By Boudewijn Rempt

Table of Contents

Introduction....................1

What Can You Do With Python?....................3

Web Sites....................5

Books....................6

The Python Community....................8

Development Environments....................9

Introduction

Boudewijn Rempt is a senior software developer with Tryllian in the Netherlands. Searching for a programming language more productive than even Visual Basic, he came across Python and became an instant enthusiast. He is the author of a popular tutorial on Python as well as the KDE GUI toolkit. In addition to the forthcoming recommendations, visit his Python page for more information.

Introduction
Python, which was created by Guido van Rossum, is a wonderfully versatile language. There are few tasks that it can't handle, and even fewer operating systems that it doesn't run on (there has been talk of a port to the IBM S/390, but there is no port for the Atari ST). You can use Python for building dynamic Web sites, analyzing logfiles, automating MUDS, scanning your mail, building GUI applications, and accessing databases. Best of all, you'll have a lot of fun when working with Python.

Here is a list of operating systems to which Python is ported:

- Unix (Linux, Sun Solaris, SGI IRIX, DEC Unix, AIX, HP_UX, SCO, NeXT, BSD, FreeBSD, NetBSD)
- Windows (3.11, 95, 98, NT, 2000, CE)
- DOS
- Macintosh
- OS/2
- BeOS
- Amiga
- QNX
- VMS
- Psion Epoc
- RISC OS
- VxWorks

Python is quite easy to learn, and it encourages you to write clear code. In fact, Python is often described as "executable pseudo−code." Python uses indentation to structure your code, which means that you no longer have to count braces! Don't be put off by this—no Python developer has ever lost the logic of his code through white, space−eating nano−viruses; after your first afternoon with Python, you'll just love the look of crisp code without comic−book curses (!@#$%^).

It's easy to use small scripts in order to automate everyday tasks, but python can be used for large, more complex tasks. For example, it has been used for relational databases (Gadfly, at http://www.chordate.com/gadfly.html), mailing list managers (Mailman, at http://www.list.org/), and complete Web publishing systems (Zope, at http://www.zope.org/).

Python is extensible, and its modules are written in C, C++, or any other compiled language. JPython is a version of Python that runs inside Java and enables you to access all Java libraries. On Windows, you can use Python to implement and use COM objects, which is basically the whole of Windows; this means that every Microsoft Office application is scriptable through Python. On Linux, you can use Python to write applications for both KDE and GNOME. Standalone applications that are portable among Unix, Windows, and Macintosh are also possible.

Python isn't "pure" in any academic sense, and this is actually one of its strengths. It's a profoundly practical language, designed not to make a theoretical statement, but for its easy learning curve and productivity. You can write object−oriented code in Python, but you can just as easily write procedural scripts if the need arises. One of the wonders of Python is that it's so dynamic. Everything is an object, so you can make lists of classes and functions and pass a class as a parameter in a function. Oh, and yes, Python is named after Monty Python's Flying Circus.

What Can You Do With Python?

There are a few things you *cannot* do with Python, such as writing device drivers or operating system kernels, but it's a perfect solution for most other purposes.

Web Publishing

Web publishing is one area where Python shines. Whether it's for CGI scripting to complete Web site authoring systems, such as Zope, Python has a solution.

If you're ambitious, you can write your own Web server. This is astonishingly easy, and Python's standard library includes the basics. For more complex approaches, take a look at either my creation, Kura, which is a dedicated Web server, or Chuck Esterbrook's WebWare, which is a lightweight Web application development kit. Another example is Medusa, which takes a very interesting approach to building a high−performance Web server in Python.

XML Handling

Python is an excellent language for writing XML applications. You can use the xmllib library that's included with Python, or you can opt for the full−featured SAX and DOM implementation created by the XML−SIG. I prefer to use xmllib (which is included in the standard distribution) for simple parsing applications such as Web templates, and then use 4DOM for complicated, DOM−based applications such as an XML editor. Lars M. Garshol has developed other libraries, including a SAX library, a validating parser, an Xpointer implementation, and a DTD documentation generator.

