Building Bridges, not walls

Ending Python2 compatibility in a user friendly manner

M Bussonnier & M Pacer

Slides available at https://short.url

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

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

IPython

Jupyter notebook

pandas

Matplotlib

SymPy

Astropy

Software Carpentry

SunPy

xonsh

scikit-bio

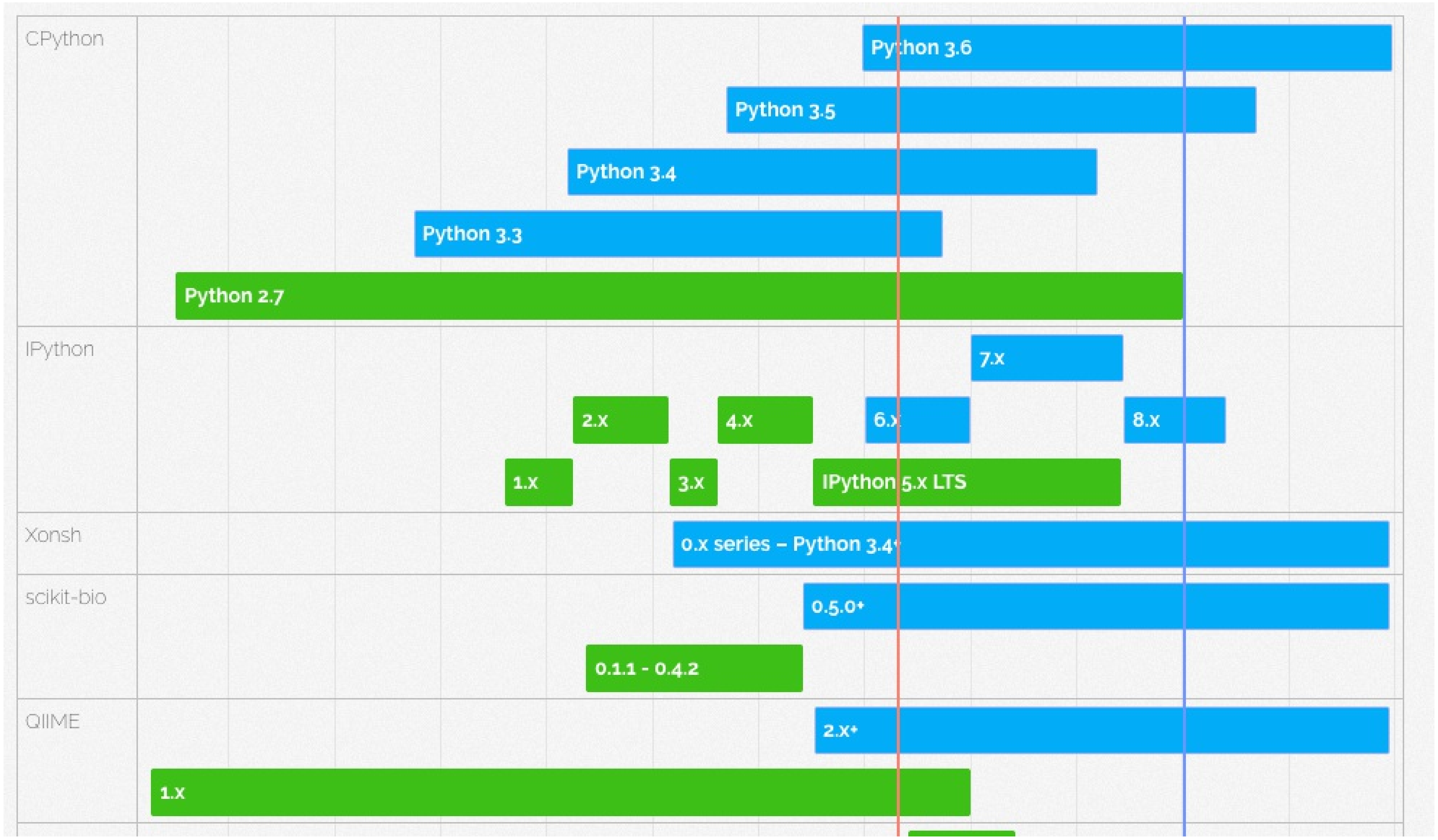
PyStan

Axelrod

osBrain

PyMeasure

rpy2


CPython
Python 3.6
Python 3.5
Python 3.4
Python 3.3
Python 2.7
IPython
7.x
2.x
4.x
6.x
8.x
1.x
3.x
IPython 5.x LTS
Xonsh
0.x series – Python 3.4+
scikit-bio
0.5.0+
0.1.1 - 0.4.2
QIIME
2.x+
1.x

Scratch your own itch

We released IPython 6, the code base is Python 3 only.

