# Python 3000

http://www.aleax.it/gs_py3k.pdf

Alex Martelli,
w/many thanks to:
Guido van Rossum

# Audience levels for this talk

守

Shu

("Retain")

破

Ha

("Detach")

離

Ri

("Transcend")

# Why Py3k

- "Open source needs to move or die" [Matz]
  - kind of like... sharks!-)

- Fix early design mistakes
  - classic classes, int division, print stmt, ...
- Time/space trade-offs change w/time
  - str/unicode, int/long, ...
- New paradigms

  - iterables vs lists, arg annotations, ABCs, ...
- So: need for backwards-incompat changes
  - AKA... "breakages"!-)

# Major Breakages

- Print function: print(a, b, file=sys.stderr)
- Distinguish sharply between text and data
  - b"..." for bytes literals
  - "..." for (Unicode) str literals
- Dict keys() gives set view (items, values too)
  - use list() in the rare case you need one!
- No default <, <=, >, >= implementation
  - ordered-comparison only when explicit
- 1/2 returns 0.5 (not 0!)
  - use 1//2 for explicit truncation



Google

# More Break[age] Dance

- No more "classic classes"
- Int/long unification
- No more "string exceptions"
  - Exceptions must subclass BaseException
- Exc syntax:
  - raise Exc(args) [from tb]
  - except Exc as var: ...

# Many Small Breakages

- **remove:** `...`, <>, apply, basestring, buffer, callable, cmp, input, reduce, reload, cmp arg to sort, dict.has_key, map(None...), tuple args, sys.maxint, lots of stdlib (compiler, gopherlib, md5, ...)
- **kw:** None, True, False, as, with, nonlocal
- **rename:** xrange→range, _nonzero_→_bool_, raw_input→input, .next→_next_, func_xxx →_xxx_
- **&:** relative import, return iterators from map/filter, metaclass syntax, octal literals, &c, &c

# But...: New Features!

- 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"
- dictcomps, set lits & comps, binary lits, bin(), class decorators, _prepare_, stdlib stuff ...

Google

# Why move Py 2 -> Py 3

- Simpler, richer Unicode handling
- Smaller, simpler language
  - makes "Python fits your brain" more true!
- OOWTDI (Only One Way To Do It)
- Fewer surprises, exceptions to rules, traps and pitfalls, and more generally less cruft
- why wait a bit, if py3 is a better language?
  - mostly, if you need 3rd party extensions or tools that don't support py3 yet!
  - stay portable to .NET, JVM, embedded...



Google

# The '2to3' Tool

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



Google

# When To Switch

- No hurry! 2.6 is (& will be) fully supported
  - Probably 3-4 years or more
  - Release of 2.7 likely, 2.8 possible
    - 2.9 less likely; 2.10 is right out;-)
- Switch when both of these facts hold:
    - 1. You're ready
    - 2. All dependencies have been ported
- Tools like 2to3 to help you switch!

# Be Prepared

- Start writing future-proof code for 2.5/2.6
- Don't bother with the trivial stuff:
  - The 2to3 tool handles much of this
  - E.g. callable(), `...`, <>, L suffix in long
- Focus on what 2to3 can't do:
  - Stop using obsolete modules &c
  - Start using iterators and generators



Google

# Things You Should Do Now

- Inherit all classes from object
  - and all exceptions from [Base]Exception
- Use dict.iteritems() etc.
- Use xrange(), sorted(key=...)
- Use // for floor division
- Define rich comparisons (__eq__ & friends), NOT __cmp__
- in general: use Python 2.5 / 2.6 as such, do NOT rely on their "legacy" features
  - ...whether you plan a Py3 port or not!-)



Google

# What About Text Handling

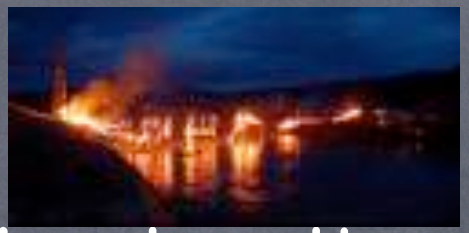
- No silver bullet
- Isolate handling of encoded text
- In 2.6:
  - Use bytes and b'...' for all *data*
    - Even though they just alias str and '...'
  - Use unicode and u'...' for all *text*
- In 2.5: '...' for data, u'...' for text

# Python 2.6

- Stable, compatible, supported!
- Many new 3.0 features backported
  - But not the new text / data distinction
- Warns about non-3.0-isms with '-3' flag
  - Especially things that 2to3 can't fix

# Transition Strategies

- If you can: burn your bridges! :-)
- Otherwise:
  - Port to 2.6 first
  - Maintain 2.6 and 3.0 version together
  - *a good suite of tests* is crucial!!!
  - Derive 3.0 version from 2.6 source
    - Use 2to3 whenever you can
    - Fork code only where you have to



# Porting C Extensions

- Fork your code or sprinkle with #ifdef
- We try to delete APIs or add new ones
  - But not break existing APIs that stay
  - # & type of arguments won't change
- 2.6: str, unicode -> PyString, PyUnicode
  - PyBytes is an alias for PyString
- 3.0: bytes, str -> PyBytes, PyUnicode
- Also: PyInt vs. PyLong

# Release Schedule

- 2.6 and 3.0 final: both on 10/01/2008
- 2.6.1: 12/04/2008
- 3.0.1: 02/13/2009
- current: 3.1 α 2 (04/04/2009)

- future: 3.1 β early 05/09, final late 06/09
  - collections.OrderedDict, importlib, lib updates (IO, email, ipaddr, ipaddr, etree...)
- future: 2.7: no schedule fixed yet

# Resources

- http://python.org/ & links therefrom
- Books:
  - Python 3 for Absolute Beginners (APress)
    - haven't seen it; target date Apr 20
  - Programming in Python 3 (AW)
    - haven't reviewed it; out since Dec 26
  - Python 3 in a Nutshell (O'Reilly)
    - Anna & I started on it, "rough cut" PDF version by Christmas (no promises!-)
  - Dive into Python3: diveintopython3.org



Google

# Questions & Answers

http://www.aleax.it/gs_py3k.pdf

Q?

A!