0.1 Running Python

0.1.1 Your first Python session

Note: Python and Ipython notebook versions of code (`.py .ipynb`).

This section assumes you have installed Python and that you know how to start it up.

Once you have started Python you should see a **Python prompt** in the same window in which you have typed the Python command. If you start python with the python command, the prompt looks like three arrowheads or chevrons all pointing the same direction, like this:

```
>>>
```

In ipython and ipython notebook, the prompt looks like this:

```
In [1]:
```

If you position your cursor after the prompt symbol you can type things. Type the following line, then hit **Enter** (if you are using ipytho notebook, you must hold down the shift key while pressing **Enter**):

```
>>> total_secs = 7684
```

Nothing will happen, but you have told Python that you are storing the value 7684 in the variable *total_secs*. Further statements can make reference to that variable, and the value will be remembered. Let's try a series of statements:

```
>>> hours = total_secs // 3600
>>> secs_still_remaining = total_secs % 3600
>>> minutes =  secs_still_remaining // 60
>>> secs_finally_remaining = secs_still_remaining  % 60
```

In each case, you set the value of a new variable by performing a computation using the value of a variable you had already defined. Python remembers the values and uses them in successive computations. Now let's get Python to print out some of what it knows:

```
>>> print "Hrs =", hours, "mins =", minutes, "secs =", secs_finally_remaining
Hrs = 2 mins = 8 secs = 4
```

This time Python responds. The *print* command asks Python to print its arguments. When the aruments are separated by commas, it prints a space in between them. So the above command asks it to print the string "Hrs =", followed by a space, followed by the value of the variable *hours*, followed by a space, followed by the string "mins =", followed by a space, followed by the value of the variable *minutes*, followed by a space, followed by the string "secs =", followed by a space, follwed by that value of the variable *secs_finally_remaining*.

You can also look at the value of a variable just by typing the variable to the Python prompt. Python responds with the value of the variable:

```
>>> hours
2
```

If you type a variable Python doesn't know the value of, Python responds with an **Error Message**:

```
>>> nanoseconds
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
NameError: name 'nanoseconds' is not defined
```

We will learn about error messages, because understanding error messages is an important part of interacting with Python. The most important line is the last one, which gives you the type of the error, and some specific information about it. In this case we have a **NameError**, which means you have used a name — in this case a variable name — that Python doesn't know.

0.1.2 Python: Workflow and learning strategy

In this section we discuss how to work with Python, trying to motivate a particular strategy for learning it.

1. Start out using `Ipython notebook`. Selected lectures and homework assignments will have associated notebooks, with worked-out examples showing you exactly what goes in to Python and what comes out. Follow the examples closely, copy them, and try to insert new code to do homework problems or to generate examples of your own.
2. Gradually wean yourself away from the notebook, working in text files to produce scripts that can be executed independently. One way to start doing this is take some ideas you have developed in a notebook and turn them into an independent program.

We discuss these two stages below, beginning with some hints on how to start up and interact with ipython notebook and the recommended workflow, and then moving on to how to write independent programs.

Stage one: Using Ipython or Ipython notebook

As detailed above, the recommended way to start out with Python is to use Ipython notebook. To do that you do:

```
gawron$ ipython notebook  --pylab
```

Both `ipython` and `ipython notebook` have excellent online tutorials introducing you to the basics.

1. Ipython help
2. Notebook tutorial/help
3. Markdown language help (for adding text and comments to your notebooks)

In addition the *Python Data Analysis* book has an extended introduction to Ipython that explains many of the most useful features. This introduction, however, will probably be of most use to the very geeky, since many of the features only make sense to those spent many hours at a keyboard trying to get programs to work.

The notebook in particular has a very intuitive user interface that is very easy to pick up, with a well-designed help menu for those features which are a little harder to remember (like keyboard shortcuts).

Very well. The notebook is easy to use. But just because something is easy to use doesn't mean you should use it. Why use the notebook?

`Ipython notebook` is a tool we use to help get you get started writing Python. It allows you to think about Python one line at a time, then a few lines at a time, and to go through the process of getting something to woork on one example, then generalizing into reusable code.

Later on, `IPython notebook` might be a tool you use for working out pieces of a program, or for sharing some ideas, or for producing slides, if it comes to that. But your scripting skills advance, you will find it necessary to produce scripts that can live on their own as executable programs. This is when you need stage two.

[missing material: start up, basic interactions, using a notebook from class]

[missing material: workflow, saving a notebook to be handed in]

Stage two: Text editor or IDE?