IPython 5 will still be stable (LTS). So if a Python 2 user runs

```
$ pip install ipython -U
```

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.")
```

[Not really] Solutions

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

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

This will result in installing incompatible versions.

Defensive packaging

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

2. Keep `setup.py` and `__init__.py` python 2 compatible, but have them err early.

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

Direct users to `pip install`

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

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

Keep `setup.py` python 2 compatible.

If installation fails before `setup()`, the most probable reason:

pip < 9.

Catch this, and don't let installation finish!

Instead: explicitly ask users to update pip.

E.g.,: in `setup.py`, before `setup()`:

```
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` before module imports:

```
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.

See IPython `README.rst` file for more information:

    https://github.com/ipython/ipython/blob/master/READ

""")
```

Results

IPython 6.0, #downloads:

First Week:

- Pip 9 - Python 3 : 45 586
- Pip 8 - Python 2 : 92 386

  > 2×, not good

Second Week:

- Pip 9 - Python 3 : 48 389
- Pip 8 - Python 2 : 13 293

  > 0.25 ×, still not great, but better!

Bug reports / complaints

Two.

- During RC: `python setup.py install` got 6.0 on Python 2 – now documented.
- "My Bad I did not read the error message"

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
    Requires-Python: >=2.3.4
    Requires-Python: >=2.5,<2.7
```

Great! How do we use it?

Setuptools

As of setuptools 24.2, your `setup()` call follows the `python_requires` keyword when building a package from source.

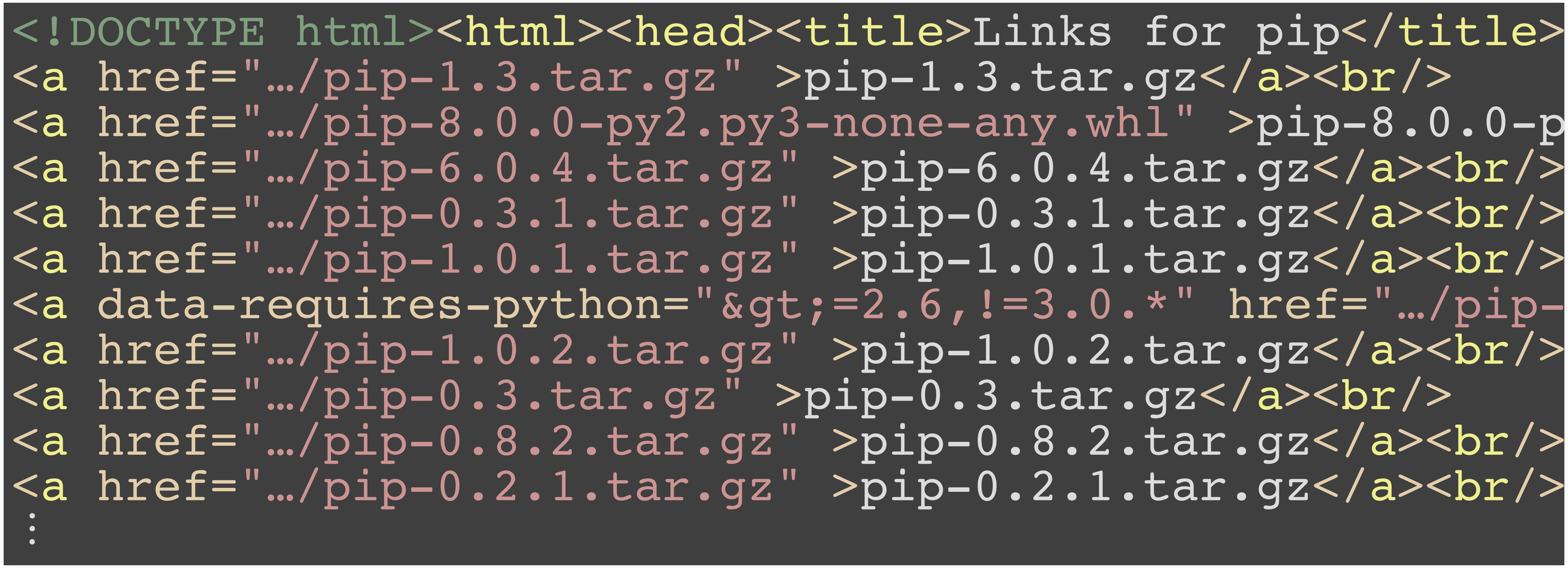
Kudos to @xavfernandez for making that possible.

Pypi

Pip should get Require-Python info **before** downloading the sdist.

Pip does that by checking the `/simple/` repository url:

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



