Building an Open Source Project and Community

In 1991, Guido van Rossum made the Python programming language, open source. Today, Python enjoys massive popularity among developers as part of the open source LAMP platform (Linux, Apache, MySQL, and PHP/Perl/Python).

Timeline

1982
Guido starts work at CWI in Amsterdam on ABC

December 1989
Started development of Python

1991 Release of Python
First open source release of Python

1995 Moves to United States
Worked for CNRI in Reston, VA as a researcher

2000 Python Labs
Starts working for Zope Corporation as Director of PythonLabs

2000 Marriage
Guido marries Kim

2001 Python Software Foundation
A non-profit foundation is formed

2001 Fatherhood
Orlijn is born

2003 Elemental Security
Moves to San Mateo, California to work for Elemental Security

Quotes

"I love Python, it follows Albert Einstein's principle — Everything should be as simple as it is, but not simpler."
Anthony Barker, August 2002

"Python programmers have it all: an elegant language that offers object-oriented programming support, a readable, maintainable syntax, integration with C components, and an enormous collection of pre-coded standard library and extension modules. Moreover, Python is easy to learn but powerful enough to take on the most ambitious programming challenges."
Book News, 2004

Introduction

On February 17, 2005, Guido van Rossum was the speaker at Xerox PARC at 7 PM as part of SDForum's Distinguished Speaker program. The speaker was introduced as "Python is my favorite programming language. It is a pragmatic, flexible language. There is an amazing community built around it, and many toolkits."

Why has Python been successful? Guido observed that many project structures are successful at a given time and place, but not later. The AT&T model of restricting Unix only to universities was successful, but it was a model that others were unsuccessful in imitating.

As the evening progressed, it became evident that the audience had a large number of Python programmers. It felt like a prophet was answering questions from his disciples. When the program ended at 9 PM, it was evident that the questions could have easily continued for another hour or more.

Pre-History

Guido van Rossum started work at the CWI (Center for Mathematics and Computer Science) in Amstersdam in 1982. He was part of the group that designed and implemented ABC, an elegant and powerful language intended for non-professional programmers.

ABC can be thought of as a language that is integrated into its environment, whereas Unix is the tying of small tools together. ABC was a "perfect" language when it was conceived, but it was difficult to extend, add, or improve the language. It was hard to add new "primitive" operations to ABC. It was a monolithic, "closed system" that had only the most basic I/O operations. While ABC had powerful data types, it was difficult to optimize usage for common cases. It also had "features" such as all keywords having to be capitalized and an integrated structured editor that almost everyone hated to use. It had no way to do graphics, and it was very difficult to use it to interact with the rest of the world.

Origins of Python

In 1986, Guido had left the ABC project and started work on Amoeba, a distributed operating system. Three years later, in December 1989, he was looking for a "hobby" programming project that would keep him occupied during the Christmas holidays. He decided to write an interpreter for a new scripting language. He chose Python as the working title for the project, being in a slightly irreverent mood and a big fan of Monty Python's Flying Circus.

Python was built on the same philosophy of ABC, keeping the things about it that were good, but adding the capability to work well with all the files within an operating system, and with C programs. Initially, Guido started with a simple and stupid parser generator, provided a crude virtual machine, parser, and objects; and wrote the first Python code. He then went back and added more data, syntax, and objects to allow Python to be used to write real applications. A major feature of Python taken from ABC was indentation for statement grouping. This was done both to reduce visual clutter and make programs shorter and faster to understand. By forcing an uniform structure, it also made it easier to read other people's code.

Initial Use and Release

In 1990, Python started to be used internally at CWI. A person or two on each floor of the building would start to use it, and in turn would make suggestions on how to improve it. This resulted in Guido writing a Python tutorial and reference manual.

Now the nature of a research organization is that it likes to share software with the rest of the world. Previously, CWI had developed some software, but had tried to use an AT&T Unix license that only allowed universities and non-profits to use it, reserving the commercial rights. But their efforts had been unsuccessful, due to a basic misunderstanding of open source. By having restrictive licenses that told anyone who wanted to use it commercially that they would owe large sums of money, the net result was that much good software did not get adopted.

But Python was under the radar of CWI management, so Guido was able to get away with using the same license as was used for X11 — this was before such software was called open source — when it was released to the world in 1991. People decided that they couldn't get hurt too badly if they downloaded the software and used it.

In 1991, Guido left the Amoeba project and joined the multimedia group at CWI. Most of this group's work started to be implemented in Python.

Initial Growth

The result of the Python open source release was that a lot of comments and feedback started to come back, much of which noted problems specific to a particular implementation of Unix that prevented them from building on their box. Python was first implemented on a DEC VAX, but people were running it on platforms such as the Amiga and Macintosh. The byte and bit order on these machines was different from the VAX. This led to Python inspecting the platform that it was running on to see what it was capable of doing, than doing the appropriate build. This was possible since Python could manipulate bytes and bits very efficiently — using a very small amount of dedicated, low-level code, you could obtain good performance.

