21 Years of Python

From Pet Project
to Programming Language of the Year

Guido van Rossum
guido@python.org
May 2011
Disclaimer

I am speaking on my own behalf. My statements reflect my views only and do not necessarily reflect those of Google.
Pre- and Proto-History

• Significant influences:
  • Algol-60, Pascal, C
  • ABC
  • Modula-2+ and -3
  • Lisp, Icon

• Why these?
ABC (’70s, ‘80s)

• Lambert Meertens, Leo Geurts, others
• Taught Algol to scientists (for EL X-8)
• Reaction:
  – Frustrated with limitations, e.g.
    • Integer range (27 bits!)
    • Memory usage
    • Program complexity
    • Arcane I/O
ABC vs. Python

HOW TO RETURN words document:

PUT {} IN collection

FOR line IN document:
    FOR word IN split line:
        IF word not in collection:
            INSERT word IN collection

RETURN collection

homepages.cwi.nl/~steven/abc/

def words(document):
    collection = set()
    for line in document:
        for word in line.split():
            if word not in collection:
                collection.add(word)
    return collection
ABC: Stamp Out BASIC

• ABC’s target audience: professionals
  – Not professional programmers
  – But serious about programming needs
  – E.g. scientists, lab personnel

• Easy to teach, easy to learn, easy to use
  – Few constructs, combined for expressivity
  – Reuse what you already know (e.g. math, English)
  – One tool to handle everything
ABC: The Good Stuff

• Design iterations based on user testing
  – E.g. colon before indented blocks
• Simple design: IF, WHILE, FOR, ...
• Indentation for grouping (Knuth, occam)
• Tuples, lists, dictionaries (though changed)
• Immutable data types
• No limits
• The >>> prompt
ABC: Critique

- Monolithic design – not extensible
  - E.g. no graphics, not easily added
- Invented non-standard terminology
  - E.g. “how-to” instead of “procedure”
- ALL'CAPS keywords
- No integration with rest of system
  - No file-based I/O (persistent variables instead)
ABC: The Downfall

- Too early; no community
- Intended audience didn’t have the hardware
- Those who did preferred the UNIX way
- Without Internet, hard to reach users
- Lack of extensibility
- Lack of integration
- Missed the boat on exciting new stuff
Fast-forward 4 Years...

- Amoeba distributed systems group @ CWI
- Writing apps: either C or sh
- Wanted something in between
- Management vacuum...
- Started a skunkworks language project
- Amoeba @ CWI stopped, multimedia started
- Python positioned for quick experiments
Skunkworks Design Philosophy

• Borrow ideas whenever it makes sense
• As simple as possible, no simpler (Einstein)
• Do one thing well (UNIX)
• Don’t fret about performance (fix it later)
• Go with the flow (don’t fight environment)
• Perfection is the enemy of the good
• Cutting corners is okay (get back to it later)
User-centric Design Philosophy

- Avoid platform ties, but not religiously
- Don’t bother the user with details
- Discourage but allow coding to the platform
- Offer multiple levels of extensibility
- Errors should not be fatal, if possible
- Errors should never pass silently
- Don’t blame the user for bugs in Python
Language Evolution

• Core language stabilized quickly in 1990-1991
  – Earliest version did not have class statement!
• Examples of later changes:
  – augmented assignment: \( x += 1 \)
  – comprehensions: \([x**2 \text{ for } x \text{ in } xs \text{ if } x\%2 == 1]\)
  – unicode: \(u"\u2043"\)
  – new-style classes: class C(object) ...
  – decorators: @classmethod
Controlled Change

- Most development in standard library
- Enabling 3rd party libraries is major goal
- PEP process keeps changes in check
- BDFL role keeps PEP process in check
- Community feedback keeps BDFL in check
- python-dev@python.org: core development
- python-ideas@: speculative ideas
- python-list@: general help, discussion
Community: Early Days

• 1990 – internal at CWI
  – More internal use than ABC ever had
  – Internal contributors
    • open design, extensibility work!
• 1991 – first release; python-list@cwi.nl
• 1994 – USENET group comp.lang.python
  – “If Guido was hit by a bus?” thread
• 1994 – first workshop (NIST)
I love to post my first message to a group that I have waited anxiously for. Now I can unsubscribe to the listserver and perhaps lower my e-mail volume. My question is something that I sent earlier to the list and didn't receive any replies.

