



Python 2 versus 3

PyHEP workshop 2018, Sofia, Bulgaria

Stefan-Gabriel Chitic, Ben Couturier *on behalf of the LHCb collaboration*
CERN

Why should you migrate to Python 3?

End of support and updates for Python 2.x

- *Python 2.6.x* ended with 2.6.9 in 2013 - SLC6
- *Python 2.7.x* will end in 2020 - Centos 7
- *No more 2.x*

Now or it will be too late: Migrate to Python 3!
341/360 top pypi packages are *Python 3* ready

Python migration in Pypi repository

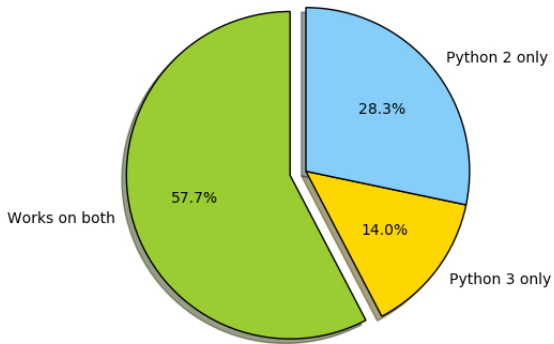


Figure: Python 2 vs Python 3 compatibility in Pypi packages¹

¹<https://goo.gl/GXtqTL>

What is new ?

Before the transition

Migration tools

Conclusion

What is new ?

What is new ?

- Cool *new* features
 - Concurrent programming (Asyncio)
 - Everything is an iterator
 - Chained exceptions, Keyword only arguments, No more comparison of everything to everything, and many others²
- *NumPy, Astropy, matplotlib, Pandas, IPython, SymPy* and many others scientific Python libraries are all compatible with Python 3 (support will drop in 2020³) and **support for some packages (e.g. CPython, osBrain, PyMeasure) is only available only for python 3.x⁴**

²<http://goo.gl/cPNjgX>.

³<https://python3statement.org>.

⁴<https://goo.gl/Br16ZH>.

print as function!

- `>>>from __future__ import print_function`

- **Not a big deal**

- **More flexible**

- **The string separator is customizable**

- `>>>print("A=", 20, sep="")`

- A=20

- **Print function can be overridden**

- `>>>import builtins`

- `>>>builtins.print = custom_logger`

Syntax changes

- Exceptions:
 - Python 2 : ...except (IoError, OSError), err
 - Python 2 & 3: ...except (IoError, OSError) as err
- Relative imports:
 - Python 2: >>>from local_package import function
 - Python 3: >>>from .local_package import function

Unicode vs Bytes

- All Strings are Unicode by default:
 - Python 2: `>>>u"Hello world"`
Python 3: `>>>"Hello world"`
- Python 3: two byte classes are introduced: bytes and bytearray
 - `>>>b"this is data"`
 - `>>>bytes([1, 2, 3, 4])`
`b'\x01\x02\x03\x04'`

Division

- `>>>5/2`
 - Python 2: 2
 - Python 3: 2.5
- Python 3 semantics in Python 2
 - `>>>from __future__ import division`
 - `-Q` flag to interpreter
- Not automatic for something other than built-in types

Python 3 and mathematics

- Matrix multiplications `>>>x@y`
- Extended iterable unpacking:
`>>>a, *b, c = range(5)`
- Integer unification:
 - `int` went away
 - `long` became `int`
 - `L` suffix does not exist anymore

Before the transition

Impact

- *Long transition time*: Keep the retro-compatibility with previous python versions: 2.6.6 (default on SLC6), 2.7.5 (default of Centos7)
- Maintain one package for all python version
- Avoid adding/removing (extra) dependencies

Needs

- Strategy on how the migration should be done
- Testing environment for all the considered python version
- Analysis of cross-versions dependencies
- Multi-python version: matrix of tests to see the failures on different versions

Continuous integration and testing

- Matrix of python versions: 2.6.6 (default for SLC6), 2.7.5 (default for Centos7), 2.7.15 (latest 2.7.x) and 3.7 (latest 3.x)
- Docker ready template usable on GitLab CI
- Automated unit and integration testing in GitLab CI, Jenkins, tox and other continuous integration systems.

Migration tools

Backported features

Many features of Python 3 are available in 2.6

- Unicode and bytes literals : `from __future__ import unicode_literals`
- Future built in functions: `from future_builtins import map, zip, hex`
- New syntax for catching and raising exceptions compatibility

Pylint

- Can warn against some things not allowed or changed in Python 3
- Use the `--py3k` to run only checks related to Python 3 compatibility

Pep8

- Yet another tool to check your Python code against some of the style conventions in PEP 8.
- Comes with an automated rules transformer called autopep

2to3

- Reads Python 2.x source code and transform it into valid Python 3.x
- Library contains a rich set of fixers that will handle almost all code
- Possible to write your own rules verifiers for 2to3
- <https://docs.python.org/2/library/2to3.html>

Modernize

- Based on 2to3 library
- Updates Python 2 code to work with Python from 2.6 to 3.x
- <https://github.com/python-modernize>

Futurize

- Like Modernize
- Backports of Python 3 features like byte type
- Part of future project
- <http://python-future.org/>

Sorry!