Initially the Python user community was a mailing list file containing several hundred emails. When people suggested setting up a newsgroup, Guido said, "if you can get it up and running, great!" And this is how things have subsequently happened in the Python community. Guido never says, "we need," but rather, "if you want it, why don't you set it up?"

By the early 1990s, there was a growing worldwide user base for Python. In 1994, Michael McLay posed the question, "What happens if Guido got hit by a bus?" People liked Python so much that they had become dependent upon it. Michael proposed having bylaws and a formal organization, but he simply was not the right guy for the circumstances.

Conferences

For two months, at the end of 1994, Guido was a guest researcher at NIST, working on Python. NIST organized the first Python workshop in Maryland where he worked with a group of 20 people. (Interestingly enough, Guido remains in touch with 10 people from this workgroup to this day.)

NIST also sponsored a trip to Santa Fe, NM where Guido was the keynote speaker at Tom Christiansen's VHLL (very high level languages) conference. There were 400 people at the conference, and many were at odds between Perl and Python. But neither Guido or Larry had much ego involved, and hence they got along fine. At the conference, Guido met people who wanted to use Python to do serious work.

The net result was that in April 1995, Guido moved to Gaithersburg, Maryland, to work as a guest researcher at Corporation for National Research Initiatives (CNRI), doing work on mobile agents in distributed systems using interpreted languages, most of which involved using Python.

(The problem with CWI was that it was government funded, and hence there was considerable budget pressure. But it was a good home for Python for its first five years.)

As the Python user community grew, a conference started to be held every six months with 200 people attending. As it grew, they went to an annual meeting in the late 1990s, with between 300 and 400 people paying to attend a conference put on by CNSI.

Python Labs and Python Software Foundation

As Python grew more successful, the issue of who owned Python started to occur with CNRI. The net result was that Guido and three other guys, and various other second class citizens formed Python Labs in 2000, with Guido starting work at Zope Corporation as a director of Python Labs in October 2000. But the Python community became vocal over this, with the result that the non-profit Python Software Foundation (PSF) was formed in 2001. While the Apache Software Foundation served as a model, PSF differs in having Guido as its single leader. As another case in point,

the Perl Foundation has no members, whereas PSF does have members.

Initially, the Foundation raised funds by charging a few members $2,000 for having their logos on the Python web page. But in 2003, the first PyCon was held (previously Python conventions were done by CNSI at a high price), and between having a great venue, the conference was a tremendous financial success, that has provided funding both for the foundation and subsequent conferences.

In July 2003, Guido came to San Mateo, CA to work for Dan Farmer's new company, a venture funded startup called Elemental Security. While still in stealth mode in February 2005, Elemental Security can be described as software for implementing cross-platform policy compliance reporting and enforcement for enterprises. As one might expect, it uses considerable amounts of Python in its implementation.

The Python Language

Python uses a block structure through indentation and does dynamic type checking. Python is extensible in many lower-level languages such as C. It has an executable pseudocode, and provides high level data types and namespaces. It is a dynamically typed object-oriented language. The interpreter in Python is capable of recognizing infinite recursion, and catching it on the fly. Debugging Python programs are comparably easy since all structures and variables can be inspected. It is easy to just look at the dictionary to see how all methods are defined.

One of the flubs in developing Python was in not having a systematic way of naming procedures to avoid conflicts with other languages. It has taken years to redo the Python implementation to eliminate these conflicts.

So how does Python differ from other languages?

- With PERL, you find that you are doing a lot of slicing and dicing of your data structures, whereas Python makes it easy to implement real data structures.
- With PHP has been very successful, it really is a point solution for doing web pages. For everything else, PHP is a horrible language.
- Java and C++ provide static typing, whereas Python has dynamic typing. But when you look at programming errors, you find that static typing only lets you catch a small number of errors at compile time. Java actually came much later into the world after Python. One of the problems with Java is value bugs.
- In C++ the programmer is responsible for doing memory management, but the reality is that most programmers are not up to this challenge. While Java provides memory management, it does not work as well as Python. While Python uses reference counting to do memory management, which is a bit more painful, in practice, it is very reliable. Whereas in Java, you have multiple heaps and it is multi-threaded, with the garbage collector having its own heap!
- The Python language has similarities to both LISP and Smalltalk. However, LISP doesn't have syntax, and code != data, whereas in Python, what is data now, can be code later. Although Smalltalk has namespaces, Python makes more use of them.

Why Use Python?

