Python 3000 and You

Guido van Rossum

PyCon

March 14, 2008

Why Py3k

- “Open source needs to move or die”
  - Matz (creator of Ruby)
- To fix early, sticky design mistakes
  - e.g. classic classes, int division, print statement
- Changing times: time/space trade-off
  - e.g. str/unicode, int/long
- New paradigms come along
  - e.g. dict views, argument annotations

Major Breakages

- Print function: *print(a, b, file=sys.stderr)*
- Distinguish sharply btw. text and data
    - b"…" for bytes literals
    - "…" for (Unicode) str literals
- Dict keys() returns a set view [+items()/values()]
- No default <, <=, >, >= implementation
- 1/2 returns 0.5
- Library cleanup

Long Anticipated Breakages

- Kill classic classes
- Int/long unification
- Kill string exceptions
    - Exceptions must subclass BaseException
- Raise syntax: raise *Exc*(*args*) [from *tb*]
- Except syntax: except *Exc* as *var*:
    - Also makes *var* undefined at block exit

Many Small Breakages

- Remove cmp() builtin
- Remove cmp arg to sorted() and list.sort()
- Kill map(None, …); use zip()
- map(), filter() return iterators
- Disallow int('- 1')
- Explicit relative import
- Removed `…`
- Removed <>
- None, True, False are keywords
- New keywords as, with, nonlocal
- raw_input() -> input()
- xrange() -> range()
- Changed metaclass syntax
- Kill compiler package
- Kill tuple parameter unpacking
- New octal literals (0o777)
- .next() -> .__next__(); next() built-in
- .func_code -> .__code__
- Removed dict.has_key()
- Removed dict.iteritems() etc.
- Removed sys.maxint; use sys.maxsize
- Removed reload(); use imp.reload()
- Removed reduce(); use functools.reduce()
- Removed apply(); use f(*args)
- Removed callable(); use Callable ABC
- Removed basestring; use str
- .__nonzero__() -> .__bool__()

- Etc, etc.

Major New Features, *e.g.*

- Argument annotations:
  - def f(a: 2*2, b: 'hello') -> 42: …
- Abstract Base Classes
- Extended iterable unpacking:
  - a, b, *x, y = range(5)   # 0, 1, [2, 3], 4
- New str.format() method:
  - "Got {0} {kind}".format(42, kind='bugs')
    - "Got 42 bugs"

Many Smaller Improvements

- I/O no longer depends on C <stdio.h>
- Source code encoding defaults to UTF-8
- Allow Unicode letters in names
- Class decorators
- __prepare__() method on metaclass
- Nonlocal statement
- Keyword-only arguments
- Default implementation of != negates ==
- Binary literals 0b10101, bin() function
- Mutable bytes type (bytearray)
- Overloadable isinstance(), issubclass()
- fractions.py defines Fraction type
- super() without arguments works
- Set literals and set comprehensions
- Dict comprehensions
- New exception attributes:
  - __traceback__
  - __cause__
  - __context__
  - Exceptions aren't sequences; use e.args
- Abstract Base Classes:
  - In abc.py: infrastructure
  - In collections.py: Set, Sequence, Mapping, MutableSet etc.
  - In numbers.py: Number, Complex, Real, Rational, Integer
  - In io.py: IOBase and more
- Etc, etc.

What’s In It For You

- More predictable Unicode handling
- Smaller language
    - Makes “Python fits in your brain” more true
- TOOWTDI (There’s Only One Way To Do It -- The Zen of Python)
- Common traps removed
- Fewer surprises
- Fewer exceptions

Enables Future Evolution

- Examples:
  - Argument annotations
  - print() function
  - str.format() method
  - Abstract Base Classes
  - Unicode letters in names

The ‘2to3’ Tool

- Context-free source code translator
- Handles syntactic changes best
    - E.g. print; `…`; <>; except E, v:
- Handles built-ins pretty well
    - E.g. d.keys(), xrange(), apply()
- Doesn’t do type inferencing
- Doesn’t follow variables in your code

When To Switch

- No hurry! 2.6 will be fully supported
  - Could be 5 years or more
  - Release of 2.7 possible, maybe even 2.8
- Switch when both of these are true:
  1. You're ready
  2. All your dependencies have been ported
- There are tools to help you switch!

Be Prepared

- Start writing future-proof code for 2.5
- Don’t bother with the trivial stuff though:
    - The 2to3 tool will handle this
    - E.g. callable(), `…`, <>, L suffix
- Instead, focus on what 2to3 *can’t* do:
    - Stop using obsolete modules
    - Start using iterators and generators

Things You Can Now

- Inherit classes from object
- Use dict.iterkeys() etc.
- Use xrange(), sorted(), zip()
- Use // for floor division
- Inherit exceptions from [Base]Exception
- Use rich comparisons (__eq__ etc.)
- Etc., etc.

What About Text Handling

- Yes, it’s a difficult issue…
- Expect more help by this summer
- Isolate handling of encoded text
- In 2.6:
  - Use bytes and b'…' for all data
    - Knowing these are just aliases for str and '…'
  - Use unicode for all text

The Role of Python 2.6

- Stable, compatible, supported!
- Many 3.0 features backported
  - But not the new text / data distinction
  - More volunteers needed!
- Warns about non-3.0-isms with ‘-3’ flag
  - Especially for things that 2to3 cant fix

Transition Strategies

- If you can: burn your bridges!
- Otherwise:
  - Port to 2.6 first
  - Maintain 2.6 and 3.0 version together
  - Derive 3.0 version from 2.6 source
    - Using 2to3 whenever you can
    - Using forked code only where you have to
  - Enables feature parity of your app or lib

Porting C Extensions

- Porting document TBD
- You’ll probably have to fork your code
    - Or sprinkle with #ifdef
- We try to delete APIs or add new ones
    - But not break existing APIs that stay
- 2.6: str, unicode -> PyString, PyUnicode
- 3.0: bytes, str -> PyString, PyUnicode

Release Schedule

- Just released:
  - 2.6a1
  - 3.0a3
- Monthly lightweight releases
- Betas starting soon
- 2.6 *and* 3.0 final around in August

Wrapping Up

- Don’t fear Py3k!
  - Have fun with the new features
  - Enjoy fewer bugs, traps, surprises
- Take your time to convert!
  - You will get lots of time, lots of help
- 2.6 will be stable, compatible, supported
  - For many years to come!

Resources

- Docs: docs.python.org/dev/3.0/
    - docs.python.org/dev/3.0/whatsnew/3.0.html
- Download: python.org/3.0/
- PEPs: python.org/dev/peps/pep-3000/
- Mailing list: python-3000@python.org
- Subversion:
    - svn.python.org/view/python/branches/py3k/

“I Have This Great Idea…”

- If your idea hasn’t made it into 3.0 yet, it’s more than likely too late to get it in
- Current focus is on:
  - Fixing bugs
  - Improving performance
  - Backporting to 2.6 (without breaking stuff)
  - Adding more “-3” warnings to 2.6