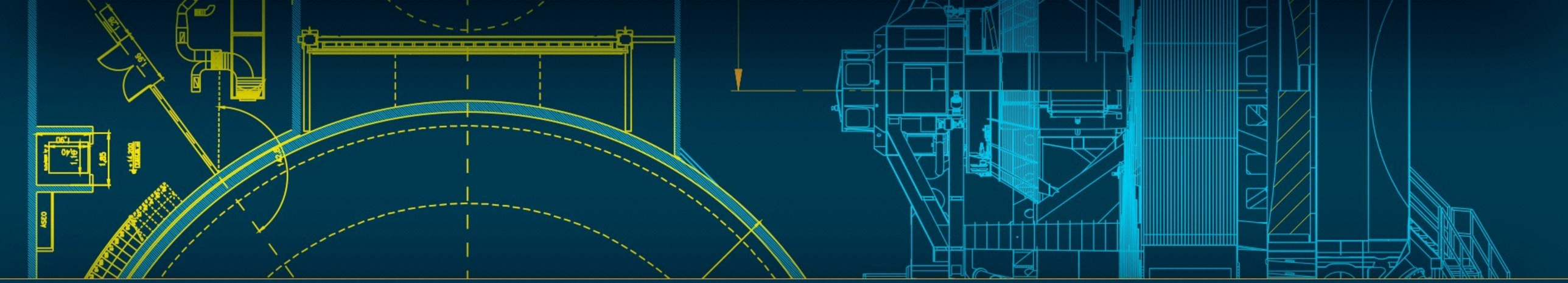


PYTHON 3 AND LSST

LSST2016 PROJECT AND COMMUNITY WORKSHOP

AUGUST 15-19, 2016 | TUCSON, ARIZONA

Background

- RFC-60: Support Python 3 and 2.7 simultaneously (Adopted Aug 2015)
- Legacy Python (2.7) support ends in 2020, before LSST is commissioned.
- Migrate the stack to support both Python 3.5 and 2.7 using the "`future`" package.
- Astropy planning to drop compatibility with Python 2 in December 2017. iPython and matplotlib are also refining timelines for dropping Py2.
- No fixed date for LSST dropping Legacy Python support.

Current Status

- `lsstsw` works with Python3. Use the "`deploy -3`" option.
- Do not attempt to share a build tree built with different Python versions.
- Jenkins CI now has an option to submit Python3 jobs.
- SCons does not work on Python 3. Builds use Python 2. SConstruct files can not assume the SCons Python is the relevant Python. Use `subprocess` to query the `python` on the PATH.
- Everything needed by `afw` works on Python 3: base, pex_exceptions, utils, pex_policy, daf_base, pex_logging, pex_config, daf_persistence. log and ctrl_events also ported.

Migration Plan

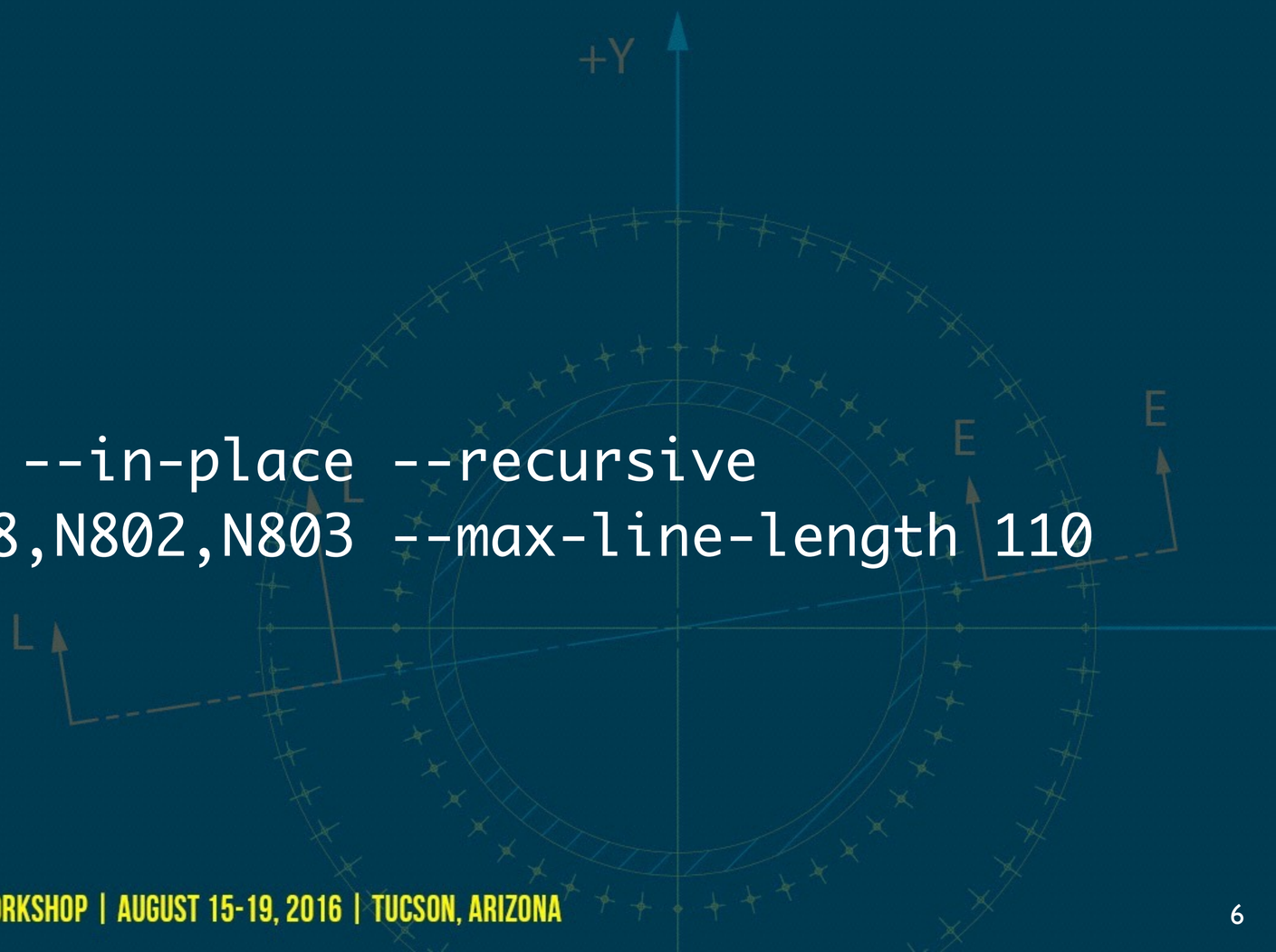
- Have to do the final migration bottom up in order to test that things really work on Python 3.
- Can do some preliminary work even on the higher level parts of the stack and ensure that it still works on Python 2 as other packages become available.
- Opportunity to clean up the code base as part of the migration. Many files will be touched.

