Building Bridges, not walls

Ending Python2 compatibility in a user friendly manner

M Bussonnier & M Pacer

Slides available at http://bit.ly/pycon2017-build-bridges

About us

We have been working on the IPython and Jupyter projects for 5 and ~1 year.

github:@Carreau/twitter:@Mbussonn

github:@mpacer/twitter:@mdpacer

Alfred P. Sloan
FOUNDATION

GORDON AND BETTY
MOORE
FOUNDATION

jupytercon.com – August 22-25; NYC

What this talk is not about

- Is Python 2 or 3 the right choice?
- Should I migrate to Python3-only?

What this talk is about

We migrated IPython to Python 3 only.

We care about all of our users, Python 2 and 3 alike.

We want to make the transition the least frustrating for users and dev.

We'll be describing how we did this.

Python 2 vs 3 is an example

The lessons we've learned are not specific a python2 to python3 transition.

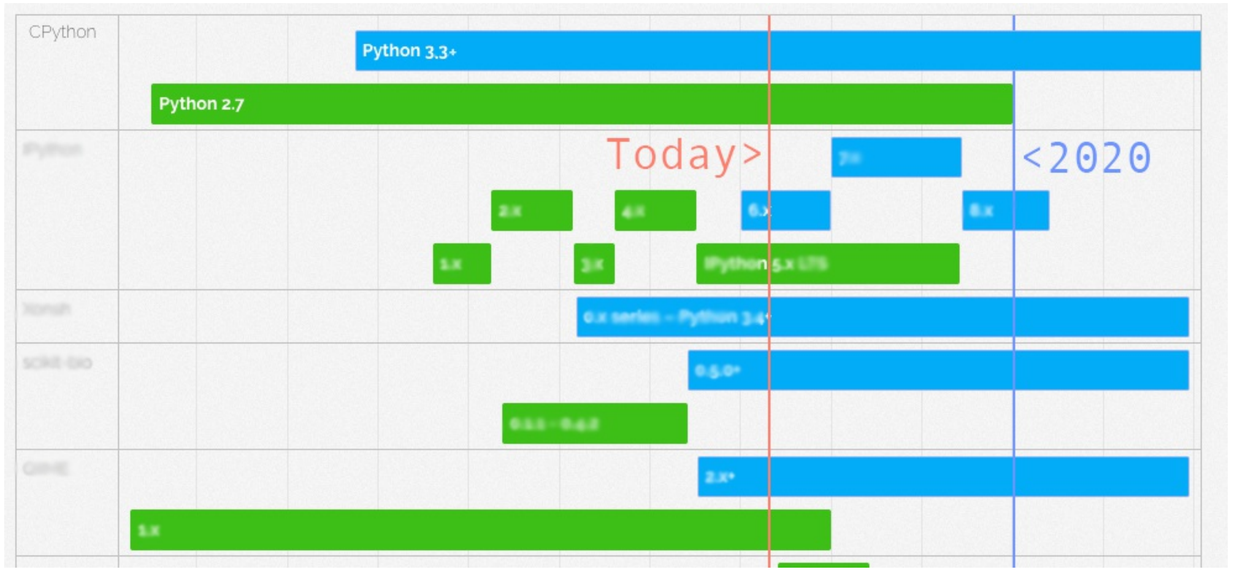
Our talk applies to stopping support for any version (e.g., 2.6 or 3.3).

Python 3 Statement

www.python3statement.org

List of who is stopping Python 2 support when. Resources on how to stop support with minimal frustration for users


IP[y]:
IPython
jupyter
Jupyter notebook
pandas
Matplotlib
SymPy
Astropy
software carpentry
Software Carpentry
SunPy
xonsh
SCIKIT-BIO
scikit-bio
PyStan
Axelrod
osBrain
osBrain
PyMeasure
rpy2
rpy2



Timeline of Python 2 End of Life for various projects.

Scratch your own itch

We wanted to release IPython 6, the code base should be Python 3 only.

We care about Python 2 users, so if a Python 2 user runs

```
$ pip install ipython --upgrade
```

it should install the latest version of IPython 5, not IPython 6!