GUI Applications

Using Python is an absolute paradise for people who want to experiment with GUIs. More specifically, you can choose from a bewildering variety of *bindings* to GUI libraries, such as MFC, Tk, or Qt. No current binding standard exists; Tkinter used to be the standard, but it's old−fashioned (for instance, it hasn't got a standard tree control) and it doesn't look very pretty. Besides, serious problems can arise from running Tkinter on Macintoshes.

wxPython is a binding to the cross−platform (Windows and Unix) wxWindows library. Its main disadvantage is that you need several libraries: the original GUI (GTK on Unix), wxWindows, and wxPython. That makes it a big library, and one that's difficult to install.

PyQT and PyKDE are bindings to Qt and KDE. These are very complete, flexible bindings that offer a great object model for your applications. You can use PyQT on both Windows and Unix, but PyKDE is usable only on Unix. PyKDE extends PyQt by offering desktop integration and complex objects, such as an HTML widget. If you'd like to read more, I've written a beginners' tutorial on the subject.

PyGTK and PyGnome are bindings to the GTK widget set and the Gnome desktop. Personally, I think that the GTK object model isn't as clean or as clear as that of PyQT or wxPython, and the GTK C documentation isn't as readable as the Qt C++ documentation.

Apart from these mainstream GUI bindings, smaller offerings exist, such as PyFLTK. This is worth checking out if you're curious—Amulet is especially interesting. Look for more at Gadfly, at, but you can also access

MySQL, Sybase, Oracle, and others with the DB−API II database access API. However, this API is rather low−level, it is not object−oriented, and not yet pervasive. For instance, there's no DB−API II module for the popular PostgreSQL database. When I do database work, I write an object−oriented wrapper API around the DB−API II, which makes it easy to switch to another access API. Of course, you can also go for ODBC, with mxODBC, but you'll pay a (not too severe) performance penalty.

Extending Python

It's really easy to extend Python. On Windows, you can use any COM object in your Python applications. If you use JPython, the Python in Java, you'll have access to every Java library and every Java bean. You can also easily wrap any compiled library in Python with either SWIG, which is best suited for C applications and very well−documented, or SIP, the "small SWIG" created by Phil Thompson to wrap C++ libraries. SIP is underdocumented, but it offers support for Qt's signals and slot mechanism.

Web Sites

Python.org
Audience: Literally everyone. Start your Python experience *here*.

Description: Python is freely available for almost every computing platform under the sun, and this is the place from which to obtain the latest version. It's also the place where people come together to discuss things such as database connectivity, Python in education, Python in SIGs, and special−interest groups.

http://www.vex.net/parnassus/
Audience: Every Python developer.

Description: The grand treasure trove of all Pythondom, the Vaults of Parnassus Web site offers an index of almost all Python modules, Python applications, Python tutorials, and Python Web sites. (It's also a Python−powered site!) Whenever you've written a bit of Python that could be generally useful, announce it here—Parnassus will announce it for you on comp.lang.python.announce and on the Python announcements mailing list.

Python Software Activity
Audience: Serious Python developers.

Description: The Python Software Activity is an organization of volunteers that keeps the Python infrastructure (Web site, FTP site, conferences, and so on) running. It's financed by memberships of Python developers; individual memberships are $50. Benefits include conference discounts, access to private mailing lists, the alpha development code of Python, and, of course, inclusion in the central network of Python developers.

Regular Expressions
Audience: Developers who are interested in scripting languages, especially Python.

Description: "Regular Expressions", a column by Cameron Laird and others, is published two times a week by Sun World. This column offers a great deal of interesting commentary on scripting languages in general. Much of it is quite interesting for Python developers.

General Interest Sites
Several sites of general interest to the Python community are worth visiting. Computer Programming for Everybody is a project started by Python's inventor, Guido van Rossum, to stimulate programming education at schools that use Python. The Python Starship is a Web site where experienced Python developers make their work available—it includes several useful modules. The O'Reilly Web site regularly publishes interesting and thought−provoking articles on Python, such as this one about using Python in schools. When working on a GUI, don't forget to pay a visit to the Interface Hall of Shame, where you can find abundant examples of horrible GUI design.

