open a file and type in a series of `print` commands, with an arithmetic calculation expression as the argument, one on each line. For instance:

Python as a calculator.

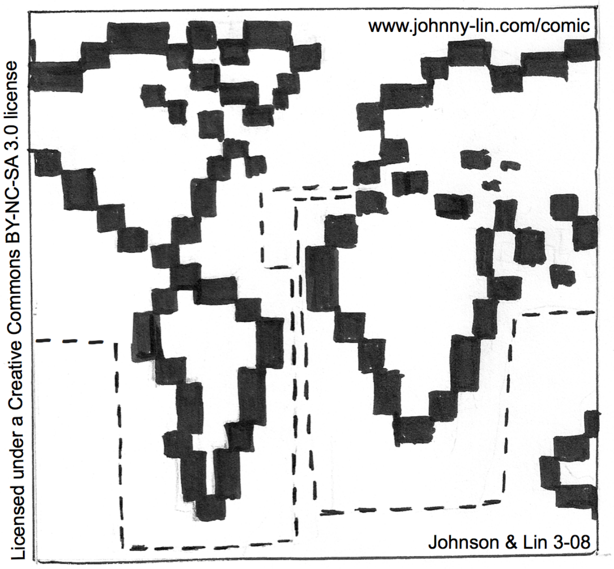
```
print 5*4
print 6/2
print 13+5+7
print 6**2
```

- Save the file. Predict what will happen when you run the file.
- In your second terminal window, run the file. Did you get what you predicted?
- Change one of the lines in the file, save the file, and re-run it.

Solution and discussion: (Cover this up if you haven't finished the exercise!) We'll be talking more about this in Ch. 3, but if you typed in a case where you divided two integers, and the division was not even, you might have encountered results you did not expect. For integer division, when the two operands are both integers, Python throws away the remainder and returns only the quotient.

Integer division discards the remainder.

▷ **Exercise 3 (Getting used to the IDLE environment):**
Do Exercises 1 and 2, but using the IDLE environment.



Magellan Circumnavigates the World: A GCM Study