Where We Came From, What's Special About Now, Where We Might Go

Stephen J. Turnbull, University of Tsukuba

PyCon 2016 Canada, Toronto
13 November 2016

First, Who Is This For?

- This isn't for developers **using** Python
- It's for developers who **develop** Python
- Especially those who might want to **start** developing Python
- Maybe intermediate and experts might not have thought about it this way before and **find it useful** (*e.g.*, in managing own projects)
- And if you already know it all, tell me **something I missed** afterward!

Next, A Word For My Sponsors

- Part of a project studying the potential of open source development methodology and project governance in the Japanese software development industry.
- I gratefully acknowledges the support of the Japanese Ministry of Education, Culture, Science, and Technology "Kaken" grants-in-aid #23530267 and #26380287.
- After a decade of trying, the proposal was finally accepted and then renewed. I'm not sure what that means.

Who am I?

- Social scientist (economist) = University prof = part of the problem
- Subfield: information, incentives, and strategic behavior in organizations
- Based on a funded study of Python development practices
- Python developer wannabe since PEP 263

Why Me?

- Participation (peripheral) in Python development more than a decade
- I have talked extensively with several core developers
- Disclaimer: I do not claim comprehensive knowledge of FOSS practices or Python internals
- Disclaimer: I teach at Matz's alma mater (the University of Tsukuba in Japan)

Excuses

- OK, so I'm not real happy with my presentation preparation.
- First, in October I broke 7 ribs. The energy drain of just healing I wouldn't have believed until I experienced it.
- Then, Raymond Hettinger's keynote seriously overlapped about 40% of presentation.
- OK, I finished reorganizing at 8:30 this morning. Then Safia Abdalla stole about 30% of my thunder.
- Found new things to say by 11:30. Then Greg Wilson made it plain that as an untrained PhD-toting University professor I'm just not qualified to be here.
- I'm honored to be upstaged by these people, but *really*, 3 times in less than 24 hours!

What This Talk is About

- Yesterday, Raymond told you *what* core developers do. They **review**.
  - To become a "senior" core developer, you need to convince other "senior" core developers that you're a good reviewer.
- Today I want to talk about the **process** and **community** that reviewing is embedded in.
  - Yes, that means *management* and *bureaucracy*.
- I want to show you that in Python, management and bureaucracy serve the project, not the other way around.
  - I have to disagree (genially) with Safia: there *are* **rockstars**.

Python Early History

- Guido van Rossum (aka GvR aka BDFL)
  - Wanted a reasonably performant language suitable for scripts, "real" programming, and education
- Start development in late 1980s, external release in early 1990s
- Sponsor history in license documentation
- Around 1999 ESR promoted his fetchmailconf (my personal interest in Python arose from his blog on that experience)
- Red Hat chose Python for many system utilities (remember 1.5.2?)
- Added Unicode in 1.6 (2000/09)

Process in Language Development

- Python 1.6 was a "final report" on the work done with CNRI support
- After the bow to CNRI, the *first* thing announced for 2.0 was "New Development Process"—using SourceForge, opening the CVS tree, informal design of "rolling consensus", official designation of the BDFL, and PEPs; in explaining PEPs are two interesting statements:
  - writing patches is easier than design
  - design and review result in lengthy mailing list threads

  (see `https://docs.python.org/2/whatsnew/2.0.html`).
- Then Unicode and list comprehensions (PEP 202), *etc.*

Where I Came In

- Early Unicode support had a glaring flaw: Python source, *including strings and comments* had to be in ASCII, and use Unicode character literals to express everything else
  - **Aside**: Alex Martelli wrote an elegant and ardent post arguing that as a rule comments should still be in English
  - Nevertheless, there is a role for non-English comments. The need for non-English string content is obvious, however.
- The answer: PEP 263 (2001/06/06, released in Python 2.3 on 2003/07/29); Barry Warsaw called me in to consult

Values in Language Development

- Multimodal: object-oriented, functional, **imperative**
- Builtins preferred to syntax, modules preferred to builtins
- Backwards compatibility in syntax and stdlib APIs
  - Dual maintenance of 2.x and 3.y was hugely expensive
- Performance regressions acceptable in x.y.0 but should be ameliorated in x.y.1
- Style guides: Zen, PEP 7, PEP 8
- Raymond said more about this than I know