This section assumes you have installed Python and that you know how to start up Python. We discuss other ways of interracting with Python.

The next thing you will want to do is to write a simple program of your own, either using ideas you have developed in some notebook sessions, or on a piece of paper. Often this entails something fairly scarey, starting out with a blank screen that you are now supposed to fill with code.

There are two ways of writing a program and running it. The first way is to use an **Integrated Development Environment (IDE)**, which is an interactive program that lets you write, edit, and run your code. When you run your code, you typically see the results immediately in a new window, and typically you can interact with that new Window just as would with Python, gathering information to help you improve or debug your code.

The second way is to use a text editor that lets you save your files as text files (which is what Python program files are), without inserting any funny characters or removing line breaks or doing anything that interferes with the intended meaning of your program. Then you can run the program in Python in a window separate from the editor window, using either simple Python or your favorite interactive Python shell, gather information, and edit again. Typically the editor for this purpose will not be Microsoft Word (although it is possible), but something designed for this much simpler purpose.

The distinction between IDE and editor is actually sort of a continuum, since many IDEs offer a rich assortment of editing aids, and since many editors offer the option of creating a new window in which you can interact with Python.

We discuss the options in more detail below.

Using a text editor

Let us say you have chosen to write your first program in a text editor.

Here are some popular choices.

1. PythonWin (Windows only). Available either as part of ActiveState's ActivePython distribution (which is not open source) or as part of the Win32All extensions from Mark Hammond's pages (which is open source). Note: the Win32 pages were running in Test mode after a Wiki attack in January (as of August 12, 2013). You just click on the Front Page link on teh left to see the usual content.
2. TextWrangler (Macintosh only)
3. Emacs. Very general editor, used for both general text editing and code editing, with good Python facilities. The downside is that it is both a big program and not particularly easy to learn.
4. Komodo Edit Open source editor, very much targeted toward Python.

Using an IDE

Let us say you have chosen to write your first program using an IDE.

Here are some popular choices.

1. Ipython. This comes with the Enthought distribution with a whole set of desirable features and is the recommended option. Available at ipython.org if you don't have the Enthought distribution.
2. Idle (comes with all standard Python installations).
3. Komodo IDE. Komodo is an award winning Python IDE from ActiveState. Fully-integrated Python Python 2.x and Python 3 support featuring code intelligence with autocomplete and calltips, Python debugger (includes remote debugging), interactive shell, remote file support, macros, templating, emacs command support and great help documentation.