Question: How does one single-step a python program (note: this is not how do you debug a python statement or group of statements but how is an entire program single-stepped)

I read the pdb.doc file and have tried several things but there doesn't seem to be as easy a method as perl's '-d' command-line parameter to enter debugging mode. The only way I have found is to comment-out mainline code and:

```python
> python
Python 1.0.1 (26 January 1994)
Copyright 1991-1994 Stichting Mathematisch Centrum, Amsterdam
>>> import pdb
>>> pdb.run('import myprogram')
> <string>(0)
```

This lets me re-enter the mainline through the keyboard but I _do_ miss being able to emulate cdb inside of a running perl program through '-d'. I _must_ be missing something!
If Guido was hit by a bus?

Michael McLay View profile

What if you saw this posted tomorrow.

> Guido's unexpected death has come as a shock to us all. Disgruntled
> members of the Tcl mob are suspected, but no smoking gun has been found...

I just returned from a meeting in which the major objection to using Python was its dependence on Guido. They wanted to know if Python would survive if Guido disappeared. This is an important issue for businesses that may be considering the use of Python in a product.

I suspect that someone else would probably pick up the Python banner if Guido dropped dead of exhaustion or if he is rubbed out by a member of a rival language following. I wouldn't bring it up, but managers of projects weigh risk heavily in selecting technology and they want to know who owns Python. (They also prefer to have someone to hold accountable when something goes wrong.) There also appears to be a perception that commercial vendors are a lower risk because they have a vested interest to continue to support a product and academic research projects are a high risk because the product can disappear when a researcher's interest change or they move to a new job. (BTW, what will be the fate vpApp?)

A somewhat related topic is one of getting the official blessing of a standards organization for Python. Lots of businesses are uncomfortable using languages that are not blessed by a standards organization. Procurements are easier if you can just call out a standard as a requirement. It also makes it possible to have a third party perform conformance testing.

Turning Python into a standard might not be very difficult or costly if it could be done as an Internet standard. The rules for participating in the IETF standard process should appeal to the Python followers. Look at the following gopher document for details.

gopher://ietf.cnri.reston.va.us/11/isoc.and.ietf/ietf/standards.and.copyright.html

Is there any interest in formalizing the standard definition of Python?

Michael J. McLay
National Institute of Standards and Technology
Bld 220 Rm A357 (office), Bld 220 Rm B344 (mail)
Gaithersburg, Maryland 20899, (301) 975-4099
Community: Growth

• 1995-1999 – from workshops to conferences
• 1995 – Python Software Association
• 1997 – www.python.org goes online
• 1999 – Python Consortium
  – Modeled after X Consortium
• 2001 – Python Software Foundation
  – Modeled after Apache Software Foundation
Welcome to the Python Language Home Page!

Table of Contents

- What is Python?
- Software and Documentation
- Support and Community Resources
- Acknowledgements

Regional Python mirror sites

News and Announcements

- Come to the Sixth Int'l Python Conference
- PythonWin 1.0 is out! (Python for Win 95, NT)
- The final version of Grail 0.3 is out
- Prior news.

What is Python?

Python is an interpreted, interactive, object-oriented, extensible programming language. It provides an extraordinary combination of clarity and versatility, it is free, and it runs on Unix, PC, Macintosh, and many other systems.

Python is free and non-proprietary. Help to keep it that way by joining the Python Software Activity. You can also support the PSA by ordering Python books via our web page, and by displaying a copy of the Python logo where you use the language.

- Python Executive Summary
- Frequently Asked Questions
- Mailing lists and newsgroups
- Python compared to other languages

- The Python copyright
- The Python Software Activity
- Workshops (past, present and future)
- The PSA bookstore
- The Python logo collection
Community: Present Day

• PSF runs largest annual Python conference
  – PyCon Atlanta in 2011: 1500 attendees
  – 2012-2013: Toronto; 2014-2015: Bay area
  – Also sponsors regional PyCons world-wide

• EuroPython since 2002

• Many local events, user groups
  – e.g. baypiggies, Chicago, Atlanta, Germany, Singapore, Brazil, Argentina, ...
Community: Web Presence

- python.org
  - docs.python.org, mail.python.org, bugs.python.org, hg.python.org, planet.python.org, wiki.python.org
- Stackoverflow etc.
- TIOBE – consistently #6 or #7
  - Tiobe Programming Language of the Year (twice!)
Python 2 vs. Python 3

• Fixing deep bugs intrinsic in the design
• Avoid two extremes:
  – perpetual backwards compatibility (C++)
  – rewrite from scratch (Perl 6)
• Our approach:
  – evolve the implementation gradually
  – some backwards incompatibilities
  – separate tools to help users cope
Python 3: Status

• 2011: The Year of Python 3
• More and more 3rd party packages ported
• Python 3.2 stable and solid
• Python 2.7 is not dead yet!!!!!
Other Pythons

- Jython
- IronPython
- PyPy
Q & A