Software Processes

- Conway's Law: "organizations which design systems ... are constrained to produce designs which are copies of the communication structures of these organizations"
- You're in the Army, now!—IBM (Brooks "The Mythical Man-Month") and SEI
- Just open your source!—(Raymond "Many eyes" and the "Bazaar")
- Process, yes, but no straitjackets, please!—"Agile" methodologies
- and a legion of others.
- ESR and your visceral reaction to "SEI" notwithstanding, everybody acknowledges that "process" is needed nowadays.

FOSS and Process

- "Process Lite": Less process! More product!
- Massively distributed development: every user a contributor
- Open development
  - Every user a "literary" critic of your code
    - Raymond's point: **you** are "every user": be critical of *our* code
  - Reusability (designed-in modularity, not cargo cult) encourages quality while decreasing long-run cost

Elements of Pythonic Process: I

- BDFL
- PEPs
- Conscious automation (trackers, repos, review tools, Buildbot, “speed.python.org“)
  - We’ve on our 4th repo move: CNRI CVS → SourceForge CVS → SourceForge Subversion → self-hosted Mercurial → GitHub
- Channel discipline: c.l.py, python-dev, SIGs

BDFL

- Guido van Rossum: Benevolent Dictator For Life
  - Arbiter of "Pythonicity"—syntax, APIs, coding style
  - Skilled delegator, attracts associates who know more than he does in specialized areas
- "Founders" with veto power are legion: RMS (many), Larry Wall (Perl), Yukihiko Matsumoto (Ruby), Theo de Raadt (OpenBSD), and of course Linus Torvalds; Brooksian "one architect"
  - But often they get tired, want to do something else
  - Bequeath values (backward compatibility) and design (significant whitespace?!); discussion courtesy

Elements of Pythonic Process: II

- Delegation
  - Module "owners"
  - Benevolent Dictator for One PEP (BDF1P)

Delegation I: Owners

- The "lieutenant" model a la Linux
- But area of responsiblity more self-contained than "subsystem": elementTree library (XML processing), the workhorse "timsort"
  - Raymond has several
  - Ownership is not absolute ("rolling consensus")
- Important: contribution of a whole module doesn't automatically make you an owner, it has to be negotiated.
  - Different from XEmacs

Delegation II: BDF1P

- Fairly recent **invention**
- Benevolent Dictator for One PEP (today "BDFL-delegate" is probably most commonly used term)
- Delegation is temporary
- Expertise relevant to a salient problem *today*—history of writing PEPs and BDF1P helpful but not required
- Broadens delegation to developers who aren't comfortable with the responsibility of "owning" something, and to aspects of the project which can't be "owned"

Zen

- "The Zen of Python" by Tim Peters; aka python -m this
- A list of sayings that constitute a style guide for Pythonic *design* (PEP 8 explains the "standard" readability style)
- Personally, I *like* the Zen as a framework for program design, but that doesn't matter. What's important is that it makes Python readable and composable, because people tend to express themselves in the same way.

PEPs

- Python Enhancement Proposals: formal proposals of changes to the language or module additions to the standard library
  - many changes to the language (syntax) are PEP'ed; most evolutionary changes to the stdlib are not
- Like IETF RFCs; also Scheme SRFIs
- IETF-like acceptance criteria ("rough consensus and running code")
- Compact record of controversies, the arguments, and the outcomes
  - No presumption of acceptance; some "written for rejection"
- Acts as a brake, helps maintain backward compatibility, provides a point for discussion of migration

Conscious automation

- 2 PEP'ed VCS migrations, and a third upcoming (to git/GitHub)
- mailing lists (what about web forums?)
- issue trackers: most bugs and simple patches are handled here, rather than on the mailing list
- review tools (Reitveld)
- automated testing (buildbot)
- proposals for further automation (continuous integration)

Channel discipline

- Courtesy is demanded and enforced gently but firmly; ostracism is very rare; appropriate channels:
  1. Python list (python-list@python.org, comp.lang.python)
  2. core-mentorship, python-tutor
  3. python-ideas, python-dev
  4. Issue-tracker and Rietveld
  5. SIGs: distutils, email, ...