Some fixes are not done automatically! They need working and **thinking!**

- Need to decide between text and binary data
- In Python 3, `range`, `zip`, `map`, `dict.values`, etc return memory-efficient iterables
- If you want a list, just wrap the result with `list`
- Explicit is better than implicit

LHCb Python 3 migration

- Python 2 is highly used in LHCb
- As mentioned in *Distributing Python for the HEP environment* by B. Couturier , LHCb software stack middleware are Python 2 based (e.g arc : 15.03.14, GFAL2 : 2.15.4, FTS3 : 3.7.8, dcap : 2.47.12, xrootd : 4.8.3, etc)
- LHCb infrastructure for CI has already started the migration:
LbInstall, LbScripts
- Testing and CI for Python 3: ready in Gitlab CI

E.g. of docker and Gitlab CI integration and testing

lbmessaging

- Overview
- Repository
- Registry
- Issues 0
- JIRA
- Merge Requests 0
- CI / CD**
 - Pipelines
 - Jobs**
 - Schedules
 - Environments
 - Kubernetes
 - Charts
 - Settings

```
2018-06-07 09:01:52.425 [Info] <0.2520.0> connection <0.2520.0> (127.0.0.1:44460 -> 127.0.0.1:5672): user 'lhcbadmin' authenticated and granted access to vhost '/lhcb-test'
2018-06-07 09:01:52.456 [Info] <0.2537.0> accepting AMQP connection <0.2537.0> (127.0.0.1:44464 -> 127.0.0.1:5672)
2018-06-07 09:01:52.461 [Info] <0.2537.0> connection <0.2537.0> (127.0.0.1:44464 -> 127.0.0.1:5672): user 'lhcbadmin' authenticated and granted access to vhost '/lhcb-test'
2018-06-07 09:01:53.452 [Warning] <0.2537.0> closing AMQP connection <0.2537.0> (127.0.0.1:44464 -> 127.0.0.1:5672, vhost: '/lhcb-test', user: 'lhcbadmin'): client unexpectedly closed TCP connection
2018-06-07 09:01:53.470 [Info] <0.2520.0> closing AMQP connection <0.2520.0> (127.0.0.1:44460 -> 127.0.0.1:5672, vhost: '/lhcb-test', user: 'lhcbadmin')
2018-06-07 09:01:53.479 [Info] <0.2558.0> accepting AMQP connection <0.2558.0> (127.0.0.1:44468 -> 127.0.0.1:5672)
2018-06-07 09:01:53.483 [Info] <0.2558.0> connection <0.2558.0> (127.0.0.1:44468 -> 127.0.0.1:5672): user 'lhcbadmin' authenticated and granted access to vhost '/lhcb-test'
2018-06-07 09:01:53.487 [Info] <0.2566.0> accepting AMQP connection <0.2566.0> (127.0.0.1:44472 -> 127.0.0.1:5672)
2018-06-07 09:01:53.494 [Info] <0.2566.0> connection <0.2566.0> (127.0.0.1:44472 -> 127.0.0.1:5672): user 'lhcbadmin' authenticated and granted access to vhost '/lhcb-test'
2018-06-07 09:01:53.677 [Warning] <0.2558.0> closing AMQP connection <0.2558.0> (127.0.0.1:44468 -> 127.0.0.1:5672, vhost: '/lhcb-test', user: 'lhcbadmin'): client unexpectedly closed TCP connection
2018-06-07 09:01:57.497 [Warning] <0.2566.0> closing AMQP connection <0.2566.0> (127.0.0.1:44472 -> 127.0.0.1:5672): missed heartbeats from client, timeout: 1s
.....
Name                               Stmts  Miss  Cover   Missing
-----
lbmessaging/__init__.py              10      1    94%    41
lbmessaging/exchanges/CommandExchange.py  30      0   100%
lbmessaging/exchanges/Common.py      269     31    88%  49-54, 58, 89, 422-423, 42
6, 457-461, 567-571, 595-596, 599-600, 653, 705, 711, 723-727
lbmessaging/exchanges/ContinuousIntegrationExchange.py  71      7    90%  58, 61-66, 77
lbmessaging/exchanges/CvmsfConDRExchange.py  10      0   100%
lbmessaging/exchanges/CvmsfDevExchange.py  10      0   100%
lbmessaging/exchanges/CvmsfProdExchange.py  10      0   100%
```

centos7 Retry

Duration: 2 minutes 43 seconds
Runner: #881
Tags: [ovmf](#)

Job artifacts
The artifacts were removed a day ago

Commit [2abb34c3](#)

Update [.gitlab-ci.yml](#)

- centos7
- centos7
- python3.6
- python3.5
- python2.7

Figure: Lbmessaging Gitlab CI

Lessons learned

- **DON'T** use 2to3, autopep in this order because first step will render the code almost python 3 ready and the second step will impact all the files, making debugging impossible
- Lint as much as possible and respect the coding rules and guidelines

Conclusion

- It is the time to migrate to Python 3.
- Extra code to keep the retro compatibility should be easy to remove when your code will drop Python 2 support
- New code should be written in Python 3 directly (Remember: 341/360 are python 3 ready)
- Infrastructure is available for new projects

Remember!

- Python 3 will become the default version on future operating systems
 - `#!/usr/bin/env python2`
- Code today in Python 3 and back port it to Python 2
 - `# -*- coding: utf-8 -*-`
`from __future__ import (division, absolute_import, print_function)`



home.cern

More stuff

- Conda environment manager:

`https://conda.io/miniconda.html`

- LHCb docker images for Python: `dockerpullgitlab-registry.cern.ch/lhcb-docker/python-deployment`

- Python 3 features: `https:`

`//www.asmeurer.com/python3-presentation/slides.html`

More links

- <http://py3readiness.org/>
- <https://python3wos.appspot.com/>
- <https://docs.python.org/3/howto/pyporting.html>
- <https://github.com/brettcannon/caniusepython3>