25
Feb
2010

The Nature and Importance of Source Code and Learning Programming with Python

Last year a client asked us for advice on getting started with programming. So I thought I'd share some thoughts about programming, its relationship with FOSS (Free and Open Source Software) management and why Python is a good language for learning programming including some great on-line resources. But first I want to make sure our business-oriented readers understand the nature and importance of source code.

The "source" aka "the code" provides a language in which computer users can create or change software. One does not have to be a programmer to work on the code. In fact, every computer user is, ipso facto, a programmer! Menus, web interfaces, and graphical user interfaces (GUIs) are some of the more facile "languages" for computer programming that everyone, even children, can readily learn and use. Of course, building complex software systems requires a more expressive specification language than a web form, for instance, can provide.

Although all computer software is specified with source code, FOSS systems are unique in that the source code is made available with the software. In contradistinction, *software lock-in* or vendor lock-in describes the unfortunately all too common practice of many organizations to block access to their source code.

Having access to the source code provides huge operational benefits. For one, the source can be used to understand how the software works: it is a form of software documentation (indeed, it is the most definitive form of software documentation possible!). Also, code can be easily changed to add diagnostics or to test a possible solution to a problem or to modify or add functionality. In addition, the source is a language both for specifying features to the computer and for discussing computing with others. So most mature FOSS languages have vibrant support communities in which one can participate, learn and get help.

The source is a tool: a powerful, multi-purpose, critically important tool.

Since LinuxForce focuses on FOSS, we are able to take full advantage of the availability of the code. We are always working with the source! Since most of our work is systems administration, we usually "program" configuration files. However, we also write systems software and scripts and we support software developers extensively, so we have a persistent, deep, and productive relationship with code.

But what to suggest to someone like our customer who wants to learn programming?

I remembered seeing a blurb in Linux Journal referencing an article they published in May 2000 by Eric Raymond entitled "Why Python" which argues persuasively for the virtues of the programming language Python. I had often felt that Perl's idiosyncrasies made it difficult to use, so Eric's critique of Perl and accolades for Python were convincing to me. In addition, I follow FOSS mathematics software and I was aware that Sage is a Python "glue" to more than fifty FOSS math libraries. I've been meaning to look into Python so I could use Sage. Another pull comes from my work at

LinuxForce where we use a lot of Python-based software including mailman, fail2ban, Plone, and several tools used for virtual machine management such as kvm, virtinst and xen-tools. Python has a huge software repository and community. So one is likely to find good libraries to build upon (thus avoiding the extra learning curve of building everything from scratch). Python is an interpreted language which makes it easier to debug and use so the learning process is smoother.

To finish the recommendation, I just needed to find some on-line resources. First, Kirby Urner suggested these two: Wikieducator's Python Tutorials and "Mathematics for the Digital Age and Programming in Python". Then, I checked out the Massachusetts Institute of Technology's (MIT) OpenCourseWare which provides extensive course materials for many of their classes (I've already watched the full video set for a couple of MIT's courses including the legendary Walter Lewin's "Classical Mechanics" and have been very impressed by the quality and content of their materials). After nearly 30 years of introducing students to programming with Scheme, **MIT switched to Python** in 2008! The materials for their introductory Python-based course "6.00 Introduction to Computer Science and Programming" are very thorough, accessible and helpful. Their free on-line materials include the full video lectures of the class plus assignments, sample test problems, class handouts, and an excellent Readings section with references to "the Python Tutorial" and a very good free on-line textbook "How to Think Like a Computer Scientist: Learning with Python".

In conclusion, if you or anyone you know wants to learn how to program computers, I recommend starting with Python using MIT's on-line course materials supplemented with the other on-line resources mentioned above (and summarized in the table below). I've now watched more than half of the videos from the MIT 6.00 course and I've worked through several of their assignments: *this is a great course*! Even with nearly three decades experience programming including a couple of college-level courses in the 1980s, I'm finding the class is more than just good review for me: I've learned a few new things (in particular, dynamic programming and the knapsack problem). Python's clean syntax and elegant design will help as one delves into writing code for the first time. Its extensive libraries and repositories will support the application of one's newly acquired computing skills to solve problems in the area of the student's special interests whatever they may be … and that's the way we learn best: by doing something that we personally care about!

Summary of On-Line Resources for Learning Python
• Python Programming Language — Official Website • Wikipedia article on Python • MIT's Course 6.00 Introduction to Computer Science and Programming including complete video lectures about learning Python and programming. • The free textbook How to Think Like a Computer Scientist: Learning with Python • The Python Tutorial • Python Library Reference • The Python Community • PyPI: The Python Package Index (a repository of software for Python) • Wikieducator's Python Tutorials • Mathematics for the Digital Age and Programming in Python • Eric Raymond's May 2000 Linux Journal article "Why Python"

Posted by CJ Fearnley *in* Programming, *0 comments*