Books

There was a time (not long ago) when only one Python book existed: O'Reilly's Programming Python, by Mark Lutz. When Internet Programming with Python was published, two books existed. Nowadays there are many good books to choose from, and the venerable *Programming Python* is going into a second edition. If you buy Python books, consider supporting the PSA by making a purchasing them from the PSA Bookstore.

Recommended Books

Internet Programming with Python
by Aaron Watters, Guido van Rossum, and James Ahlstrom.
Published by MIS Press/Henry Holt.

Audience: Beginning Python developers.

Description: Almost as venerable as the language itself, *Internet Programming with Python* has a notable distinction: The creator of Python is one of its authors. The layout of this book is horrible, but its contents are valuable. This book deals with the ins and outs of the language, and goes on to teach you about several important areas of Python programming: text handling, CGI scripting, network programming, GUI building. It ends with extending and embedding Python. This book spends a fair amount of space warning you that Python doesn't work as you might expect; this can be a bit off−putting, but the explanations are generally clear, concise, and correct.

Python and Tkinter Programming
by John E. Grayson.
Published by Manning.

Audience: *Every* developer of GUI interfaces.

Description: This is the definitive reference to the aging standard GUI of Python, Tkinter. Even though you might decide not to use Tkinter for your GUI—after all, far better options are available—you will want to buy this book. This resource provides good, general coverage on GUI building and GUI design. Its tone is light, the explanations are exceptionally clear, and the author consistently explains why he did things one way rather than the other. This book covers all aspects of GUI design, including even the latest trend of "real−world metaphors", in which an application looks like its counterpart in the real world. You can even download sample chapters in order to get an idea of what the book is like.

. **Python Programming on Win32,**
by Mark Hammond and Andy Robinson.
Published by O'Reilly.

Audience: Windows developers who want to use Python.

Description: Try the sample chapters for yourself. If you're going to use Python to build Windows−specific applications, you will need this book. Python is a great tool for scripting COM objects, and is even more robust than Visual Basic. Mark Hammond and Andy Robinson are technical experts who are also able to write about complex concepts, such as COM, in a clear style.

Books of Note

Learning Python,
by Mark Lutz and David Ascher.
Published by O'Reilly.

Audience: Beginning Python developers.

Description: This is a very good introduction to the Python language. If you find that the online tutorial is lacking, supplement it with this book—it seems that everybody who has read it raves about it. Try the sample chapters to see whether the style fits you. If this book isn't right, you may prefer *The Quick Python Book*.

The Quick Python Book,
by Daryl Harms and Kenneth McDonald.
Published by Manning.

Audience: Good for non−programming professionals who need a programming language for their work.

Description: *Learning Python* is a book for people who already know one or more programming languages. *The Quick Python Book* is suited for people for whom Python is their first programming language. This is not to say that the book doesn't present the reader with advanced subjects such as COM and Zope—it does. Sample chapters on file handling and modules are available online.

Python Essential Reference,
by David Beazley.
Published by New Riders.

Audience: All Python developers.

Description: Although the typeface in this book is very small, it is a good reference for all the Python modules and is better organized than the online library documentation.

(The eff−bot Guide to) The Standard Python Library,
by Fredrik Lundh.

Audience: All Python developers.

Description: This is an electronic book, usable only with US versions of Acrobat Reader on Windows. Being a collection of 320 examples from years of Usenet postings to comp.lang.python, it's supposed to be good. It might be worth your time if you prefer to learn programming by looking at snippets.

The Python Community

Python is widely used by millions of people, but despite its size, this community is still very friendly and welcoming to beginners. This is probable because Python is such an excellent first language to learn.

comp.lang.python

Audience: Everyone.

Description: This newsgroup is the most important Python community resource. It is the place to ask a beginner's question such as "How do I write a CGI script in Python?", as well as to discuss complex matters such as type inference, garbage collection schemes, microthreads, or stackless versions of Python.

