# How to write cross-interpreter programs

Maciej Fijałkowski

merlinux GmbH

Pycon 2010, February 19th 2010, Atlanta

# This talk

- most people only target CPython (or Jython or IronPython)
- sometimes, you want your program to run on each of those
- libraries are more often cross-interpreter

# This talk

- most people only target CPython (or Jython or IronPython)
- sometimes, you want your program to run on each of those
- libraries are more often cross-interpreter
- hope to give you more reasons tomorrow

# This talk

- most people only target CPython (or Jython or IronPython)
- sometimes, you want your program to run on each of those
- libraries are more often cross-interpreter
- hope to give you more reasons tomorrow
- won't talk about py3k

# Have a way to verify compatibility

- extensive test suite
- good coverage

# Exceptions

- TypeError vs AttributeError change often between implementations, even CPython versions
- don't rely on exception string messages (they may differ)

```
try:
   ...
except ImportError, ie:
   if str(ie) != '...':
      raise
```

# Subclasses of builtin types and overriding

- in general overridden methods on subclassed builtin types are not invoked by preexisting other methods

```
class d(dict):
    def __getitem__(self, e):
        ...
```

- would `keys()` go via this getitem?
- there can be corner cases, for example when there are more than one object involved
- tests are your friend

# Access to 3rdy party libraries

- there is no good story here
- ctypes based access is going to be supported by all Pythons
- are there pure Python replacements/options?
- separate out dependencies/especially optional ones

# Don’t rely on refcounting

- example `open('x', 'w').write('stuff')`
- on refcounting, flushes file immediately
- on any other GC, it might be deferred for a while
- the single most-common problem when fixing twisted

# __del__

- in case of resurrection CPython will call `__del__` multiple times, other Pythons exactly once
- cycles with __del__s are not collected by CPython, PyPy breaks them randomly instead
- in PyPy (xxx Jython?) __del__ cannot be attached to classes after creation

# Use new-style classes

- 3.x ready
- faster on PyPy, too

# sys.prefix

- implementations may have different installation layouts
- open issue, at least for PyPy, has compatibility consequences with setuptools

# IO bytes vs unicode

- convert/decode as soon as possible, keep text and bytes apart
- for 2.x Pythons use str for bytes and unicode for text
- the distinction is deeper in 3.x (str is unicode, bytes exist with slightly different interface than old str)

# Obscure corners

- non-string keys in type dictionaries
- introspection results, implementation objects (e.g. builtin methods etc), may have different types
- exact naming of things (like list-comprehension variable)