

Python3 support in the RDKit

Riccardo Vianello

3rd RDKit UGM – Darmstadt, Oct 22-24 2014

Python3

- Designed to be a better Python language
 - Fixes some known flaws
 - Removes some deprecated features
 - (also introduces some original improvements)
-
- Very similar to Python2, but not backward compatible

Python3 is almost 6 years old now

Since 2008 there have been 4 Python3 releases
and 1 single Python2 release

Python 2.7 was to reach EOL in 2015
(maintenance extended to 2020)

“There will be no Python 2.8”

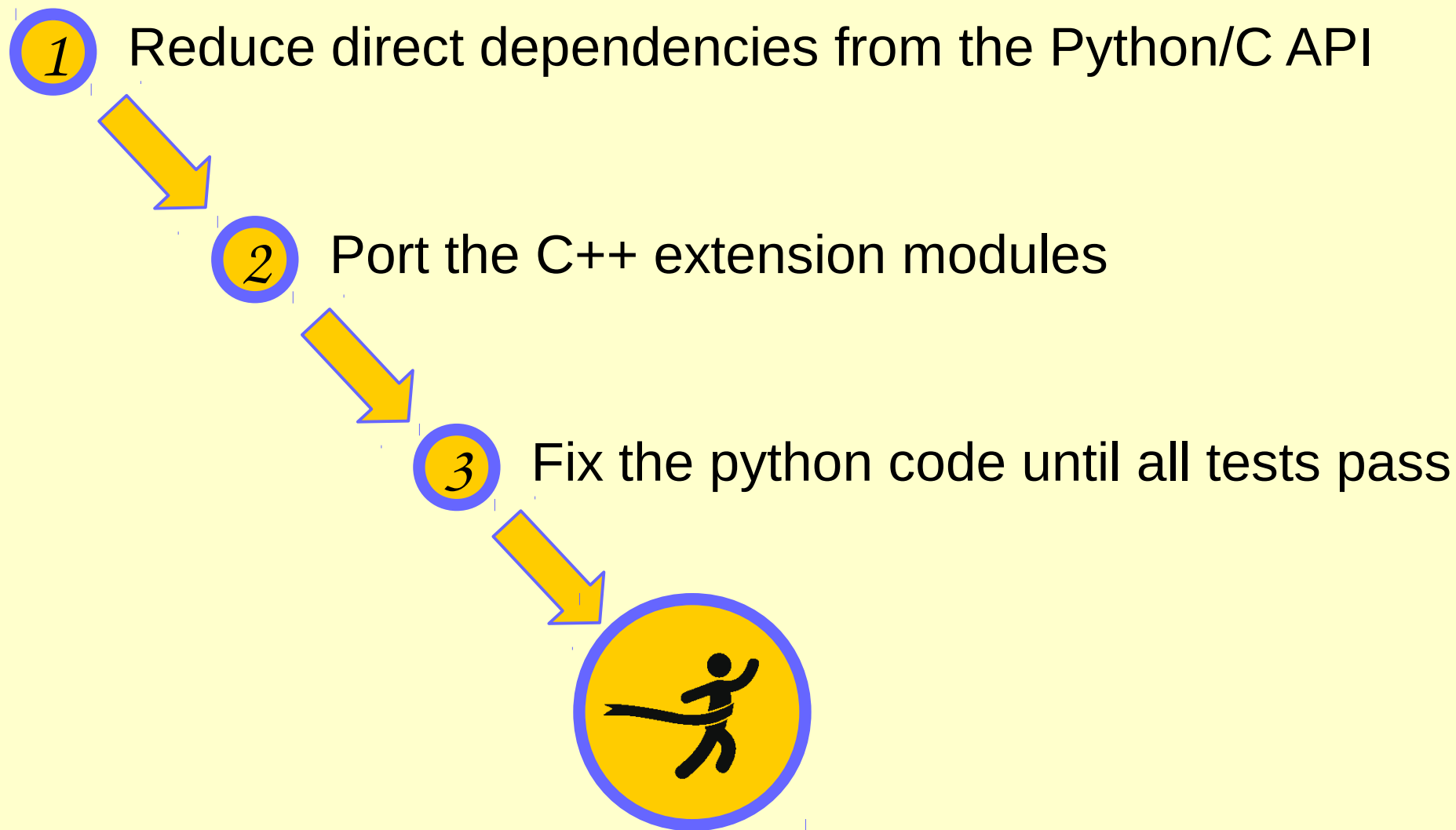


First python 3 releases haven't been very successful at supporting a 2 -> 3 migration