There really are a number of options. The following list was posted on StackOverflow, a popular Internet Software question answering site.

```
                                      Rapid Application Development -.
                                           Integrated DB Support -+   |
                                               GUI Designer  -+  |   |
                                            Unit Testing -+   |  |   |
                                       Code Templates -.  |   |  |   |
                                      Code Folding -+  |  |   |  |   |
                          UML Editing / Viewing -+  |  |  |   |  |   |
                             Line Numbering -+   |  |  |  |   |  |   |
                        Bracket Matching -+  |   |  |  |  |   |  |   |
                         Smart Indent -+  |  |   |  |  |  |   |  |   |
        Source Control Integration -+  |  |  |   |  |  |  |   |  |   |
                  Error Markup  -+  |  |  |  |   |  |  |  |   |  |   |
 Integrated Python Debugging -+  |  |  |  |  |   |  |  |  |   |  |   |
   Multi-Language Support -+  |  |  |  |  |  |   |  |  |  |   |  |   |
 Auto Code Completion -+   |  |  |  |  |  |  |   |  |  |  |   |  |   |
Commercial / Free --+  |   |  |  |  |  |  |  |   |  |  |  |   |  |   |
Cross Platform -+   |  |   |  |  |  |  |  |  |   |  |  |  |   |  |   |
               _|___|__|___|__|__|__|__|__|__|___|__|__|__|___|__|___|_
               |CP|C/F|AC|MLS|PD|EM|SC|SI|BM|LN|UML|CF|CT|UT|UID|DB|RAD|comments
               +--+---+--+---+--+--+--+--+--+--+---+--+--+--+---+--+---+
BlackAdder     |Y | C |  |   |  |  |  |Y |  |  |   |Y |  |  |   |  |   |
BlueFish       |L |   |  |   |  |  |  |  |  |  |   |  |  |  |   |  |   |
Boa Constructor|Y | F |Y |   |Y |Y |  |Y |Y |Y | Y |Y |Y |  |   |  |   |
ConTEXT        |W | C |  |   |  |  |  |  |  |  |   |  |  |  |   |  |   |
DABO           |Y |   |  |   |  |  |  |  |  |  |   |  |  |  |   |  |   |
DreamPie       |  | F |  |   |  |  |  |  |  |  |   |  |  |  |   |  |   |
Dr.Python      |  | F |  |   |  |Y |  |  |  |  |   |  |  |  |   |  |   |
Editra         |Y | F |Y | Y |  |  |Y |Y |Y |Y |   |Y |  |  |   |  |   |
Emacs          |Y | F |Y | Y |Y |Y |Y |Y |Y |Y | Y |Y |Y |Y |   |  |   |
Eric Ide       |Y | F |Y |   |Y |Y |  |Y |  |Y |   |Y |  |Y |   |  |   |
E-Texteditor   |W |   |  |   |  |  |  |  |  |  |   |  |  |  |   |  |   |
Geany          |Y | F |Y*| Y |  |  |  |Y |Y |Y |   |Y |  |  |   |  |   |* very limited
Gedit          |Y | F |Y¹| Y |  |  |  |Y |Y |Y |   |  |Y²|  |   |  |   |¹ with plugin ² sort of
Idle           |Y | F |Y |   |  |  |  |  |  |  |   |  |  |  |   |  |   |
JEdit          |Y | F |  | Y |  |  |  |  |Y |Y |   |Y |  |  |   |  |   |
KDevelop       |Y | F |  | Y |  |  |Y |Y |Y |Y |   |Y |  |  |   |  |   |
Komodo         |Y |C/F|Y | Y |Y |Y |Y |Y |Y |Y |   |Y |Y |Y |   |Y |   |
NetBeans       |Y | F |Y | Y |Y |  |Y |Y |Y |Y | Y |Y |Y |Y |   |  | Y |
NotePad++      |W | F |  | Y |  |  |  |  |  |Y |   |  |  |  |   |  |   |
Pfaide         |W | C |Y | Y |  |  |  |Y |Y |Y |   |Y |Y |  |   |  |   |
PIDA           |LW| F |Y | Y |  |  |  |Y |Y |Y |   |Y |  |  |   |  |   |VIM based
PTVS           |W | F |Y | Y |Y |Y |Y |Y |Y |Y |   |Y |  |  |Y* |  | Y |*WPF bsed
PyCharm        |Y | C |Y | Y*|Y |  |Y |Y |Y |Y |   |Y |  |Y |   |  |   |* javascript
PyDev(Eclipse) |Y | F |Y | Y |Y |Y |Y |Y |Y |Y | Y |Y |Y |Y |   |  |   |
Pyscripter     |W | F |Y |   |Y |Y |  |Y |  |Y |   |  |Y |Y |   |  |   |
PythonWin      |W | F |Y |   |Y |  |  |Y |Y |  |   |Y |  |  |   |  |   |
SciTE          |Y | F |  | Y |  |Y |  |  |Y |Y |   |Y |Y |  |   |  |   |
ScriptDev      |W | C |Y | Y |Y |Y |  |Y |Y |Y |   |Y |Y |  |   |  |   |
SPE            |  | F |Y |   |  |  |  |  |  |  | Y |  |  |  |   |  |   |
Spyder         |Y | F |Y |   |Y |Y |  |Y |Y |Y |   |  |  |  |   |  |   |
Sublime Text   |Y | C |Y | Y |  |  |  |Y |Y |Y |   |  |Y |  |   |  |   |extensible w/python
TextMate       |M |   |  | Y |  |  |  |Y |Y |Y |   |Y |Y |  |   |  |   |
UliPad         |Y | F |Y | Y |Y |  |  |Y |Y |  |   |  |Y |Y |   |  |   |
Vim            |Y | F |Y | Y |Y |Y |Y |Y |Y |Y |   |Y |Y |Y |   |  |   |
WingIde        |Y | C |Y | Y*|Y |Y |Y |Y |Y |Y |   |Y |Y |Y |   |  |   |* support for C
```

```
 Zeus           |W | C |  |   |  |  |Y |Y |Y |Y |   |Y |Y |  |   |  |   |
                +--+---+--+---+--+--+--+--+--+--+---+--+--+--+---+--+---+
                |CP|C/F|AC|MLS|PD|EM|SC|SI|BM|LN|UML|CF|CT|UT|UID|DB|RAD|
                |__|___|__|___|__|__|__|__|__|__|___|__|__|__|___|__|___|
```

Source Stack Overflow

If you really want another uptodate list of some of the options has been posted on this page Python Editors page for a list of editors and IDEs. Most of the options offered are free. This is a good thing, not a bad thing, as we explain below.