Lessons

- Delegation frees leader time, strengthens associates' leadership skills
- Collaboration in a diverse dispersed community is hard. There is substantial dissatisfaction with the communication channels, but no consensus on how to improve the situation
  - Python is a huge community (thousands of active posters)—YMMV on specifics
- For "platform" tools, emphasis on backward compatibility from early times is very important
  - GvR and others still recall the "Boolean" mini-fiasco with pain
  - Process automation, especially tests, helps a lot

Where We Are Now: Dev Culture

- The consistent long-term *self-awareness* is **special**
- Current focus is on improving existing tools
- Mailing lists are *so* **1990s** (sprint on Mailman 3? please?)
- Inclusion of "traditional developer personalities" is pretty good
- CoC and "polite" communication prevalent and widely appreciated
  - Check out PyCon.CA's session chairs!!

Where We Are Now: Feature Complete?

- 16 new PEPs in 3.6—the delta is too big to disappear
  - If you went to Brett's talk yesterday, you know that Python 3.6 includes 16 new PEPs, and many features that don't require PEPs. The GIL still exists, the PEPs included in 3.6 are conservative and may get extensions—development of new Python features is not dead
  - But almost everything you might want to do in Python (*i.e.*, most anything you might want to do with a computer), you already can do, albeit at a performance cost.
  - Fixing the "can't do at all" and "can't do efficiently enough" issues is hard; *e.g.*, Larry Hastings says "you have to be **this tall** to sprint on GILectomy"
- So it's like getting a new drug approved:
  - must be backward compatible API-wise; must clear the improvement bar

Attention to Community Building

- If it's hard to contribute code, what to do? Build community!
- Legal: Contributor Agreement, better licensing, PSF incorporation (long since done)
- PyCon (would happen anyway)
- core-mentorship—overcoming bureaucracy and different POV
- CoC

Where We're Going I: Diversity

- Diversity is a value: Guido says so (several keynotes, etc)
  - Guido isn't going to push it organizationally; he prefers to mentor individuals
  - PyCon is already pretty good for gender balance, CoC complaints are way down (CoC enforcement was always low, though occasionally needed).
- Core dev is (considered by some to be) a disaster area
  - Eg, PyCon 2016 Language Summit: 80 participants, 2 women
  - Ethnicity, sexual orientation *may* be better represented, but no polls were taken
- Concrete plans: keep talking, core-mentorship
  - Aside from gender and LGBT: nationalities, age, ...
  - There's interest, but if you don't already know core folks to talk to about it, feel free to send ideas, anecdotes (anonymized, please), rants (anonymized, please), etc to me
- Random ideas: yesterday I grabbed James Powell because I think I know some people who might sponsor a PyData in Japan, and got an education on how these conferences can improve inclusiveness

Where We're Going II: Overload

- Raymond mentioned this; nothing we can do technically will fix it—a finite set of developers can't scale
- Over the last couple of years a couple of core developers have actually unsubscribed from python-ideas and even python-dev
  - traffic has exploded
  - non-experts post a lot to highly technical discussions, frustrating people who are very busy but must contribute to a high priority issue (eg, security)
- Closed lists aren't really a solution—even the Language Summit (venue limitations) has been opened up to lurkers with Jake Edge's articles on LWN.net: https://lwn.net/Articles/688969/
- Technologies such as Discourse and github issues are being considered

Where We're Going III: Financing FOSS

- Warning: **not limited to Python**
- Several Python community senior developer have been tweeting about financing FOSS lately; it's not just the traditional whining
- Some major companies have a standing policy of giving developers some paid time for FOSS contributions (not necessarily related to the day job), but covers a small minority of developers
- Community-run FOSS projects are becoming more sophisticated, using automated testing and continuous integration; popular projects may need CDNs for distribution; administrative expenses

Python 3

- Not really in scope today, but feel free to ask me out-of-band

Thank You For Listening!

- Questions? Comments?