Mailing Lists

Audience: People with a special interest in some area of development, such as XML or internationalization.

Description: Then there are the various mailing lists[md]mirrors of comp.lang.python and comp.lang.python.announce, and mailings lists of the diverse SIGs. The SIGs are groups of Python developers who have a special interest, such as an interest in database development or XML. The SIGs and their mailing list archives are hosted at Python.org.

Development Environments

Compared to C++ or Java, comparatively few IDEs exist for Python. This notwithstanding, most programmers' editors have a Python mode, and you can find a number of useful tools for Python development.

Editors NEdit

Audience: Linux developers with a Windows background.

Description: Chances are, your favorite editor supports Python. My own preferred editor, NEdit, has good Python syntax highlighting but doesn't try to be overly clever with indenting rules—it also cannot run Python from the editor. Still, NEdit does have keystrokes that are very much like those of most Windows editors, which is a blessing because I spend much of my daytime development work on Windows. NEdit runs on only X11 systems.

XEmacs and VIM

Audience: Traditional Unix developers.

Description: If you want more intelligent indentation and you aren't afraid of some serious learning, try XEmacs—it has the most extensive Python support of all of the editors. VIM has a good Python mode too, but I don't personally like the autoindenting. Both Emacs and Vim are available on almost any platform.

Visual Slickedit

Audience: Professional software developers.

Description: A couple of commercial editors (such as Visual Slickedit) offer Python support, but I do not feel that they are really worth the effort. For instance, Visual Slickedit is as involved and idiosyncratic as Emacs, but it costs a great deal of money.

IDEs IDLE

Audience: All Python developers.

Description: Three IDEs are worth mentioning. The first is IDLE, which you already have—it's hidden in the Tools subdirectory in the Python distributions,. IDLE has a good class browser, but the editor isn't very strong: it's a bit slow and doesn't allow for extensive customization or scripting. IDLE runs everywhere Python and Tkinter runs, which excludes Macintosh. IDLE is not suited for very intensive development, however, due to its lack of speed and its awkward menu interface; however, to be fair, a lot of people do all of their Python work with IDLE.

Pythonwin

Audience: Windows developers.

Description: This is a Windows−only IDE, but it's very, very comfortable. It's not included with the standard Python distribution, but you can get it at thePython Web site. Pythonwin has a good source navigation, a useable debugger, an excellent (folding!) editor, integrated help, and a COM browser. Another nice feature is a Python syntax checker that can run without running your script. Pythonworks

Audience: Professional developers.

Description: The last recommended IDE is Pythonworks, from Secret Labs. Pythonworks looks gorgeous—it's written in Tkinter but runs only on Windows. It's also very expensive, but it promises to have a great editor, a visual layout editor, and a good debugger.

Debuggers: IDLE, PyDebug, DDD, Pythonwin

Audience: All Python developers.

Description: Unfortunately, I can't give a good recommendation for a debugger. I have tried four: the IDLE debugger, PyDebug, DDD with a pydb interface, and the Pythonwin debugger. None of these are really as good as you'd hope. For instance, none of these debuggers supports altering the code while running the script. This is a pity because Python offers all the hooks to write a really good debugger—and I fully intend to write one, one day, unless you beat me to it! Another problem is that most Python debuggers don't know where to stop: If you're not careful, you get lost in the system library, which can be terribly confusing—especially if you wander into the functions that the debugger uses to print its messages!

Profilers: Python Profiling Module

Audience: Developers who have a performance problem.

Description: Python offers a standard profiling module and, in contrast to the debugger situation, this is quite ample. You can easily use this to locate your bottlenecks. This profiler is included in your Python distribution—just look at the library reference, Chapter 10.

Class Browsers: IDLE, Pythonwin, Kpybrowser

Audience: Developers who are building a large application.

Description: Python has a standard module, pyclbr.py, that offers services for class browsers. I know of three implementations: one in IDLE, one in Pythonwin, and one that I have written myself, Kpybrowser, which works only with Qt or KDE. Kpybrowser enables you to open new modules and module paths in the browser and can be made to work with your preferred editor.