People use Python because it lets them be more productive, it is more readable, easier to maintain, and it is fast and has built-in high level data types. A Python program takes 3 to 5 to 10 times fewer lines of code than the same program written in C, C++ or Java. This is because you don't have to declare variables, there is less typing, and thus fewer lines of code. It is the difference between a supertanker and a small sailboat.

Fundamentally, development time is much more expensive than cpu time. With respect to Perl and PHP, Python is the only language that remains maintainable when its code exceeds 1,000 lines. Examples of Python program with several hundreds of thousands of lines of code include Zope and Plone.

There are only so many lines of code that a programmer can grasp. The amount of code is proportional to the effort necessary to maintain it, as code becomes intertwined and dependent. When there is less code to change, it is easy to have concise code.

But there are things that Python is not optimal for, such as packet filters and MP3 codecs. It is preferable to write these applications in C or C++ and wrap Python around them.

Who Uses Python?

Python runs on a very large number of platforms, including Unix, Windows, Macs, Palm, VxWorks, and PlayStation 2 platforms, and even high-end Nokia phones! It runs in any Java environment via Jython, the Java version of Python, and in .NET environments via IronPython, the Python implementation for .NET platforms.

Examples of organizations using Python include:

- Apple (bundles Python with OS X and uses internally)
- Google (powered by Python)
- Industrial Light & Magic (film production process)
- NASA (CAD/CAM system and graphical workflow modeler)
- Disney (animation production process)
- Yahoo (group sites)
- BitTorrent (generates one-third of all Internet traffic)
- GNU mailman
- Totally Games (Star Trek Bridge Commander game)
- Real Networks (load and feature testing)

Python is used for XML processing (XML-RPC and SOAP) and in databases such as Oracle, MySQL, ODBC, and PostgreSQL. It is used in GUI programming (Qt, GTK+, Tcl/Tk, wxPython), scientific computing, testing, scripting Unix and Windoes, rapid prototyping, and teaching programming. It is used in server-side web programming (CGI, application servers) and client-side web programming (HTML, HTTP).

Python Release Process

The Python hierarchical organization starts with Guido van Rossum as BDFL (Benevolent Dictator For Life) at the top, 10 key developers or lieutenants, 100 core developers who have checkin privileges, and 1,000 contributors. As BDFL, Guido wields no actual power, except through persuasion. He primarily acts to break ties. You work your way up the hierarchic by technical merit; there is no shortcut to the top. Frequently a core developer has niche expertise on either a specific module or platform.

The PEP (Python Enhancement Proposal) is how both changes to the Python software and community are made. It provides checks and balances that ensure the slow and steady growth that the user community wants.

We are trying to have a consistent Python release schedule, with a functional release every 12 to 18 months and a bug fix release every 3 to 9 months. Our focus is to have slow and steady growth.

New Features

Originally, Python 3000 was intended to be a complete rewrite and redesign of the language. It would allow making incompatible changes in order to fix problems with the language design that aren't solvable in a backwards compatible way. Today our plan is that necessary changes will be introduced gradually into the current Python 2.x line of development, with a clear transition path that includes a period of backwards compatibility support.

- There were many mistakes made early on in developing Python. String objects should have been classes, making them a bit of a pain, having to deal with the exception machinery today.
- Arbitrary sized integers were in the original design, you had to use a specific type of integer.
- You shouldn't have a language silently give you the wrong result. For example, if you multiply two 16-bit numbers together and it generates an overflow, an exception should have been generated, or you should automatically be given a larger integer.
- Classes were an afterthought to the Python language. This resulted in all sorts of inconsistencies, that resulted in a rewrite with the 2.2 version. But you still have classic classes and new style classes to contend with.
- We are thinking about optional type declarations, since they would be slightly more static in large frameworks — letting you state that a particular class implements a particular interface.

Will there be a core GUI toolkit added to the core library? It seems more appropriate to use a 3rd party toolkit to address most of this space. Trying to come up with a general purpose GUI is a much harder problem than most people realize.

Often putting print statements in a Python program is more efficient than using the debugger. The debugger tends to be most useful in analyzing crashes — about 90% of the time, the debugger is used for post-mortem work.

For More Information

The Python Software Foundation is at www.python.org.

Guido van Rossum's home page is www.python.org/~guido.

Author's Observations

Python is market / customer driven. For example, Python runs on many different platforms because of the feedback of its initial users. Changes and improvements to Python happen when the need is such that someone is willing to implement a given change.

Guido's management style works well for an open source project. Participation is based upon merit and past work, there are no shortcuts to the top. Changes are made gradually in consensus with the Python community. Guido does not command, but rather coordinates. He encourages and supports the evolution of Python in a consistent and systematic manner.

About the Author

Mark Duncan is a marketing consultant who focuses on emerging technologies, assisting companies in entering new markets and developing new business opportunities. He can be contacted at mark@askmar.com.