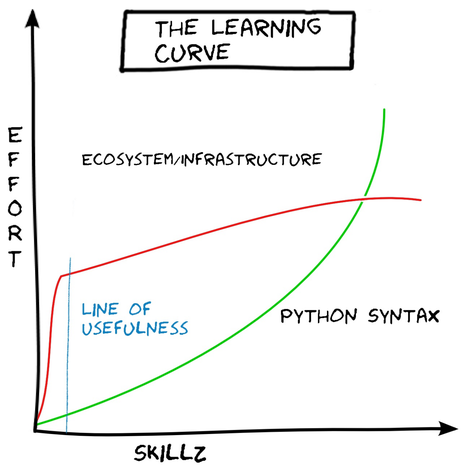
Is Python Beginner-Friendly?

Python is a great language to introduce beginners to programming. Its roots as a teaching language are seen in the simple uncluttered syntax and ability to write code in several paradigms (procedural, functional, oop). But the ecosystem and infrastructure needed to be productive with Python has many poor user experiences. Beginners can get stuck trying to use the beautiful language usefully.



"There should be one-- and preferably only one --obvious way to do it."

- the Zen of Python

Joe the Windows User

Joe installs Python by downloading an installer for the 64 bit Python 2.7 from python.org. Once the installer is finished he looks in his start menu to find new programs. He finds IDLE and starts it but has the misfortune of clicking on the window moving the position of the cursor and consequently IDLE ignores his typing. He tries the "Python Console" entry on the Start Menu but can't figure out how to import the simple python script on his Desktop.

After doing some reading Joe tries starting up a command prompt, changing to the same directory as his script and typing

```
C:\first_project> python
```

but it just says file not found. After some more reading he decides to try a different console and tries

```
C:\> easy_install ipython
```

but easy_install isn't available either.

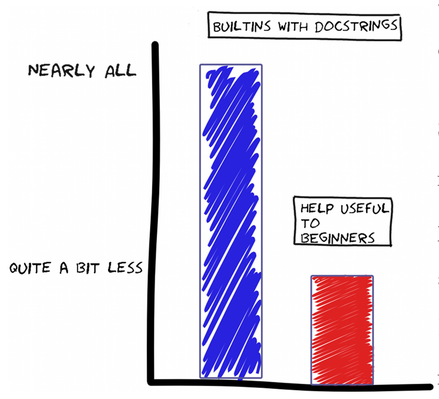
Joe reads some more and downloads setuptools from pypi but it won't work with his 64bit install. He uninstalls and reinstalls the 32bit version and this time setuptools installs sucessfully but he still can't run easy_install from a command prompt. He creates an account on StackOverflow where immediately he is told he should be using distribute instead of setuptools. And virtualenv. And pip instead of easy_install. Joe wonders how well Ruby works on Windows...

Jane the Scientist

Jane is familiar with a variety of programming languages and environments like IDL, R, and Matlab. Jane's developer friends tell her that Python is a great multi-purpose language that will let her tackle a variety of tasks in addition to the things she typically does. Python, they say, has a great interactive shell, built in help, easy syntax, and a large standard library! Plus it's already installed on your Mac!

Jane decides to learn the basics of syntax and try her hand at a few tasks that have stumped her in other languages. The shell is nice at first but she quickly discovers it has no tab completion, no syntax highlighting, and very limited history support. The help browser isn't as much help as she hoped - the "help" for basic statements like "if" and "for" looks very technical and doesn't have any examples. She clicks the link for documentation at python.org but is stumped by the choices. Does she want What's New, the Tutorial, or the Library Reference? Perhaps the convenient search box will help, but searching for "if" just brings up lots of information on Python bytecode.

Jane has helpful coworkers who give her a few quick pointers. She learns some basic syntax but exploring the language isn't always easy. Some of the builtin functions don't have very useful help (*sorted*) and some have mathy descriptions (*zip*). Fortunately Jane is fond of saying "i-ith" and presses on.



She has an xml file to parse so she starts Python. Importing xml, help tells her that there are four modules available but she doesn't see how to choose between them. Using search on python.org is unhelpful again. Jane tries help on xml.etree but it's blank. "Maybe I need to wait till I have time read a book" she thinks...

What Should We Do?

1. Apply the Zen of Python to more than the language
2. "Make Usage Easy and Obvious" [Apple HIG]
3. Communicate first for those who need it most
4. Consider doing less

I love Python and I see reasons for optimism. The 3.3 release adds Python to the path on Windows by default and added virtualenvironments as a built-in feature! It added a launcher to Windows that respects !# lines. xml.etree has better docs! Yay! I also appreciate 3rd party efforts to improve Python libraries (Python for Humans!) and improve/document the packaging tools (Hitchhiker's Guide to Packaging).

But we can do more. The official website should have an authoritative and opinionated voice about the obvious way to do everything related to python. The official documentation must be accessible to the beginner. Docstrings (particularly builtins) should communicate by example to the beginner. Tools like IDLE should be improved or moved to separate projects. All these suggestions have at their root the idea that the user is never wrong - and that we could minimize non-essential barriers to entry simply by applying our own philosophy more broadly and consistently.

Simeon Franklin / @simeonfranklin / http://simeonfranklin.com

http://marakana.com - Helping people get better at what they do. Python/Javascript/Android/Scala/Hadoop and much more!