Core of the problem

```
$ pip install ipython --upgrade
Installing ipython... doing magic... success

$ python
>>> import IPython
SyntaxWarningErrorError("I Just don't like you.")
```

Solutions (Not really)

Let's go back to 2016.

Just use `$ pip install "ipython<6"` on Python 2

- Users do not always read documentation before installing.

- Scripts do not read documentation before installing.
- Users/scripts do not read error messages.
- dependencies – all packages need update to have conditional dependencies.

Rename ?

That's going to be confusing and void most of the documentation on the Internet.

Import names different from package name is also a bit tricky to explain sometime.

Wheel only ?

Ok-ish for Pure-Python packages.

Many downstream distribution requires sdist.

Use a metapackage

Use a package with virtually no-code that have conditional dependencies, and move the "real" code to a sub package.

- You kinda need to re-release old code (can't requires old-yourself)
- `pip upgrade metapackage` will not pull `core` unless pinned deps

use a pip ~~bug~~ "Hidden Feature":

```
# somewhere in pip
_py_version_re = re.compile(r'-py([123]\.?[0-9]?)$')

# somewhere else
if is_tar_gz(file):
    match = self._py_version_re.search(version)
    if match:
        version = version[:match.start()]
        py_version = match.group(1)
        if py_version != sys.version[:3]:
            self._log_skipped_link(
                link, 'Python version is incorrect')
            return
```

use a pip ~~bug~~ "Hidden Feature":

You can publish `ipython-py3.3.tar.gz` and `ipython-py3.4.tar.gz` and `ipython-py3.5.tar.gz` and `ipython-py3.6.tar.gz` and `ipython-py3.7.tar.gz` to be future proof.

But it does not work beyond Python 3.9...

As Raymond Hettinger would say if he is in the room

There must be a better way !

The new way: Python-Requires

Since December with pip 9.0.1, and setuptools 24.3:

```
# setup.py

setup(...,
    python_requires='>=3.4'
)
```

Use `pip install` and it will adhere to `python_requires`.

N.B.: Do not invoke `setup.py` directly!

In greater detail

`python_requires` metadata comes from pep 345, 2005.

But for 11 years nothing implemented or understood it.

setuptools >= 24.3

The `python_requires` keyword in known only by setuptools versions > 24.3.

- Required to **build** the sdist/wheel and publish the package
- Required to **install** from sdist.

pip >= 9.0.1

Versions of pip < 9 ignore `data-requires-python` attributes.

This will result in installing incompatible versions.

Under the Hood

The old PEP

PEP 345

```
Requires-Python
===============

This field specifies the Python version(s) that the
distribution is guaranteed to be compatible with.

Version numbers must be in the format specified in
Version Specifiers.

Examples:

    Requires-Python: 2.5
    Requires-Python: >2.1
```

Great! Now what? How do we use it?

Patch all the things!

Build: Setuptools

As of setuptools 24.2:

```
# setup.py

setup(...,
    python_requires='>=3.4'
)
```

Upload: PyPI & Warehouse

Updates the `release_files` table backing PyPI-legacy and

Updates the `release_files` table backing PyPI-legacy and Warehouse

Surface: PyPI

PEP 503 defines `/simple/` repositories formats.

Require Python inside the `data-requires-python` attribute

view-source:https://pypi.python.org/simple/pip/

```
<!DOCTYPE html><html><head><title>Links for pip</title>
<a href="../pip-8.0.0-py2.py3-none-any.whl" >pip-8.0.0-p
```

```
<a href="…/pip-6.0.4.tar.gz" >pip-6.0.4.tar.gz</a><br/>
<a href="…/pip-0.3.1.tar.gz" >pip-0.3.1.tar.gz</a><br/>
<a href="…/pip-1.0.1.tar.gz" >pip-1.0.1.tar.gz</a><br/>
<a data-requires-python="&gt;=2.6,!=3.0.*" href="…/pip-
```

Install : Pip

Pip 9+ checks `data-requires-python` before downloading

Tying it together

Setuptools, PyPI, Warehouse and pip are updated and deployed!

