EuroPython Keynote

June 26, 2002

Guido van Rossum
Director of PythonLabs at Zope Corporation

guido@zope.com
guido@python.org

Recent Releases

- Python 2.2
  - iterators!
  - generators!!
  - new-style classes!!!
  - and too much to summarize here...
- Python 2.1.3
  - bug fix release for 2.1; focus on stability
- Python 2.2.1
  - bug fix release for 2.2; ditto
- What's with this stability focus...? (see later)

ZOPE

Python Organizations

- Python Software Foundation
  - www.python.org/psf
  - US non-profit for research and education
  - owns the current Python copyright
  - looking for donations and sponsors
- Python Business Forum
  - www.python-in-business.org
  - EU non-profit for businesses based on Python
  - plans:
    - Python in a tie
    - Compile farm

Python in a Tie

- Result of stability discussion on c.l.py
- Plan:
  - pick a release and maintain it for 18+ months
  - bleeding edge development releases continue
- Purpose:
  - have a reliable target for commercial users
  - stability more important than latest features
- Which release????
  - Python 2.2.x most likely candidate
- See BOF on Friday

ZOPE

Compile Farm

- Joint venture of PBF and Lysator
  - Lysator: oldest Swedish computer society
  - Lysator owns a very diverse hardware farm
  - PBF provides motivation, funding
- Goals:
  - testing on many platforms
    - Python-in-a-tie as well as bleeding edge code
    - core Python as well as 3rd party extensions
  - build binary releases for Python-in-a-tie
    - hopefully "sumo releases"
- See BOF on Friday

ZOPE

Python.Org HTTP Statistics

- May 2002
  - 7.9M HTTP requests from 257K hosts
    - 291K hits for "/"
  - 52K downloads of Python 2.2.1
    - about 70% Windows installer
- Feb 2001
  - 5.5M HTTP requests from 164K hosts
    - 212K hits for "/"
  - 23K downloads of Python 2.0
    - over 70% Windows installer

What's With SourceForge?

- Sad to say, unhappy with many services
  - main problem: SF no longer listens
- CVS still fine
- Mailing lists: Geocrawler archives stink
- Moved dev guide to www.python.org/dev
- Moved file downloads to www.python.org
  - (too much work to upload, less used)
- Issue trackers: lots of issues...
  - watch this space

ZOPE

"No Uncontroversial Topics"

- The yearly recap of a recent flame war
- It's a growth opportunity!
- QOTY:
  - "When a group becomes large enough there are no uncontroversial topics any more."
    - Erik van Blokland (in personal email)
- This year's topic:
  - to bool or not to bool

ZOPE

Why bool()?

- I always regretted having left it out
- If it's not built-in, people define their own
- Explicit is better than implicit: "return True"
- A bool result is distinguished in output
  - >>> x == y
    True
    >>>
- "bool(x)" normalizes Booleans
  - was "not not x"
- RPC tools can special-case Booleans

Why not bool()?

All misunderstandings (in my opinion)

- Will "if x:" require x to be a bool? (***Never!***)
- Some people write "if x == True:" (Yuck)
- "No function should return a bool" (Huh?)
- It's confusing to teach
  - I don't buy this:
    - You need to explain the Boolean concept anyway
    - You need to pick representatives anyway
    - You need to explain that (almost) all types have a Boolean interpretation anyway

How to bool()?

- bool is a new built-in type
- True and False are the only values
  - singletons like None ("dualtons"?)
- Cannot be subtyped
- Subtype of int, for compatibility
  - True + 1 == 2
  - True == 1
  - str(True) == 'True' # The only incompatibility
  - will stay this way in Python 3.0
    - it's useful and harmless

Lessons Learned

- Everything is controversial
- Anticipate potential misunderstandings
  - explain in advance
  - I *thought* the PEP was clear - not so :-(
- In the end, do what's right

The Future: Python 2.3

- No new syntax, except yield w/o __future__
- Library focus, e.g.:
  - support extended slices, e.g. "dlrow olleh"[::-1]
  - bool() and enumerate()
  - more callable types; basestring
  - import from zip files
  - timeouts for sockets
  - logging module
  - gnu_getopt and option parser modules
  - new compiler package
  - berkeleydb module
- Fixing bugs
  - e.g. disappearing unwise.exe

ZOPE

PendingDeprecationWarning

- Discourage certain things in new code
  - But don't warn about them normally
    - Because they are too common
- Potential examples:
  - string module (use string methods)
  - types module (use built-in type names)
  - has_key (use 'in' operator)
- To get the warning:
  - python -Wall # also warns about overflows
  - python -Wall::PendingDeprecationWarning

Python 2.3 Miscellanea

- Make None a keyword?
  - can't do this at once
    - it's surprising how much code would break
      - typical idiom: def func(x, y, None=None): ...

        (trying to save repeated lookup time of built-in None)

- Stage 2 of int/long integration
  - add warning for hex/octal of negative short ints
  - add warning for certain left-shifts of short ints

2.3 Release Schedule

- Surprise: we have none!
- Focus on feature completeness, not dates
- Hope: alpha before OSCON, final in 2002
- See PEP 283 for details

Pace of Change

- Users demand a stop to all new features except for their personal favorite
  - this contradiction seems unavoidable
- What do do about this?
- Is Python-in-a-tie sufficient?
- "Would you rather..." [idea due to Barry]
  - learn more syntax or use a library module?
  - understand a deep concept or have fuzzy rules?
  - fix design mistakes or be backwards compatible?
  - use indentation or braces? :-)

Example: String Interpolation

- Problem: % interpolation is cumbersome
  - print x, "+", y, "=", x+y
  - "%s + %s = %s" % (x, y, x+y)
  - "%(x)s + %(y)s = %(z)s" % vars()
  - str(x) + " + " + str(y) + " = " + str(x+y)
- Solution 1: "$foo".sub() # runtime
  - "$x + $y = $z".sub()
- Solution 2: x"$foo" # compile-time
  - x"$x + $y = $(x+y)"
- Solution 3: x"`foo`" # compile-time
  - x"`x` + `y` = `x+y`"

ZOPE

Python 3.0

- No release schedule either :-)
- Not within two years
- Question: what to focus on???
- Zope 3 experience may be relevant
  - Rebuild from scratch
    - Refactor mercilessly during development
    - No concern for backwards compatibility
      - But learn from past: good ideas, bad ideas
    - Use coding "sprints"
  - Later, add compatibility (Zope 3x -> Zope 3)
  - Or: Later, merge best features back into 2.x

Open Mike

It's your turn!

ZOPE