But writing python 2.6+/3.3+ compatible code is now definitely simpler

The possibility that porting a project is blocked by missing dependencies is now much more unlikely

Porting the RDKit



Porting the RDKit – the C++ layer

- Some Code/ML extension modules have first been ported from using the Python/C API to Boost.Python
- Main portability fixes affecting the C++ layer:
 - Numpy initialization `import_array()`
 - Interface of the implemented iterators
`next -> __next__`
 - Handling some dependencies from `PyLong_*`

Porting the RDKit – the Python layer

- 1st set of changes

Large number of simple changes introducing a py2/py3-compatible syntax and more portable idioms

- About 2/3 of the python tests passing
- Branch moved to the official RDKit repository on GitHub (PR affecting 200+ files and almost 5K loc)

Porting the RDKit – the Python layer

- Second round of portability fixes:
 - Language changes related to object comparison
 - Some subtle issues due to changes in the random module and hashing
 - A fair number of more complicated issues with pickled files, and text vs. binary data.
- Greg has done most of the work on this part
- When all tests passed the code was merged onto the master branch

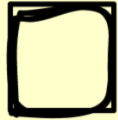
Current status

- Travis-CI is now building and testing w/ both Python 2.7 and 3.4
- The conda recipes support building Boost and the RDKit with Python3
- Python3 support required no significant changes in the RDKit python API

The RDKit is now available for Python3

(but there's still space for contributing)

TO DO LIST



Prepare your Python2 code

- Drop Python 2.5 and possibly upgrade to Python 2.7
- Start using Python3 features in Python2 :

```
from __future__ import absolute_import
```

```
from __future__ import print_function
```

```
from __future__ import division
```

```
from __future__ import unicode_literals
```

Tests coverage


Tests, tests and more tests

You'll want to exercise the code with Python3


..but you'll also need to avoid regressions with
Python2

Development environment

Quickly switch between different versions of python, rebuild and run tests.

- Conda 
- Tox + virtualenv
- (CI may be also nice to have)

A Python 2/3 portability library

- six 

single module, easy to embed into projects

<https://pythonhosted.org/six/>

- future

more complete and comprehensive package

<http://python-future.org>

Common compatibility issues



The print function

```
# python2
```

```
print >> sys.stderr, 'spam'
```

```
print 'spam',
```

```
print ', '.join([ 'spam', 'spam', 'bacon', 'spam' ])
```

```
# python3 and python2
```

```
from __future__ import print_function
```

```
print('spam', file=sys.stderr)
```

```
print('spam', end='')
```

```
print('spam', 'spam', 'bacon', 'spam', sep=', ')
```

Handling Exceptions

```
# python2
```

```
raise Exception, v
```

```
try:
```

```
    # ....
```

```
except Exception, e:
```

```
    pass
```

```
# python2.6 and 3
```

```
raise Exception(v)
```

```
try:
```

```
    # ....
```

```
except Exception as e:
```

```
    pass
```

Relative import statements

```
from __future__ import absolute_import  
# import mymodule  
from . import mymodule
```

Iterators Everywhere

python2

`range() / xrange()`

`dict.iteritems()`

`dict.iterkeys()`

`dict.itervalues()`

`zip() / map()`

python3

`range()`

`dict.items()`

`dict.keys()`

`dict.values()`

`zip() / map()`

Standard library modules

Python2	Python3	six
cPickle	pickle	cPickle
cStringIO.StringIO	io.StringIO	cStringIO
...

`urllib`, `urllib2` and `urlparse` have been reorganized into `urllib.request`, `urllib.parse` and `urllib.error`.

(limited support from `six`, but `requests` is already available for python3)

Strings, bytes and unicode

- In python2 `'str'` is a sequence of bytes (8-bit characters, either text or binary data), unicode text is `'unicode'`
- In python3 `'str'` is unicode text, `'bytes'` is a sequence of bytes (8-bit integers, always binary data)

Strings, bytes and unicode

- Avoid mixing binary and text data (especially in pickles)
- Specify when opening a file as binary (now it really matters)
- Avoid bytes/text comparison operations

```
>>> b'' == '' # result depends on version
```




- “Porting to Python 3 – An in-depth guide”
<http://python3porting.com/>
- Python-Future's compatible idioms page
http://python-future.org/compatible_idioms.html

Thank you