```
<!DOCTYPE html><html><head><title>Links for pip</title>
<a href="…/pip-1.3.tar.gz" >pip-1.3.tar.gz</a><br/>
<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-
<a href="…/pip-1.0.2.tar.gz" >pip-1.0.2.tar.gz</a><br/>
<a href="…/pip-0.3.tar.gz" >pip-0.3.tar.gz</a><br/>
<a href="…/pip-0.8.2.tar.gz" >pip-0.8.2.tar.gz</a><br/>
<a href="…/pip-0.2.1.tar.gz" >pip-0.2.1.tar.gz</a><br/>
 ⋮
```

This lists files and now have `data-requires-python` with version specifications for each file.

This was done by amending PEP 503.

N.B.: If you are running (or maintain) a PyPI proxy please make sure it surfaces new `data-requires-python`.

Pip

Pip 9+ checks `data-requires-python`.

https://github.com/pypa/pip/pull/3877

In the same place that pip process the wheel filenames (to get `-py2` , `-py3` suffixes) and filter "compatible" files.

That's the main reason you want final users to upgrade to pip 9+ if you are not on pip 9+, pip will consider incompatible packages, download them and ... fail at some point.

Patching PyPI & Warehouse: PyPI-legacy

You likely know PyPI-legacy, that's usually where most people download their packages from when then `pip install`.

But PyPI is old, its testing is sparse, and its documentation is not always accurate. It's not easy to run PyPI locally.

Patching PyPI & Warehouse: Warehouse

The PyPA stated developing Warehouse (the new, improved PyPI) with 100% test coverage and solid documentation.

It even has a one liner to run it locally using Docker!

Patching PyPI & Warehouse: Postgres

PyPI and warehouse are connected to the same Postgres database.
So any updates need to be coördinated between them.

Tying it together

It seems like it should be straightforward...

When you need the `/simple/<package>` webpage, the sql query should simply be:

```
SELECT * from release_files where package_name=package
```

Parse that, build a list of `hrefs` and `data-requires-python` values, and you're done. Right?

Patching PyPI & Warehouse: dancing between tables

PEP 345 specifies that `requires-python` is an attribute on releases, not release files.

On the one hand, that makes sense:
we distribute files, which make up releases.

On the other hand, the simple implementation won't work:

- release files are specified in the `release_files` table
- releases are specified in the `releases` table

Patching PyPI & Warehouse: dancing between tables

In theory, we could use a `JOIN` on the two tables.

Except, with the number of available packages, a `JOIN` is too slow.

At the same time, we cannot safely refactor the database because PyPI-legacy is not well tested.

Solution: Triggers

The `JOIN` was doing too much work;
it'd be better if we could update only the necessary rows.

Triggers solve exactly that problem.

We use a trigger to update the `release_files` table when it or `releases` are updated (or a row is inserted in either table).

Detail: `UPSERT` is a combination of update and insert, greatly simplifying the logic.

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 switching your package to Python3 only

- upgrade setuptools
- use pip 9+, encourage your users to do the same
- fix your documentation (use pip, not `setup.py`!)
- catch early in py2 compatible `__init__.py` and `setup.py`
- Read and contribute to python3statement practicalities section
    - questions, gotchas, &c.

On contributing to packaging infrastructure

- We've improved the documentation of both warehouse and PyPI, to make new contributions easier.
- You should contribute — there's tonnes of low hanging fruit!
- Add tests, clean up the codebase, or bring features from PyPI to Warehouse.