ZOPE CORPORATION

"The State of the Python Union"

Python10 - Alexandria, VA - February 7, 2002

Guido van Rossum

Director, PythonLabs at Zope Corporation

guido@python.org

guido@zope.com

Overview

- Toy of the year
- Where are we now
- Parade of the PEPs
- Open mike

ZOPE

Toy of the Year

- Warning: shameless plug ahead

- "familiar" Linux on iPAQ
- Python as main development language
- Go to handhelds.org

ZOPE

Where Are We Now

- Python 2.2 released (Dec 21)
  - iterators, generators
  - type/class unification
  - new bugs :(
- Working on Python 2.2.1 (within a month)
  - fix the bugs :)
- Planning Python 2.3 (this summer)
  - library improvements
  - next phase of int/long unification
  - WHAT ELSE?!?!

ZOPE

Too Many Choices!

- New languages features
- Extend import
- Expand library
- Improve performance
- Restructure implementation
- Other wild ideas

- Too many choices! What to work on????

ZOPE

New Language Features?

- Syntactic sugar for new class features?
  - static/class methods
  - slots
  - properties
  - super
- Dict comprehensions? (PEP 274)
- Generator comprehensions?
  - [yield x**2 for x in range(10)]
- TBL's triple graph notation?
  - {sky color blue, gray; sea color green}

ZOPE

Static Methods, Class Methods?

- class C: # extra keyword after def
  def static foo(arg1, arg2): ...
  def class cfoo(cls, args): ...
- class C: # list of properties
  def foo [static](arg1, arg2): ...
  def foo [class](arg1, arg2): ...
- class C: # keyword in front of def
  static def foo(arg1, arg2): ...
  class def cfoo(cls, arg1, arg2): ...
- class C: # implicit by lack of 'self'
  def foo(arg1, arg2): ...
  def cfoo(cls, arg1, arg2): ...

ZOPE

Slots???

- class C:
  slot a, b, c
- class C:
  a, b, c: slot
- class C:
  slots:
  a: int
  b: str
  c
- Is 'slot' the right word?

Properties?

- class C:
  property a:
  """Computed variable a"""
  def get(self): ...
  def set(self, value): ...
  def delete(self): ...
- class C:
  properties:
  a:
  def get(self): ...
  b:
  def get(self): ...
  def set(self): ...

ZOPE

Super?

- class C(A,B):
  def save(self, file):
  ...dadada...
  super.save(file)
  ...tatata...

ZOPE

New Keywords?

- super, property, slot to become keywords?
- Need to allow "old" (2.2) usage too!
- Context-sensitive keywords???
- E.g.
  - property is only a keyword at start of line
  - static is only a keyword if followed by dot

Extending Import?

- ihooks is for all practical purposes dead :)
- imputils might as well be dead :(
- import from zip file is very much alive!

- Webizing Python (TBL)?
  - add "http://python.org/Python2.2/" to sys.path?
  - Mike McLay asked for this 5 years ago :)

Expand Library?

- Logging module?
- PyChecker?
- Persistence, ZODB?
- YAPPS or SPARK?
  - Expose pgen?
- ...?
- DB-API 3?
- TBL's triple store API?

ZOPE

Improve Performance?

- 2.2 slower than 2.1 slower than 2.0 slower than 1.5.2!
  - (But sometimes 2.2 is fastest!)
- Improve virtual machine?
- PyMalloc?
- "Low-hanging fruit" in bytecode
  - Faster globals
  - Recognize built-ins?
  - Faster attributes???
- Come to session with Jeremy and Skip

ZOPE

Restructure Implementation?

- More can be done in Python
  - saves C code
- Much already exists as alternative!
  - print tracebacks
  - bytecode compiler (from abstract syntax tree)
  - interactive command line
  - import
- What else?
- Careful: code may be compromised

ZOPE

Parade of the PEPs

- Switch to IE window

ZOPE

Open Mike

- Your turn

ZOPE

And Don't Forget...

```
>>> import this
The Zen of Python, by Tim Peters

Beautiful is better than ugly.
Explicit is better than implicit.
Simple is better than complex.
Complex is better than complicated.
Flat is better than nested.
Sparse is better than dense.
Readability counts.
Special cases aren't special enough to break the rules.
Although practicality beats purity.
Errors should never pass silently.
Unless explicitly silenced.
In the face of ambiguity, refuse the temptation to guess.
There should be one-- and preferably only one --obvious way to do it.
Although that way may not be obvious at first unless you're Dutch.
Now is better than never.
Although never is often better than *right* now.
If the implementation is hard to explain, it's a bad idea.
If the implementation is easy to explain, it may be a good idea.
Namespaces are one honking great idea -- let's do more of those!
>>>
```

ZOPE