The State of Python

… and the web

Armin Ronacher // @mitsuhiko
Who am I

- Armin Ronacher (@mitsuhiko)
- Founding Member of the “Pocoo Team”
- we're doing Jinja2, Werkzeug, Flask, Pygments, Sphinx and a bunch of other stuff.
Crossroads
Python 3

- The Elephant in the Room
Python in 2011

- The big Python 3 versus Python 2 debate
- PyPy is making tremendous progress
Recent Python News

- Unladen Swallow is resting
- Python 3.2 was released
- Python's packaging infrastructure is being worked on.
- `distutils2 / packaging in Python 3`
Recent PyPy News

• PyPy gets experimental support for the CPython C API
• PyPy got 10,000$ by the PSF
• PyPy 1.5 released
PyPy Right Now

- “Python written in Python”
- PyPy trunk 3.7x faster than CPython over a wide variety of benchmarks
- Up to 40x faster for certain benchmarks
- Compatible with Python 2.7.1
Really Fast

- http://speed.pypy.org/
Things that will break

- There is only experimental support for the Python C API and it will always be slow.
- Different garbage collection behavior, no reference counting.
Things that work

- Django
- Flask
- ctypes
- pyglet
- twisted
- sqlite
The Bonus

- Sandbox support
- Stackless execution mode
- A .NET backend
Python 3
Python 3 is …

• … where all new language developments are happening
• … adding unicode to the whole stack
• … cleaning up the language
• … breaking backwards compatibility
The Good Parts

- Introduces unicode into exceptions and source compilation as well as identifiers
- Greatly improved IO API regarding unicode
- New language constructs
- Implementation cleaned up a lot
New Constructs

- Extended iterable unpacking
- Keyword only arguments
- nonlocal
- Function parameter and return value annotations
Improved Things

- print as a function
- Improved syntax for catching and raising exceptions
- Ellipsis (...) syntax element now available everywhere
Different Behavior

• More powerful metaclasses (but removed support for some tricks people relied on*)

• List comprehensions are now from the behavior much closer to generator expressions

  • * don't abuse undocumented “features”
Warts removed

- Argument unpacking
- Unused nested tuple raising syntax
- Longs no longer exposed
- Classic classes gone
- Absolute imports by default
- Obscure standard library modules
New in 2.6/2.7

- Explicit byte literals, make upgrading easier
- Advanced string formatting
- Print as a function
- Class decorators
- New IO library
New in 2.6/2.7

- The multiprocessing package
- Type hierarchy for numbers
- Abstract base classes
- Support for fractions
Going Forward
Beauty or Speed

• Right now it's a decision between the beauty of the code (Python 3) or the raw performance (PyPy).

• PyPy itself will probably always be written in Python 2, but the interpreter might at one point support Python 3.
Library Support

- Numeric libraries work great on Python 3 and benefit of improvements in the language.
- PyPy still lacks proper support for the C-API of Python.
Predictions

• Most people will write their code against 2.7 with the intention of supporting PyPy.

• Libraries that require the Python C API will become less common

• We will see libraries that support targeting both Python 2.7 and Python 3.x.
Python and the Web
WSGI

- New revision for Python 3
- There is some work done to port implementations to Python 3
- No longer something people actively care about. “It works”
New Developments

- Improvements to PyPy's support for database adapters
- Improvements in template compilation to take advantage of PyPy's behavior.
- Porting some libraries over to Python 3.
Making Python 3 work
Python 3 can work

- Start porting libraries over.
- Issues with Python 3 will only be resolved if people actively try to port.
- The higher level the application, the easier to port. Libraries are the culprit.
And it's not hard

• When you're at the port where you can drop Python 2.6 support, you can write code that survives 2to3 mostly without hacks in the code.

Frameworks
WSGI works out well in practice.

Pylons and BFG -> Pyramid, nice introduction into the ZOPE world.

Less and less framework specific code out there, easier to reuse.
Less Frameworks*

- Django
- Pyramid
- Flask
- Bottle
- web.py
Low Level

- Werkzeug
- WebOb

- these two might actually merge at one point in the future
Frameworks are Good

- New frameworks are necessary to explore new paradigms and concepts.
- It's surprisingly easy to switch frameworks or parts of frameworks in Python.
- Frameworks are merging and evolving.
Thank you
Contact / Slides

- Armin Ronacher
- @mitsuhiko
- http://lucumr.pocoo.org/
- http://lucumr.pocoo.org/talks/ <- Slides