Alternative packaging tool/service maintainers

As of yesterday, PEP 518 implementation was merged

`pyproject.toml` **is now valid**!

many tools/services are already compatible, e.g., `flit` and `twine`

Call to all package maintainers

- Use tools and services that respect Require-Python
  - pip 9+
  - setuptools >24.2

But, most of all:

Tell users to update pip!

Defensive packaging

Even if you adhere to that, problems will arise.

Some helpful principles to keep your users as happy as possible

1. Update your documentation, CIs, and scripts to use `pip`.

2. Keep `setup.py` and `__init__.py` python 2 compatible,
   but catch errors early.

3. For clear error messages in complicated situations,
   use multiple lines.

Tell everyone to `pip install`

Update your documentation and scripts to use `pip install [-e] .`.

Reiteration: Do not use `python setup.py <…>`;
it ignores `requires_python` and `pyproject.toml`

Keep `setup.py` python 2 compatible.

If `setup.py` runs most probable reason:

pip < 9.

Don't just say your package is incompatible with `python2`.

Instead: Ask users to update pip.

E.g.,: in `setup.py`:

```
if sys.version_info < (3, 3):
    error = """
IPython 6.0+ does not support Python 2.6, 2.7, 3.0,
3.1, or 3.2. Beginning with IPython 6.0, Python 3.3
and above is required.

This may be due to an out of date pip.

Make sure you have pip >= 9.0.1.
"""
    sys.exit(error)
```

Keep `__init__.py` python 2 compatible

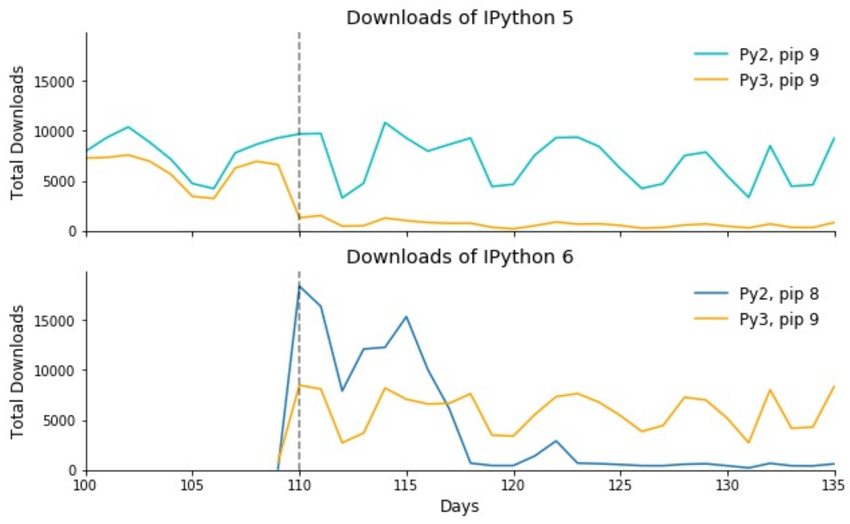
Users will still find ways to avoid `pip` and `setup.py`.
E.g.:

```
$ pip install -e .
$ ...
$ git pull  # update without install
```

E.g., in `__init__.py`:

```
import sys
if sys.version_info < (3,3):
    raise ImportError(
"""
IPython 6.0+ does not support Python 2.6, 2.7, 3.0,
3.1, or 3.2. Beginning with IPython 6.0, Python 3.3
and above is required.
""")
```

Results



Bug reports / complaints

Two.

- During RC: `python setup.py install` got 6.0 on Python 2

Remember: don't use `python setup.py`!

My Bad. I thought the pip upgrade error
message was irrelevant —
when I upgrade pip it works.

Remember: upgrade `pip`!

Conclusions

On IPython

- IPython 6+ is Python3 only
- We're still updating the IPython 5.x – Python 2 LTS branch
- Transition has gone relatively well for IPython!
  - It will only get easier!

On converting packages to Python3 only

- use packaging tools that respect `Requires_Python`
- encourage everyone, everywhere to use pip 9+
- follow defensive packaging practices
- Read and contribute to python3statement practicalities section

Thanks!

Python 3 \u2661 Python 2

(that's a ♡)

Slides at: http://bit.ly/pycon2017-build-bridges

Slides at: http://bit.ly/pycon2017-build-bridges

@mpacer & @Mbussonn