Suggested Process

- (commit after each stage)
- run autopep8 to run a simple clean up before starting. This removes trailing whitespace and sorts out common whitespace issues. (instructions on next slide)
- Run "`futurize -1 -w .`" to modernize the code to Python 2.7 standard (print function, "`as e`" exceptions, absolute imports). Keep an eye out for "`types.StringType`" as that will be changed to `bytes`.
- Update deprecated `assert_` and `assertEquals` in tests and replace with modern equivalent. Replace `assertTrue` calls that could be replaced with more descriptive modern versions.

Running autopep8

```
% pip install autopep8
% autopep8 {{package_dir}} --in-place --recursive
--ignore E26,E133,E226,E228,N802,N803 --max-line-length 110
```



Support Python 3

- Aim of futurize is to allow you to write code in the style of Python 3 while supporting Python 2. `zip`, `range`, `items`, `keys`, `values` all return iterators rather than lists in Python 3.
- Run the stage 2 futurize:
  - "`futurize -2 –x division_safe -w .`"
- Check what it does. You do not want to accept everything it does. Use "`git add -p`" and look at each change in turn.

Common issues with futurize

- `futurize` will play safe and wrap items in `list()` if they used to return lists but now return iterators or views. Sometimes you do not want the list sometimes you do. In loops you are mostly ok, unless the list changes. When passing the result to another function the safe thing is to wrap it in a list.
  - For example, instead of `for key in list(data.keys()):` Use `for key in data:`
- `futurize` will preserve old style division. If you want modern division declare that in the file and remove `old_div`. In many cases you do not want the division changes and want to use `__future__` division. You can disable the division changes by using "`--nofix=division_safe`" in `futurize`.
  - "`__future__ division`" should be enabled and care must be taken to use `//` when appropriate.

- "L" constants (e.g. 5L) can no longer be used. All ints are long in Python 3. If you really want an "L" on Python 2 use `long()` and "`from past.builtins import long`".
- `basestring` has gone. Checking "`isinstance(x, basestring)`" requires "`from past.builtins import basestring`" as a temporary fix.
- `range` and `zip` return iterators and not lists. Can not use `xrange`, `itertools.izip`. Futurize lets you use "`from builtins import range`" and similarly for `zip`.
- `raw_input` becomes `input` (and "`from builtins import input`")
- Dict `itervalues`, `iteritems` are gone. Use `values` and `items` for `dict` and enable efficient python2 with "`from builtins import dict`".

- `str.translate` behaves differently and `string.maketrans` has gone.
- Python 3 cares about strings vs bytes. All external data will arrive in bytes and must be decoded (important for `subprocess` calls). Open files in binary mode, "rb", for bytes, "r" for strings. Pickle files must be binary mode. `tempfile` makes binary temp files by default.
- `sorted` no longer has a `cmp` argument. Use `key`
- Python 3 renames `next()` to `__next__()` (consistent use of double underscores for special methods). This works in python 2.6 so updating is easy.
- Checking for `__iter__` attribute is no longer sufficient for disambiguating string from sequence.
- Multiple inheritance of two classes that each have their own metaclass needs extra work.
- `ConfigParser` becomes `configparser` (get back port from PyPI).

Python/C++ interfaces

- SWIG on Python 2 will not, by default, assume that a `unicode` string can be mapped to `std::string`.
- When `future` is involved some strings become `unicode` on Python 2.
- Must enable `unicode` to `std::string` mapping in `.i` file.
- `p_lsstSwig.i` from `utils` does this for you.
- Sometimes your C++ routine actually wants bytes and not strings but uses `std::string`. Then in your `.i` file you have to define `SWIG_PYTHON_STRICT_BYTE_CHAR`. This will then expect `bytes` but may change the entire SWIG interface to expect bytes.

Rinse and repeat

- Have two separate stacks: one for 2 and one for 3. Do not mix.
- You will inevitably break Python 2 when you get Python 3 working.
- Switch back to Python 2 and fix the issues.
- Then iterate until it works on both.
- Don't forget to add `python_future` to EUPS table file.
- When you are ready, build everything on python2 with the ticket branch (include `ci_hsc`, `lsst_ci`). If that all works send for review and merge.
- When everything works on Python 3 new code will have to pass Jenkins on both Py2 and Py3.

Suggested packages to start with

- skymap, shapelet, display_ds9
- geom >> skypix >> daf_butlerUtils
- obs_test >> pipe_base >> coadd_utils
- db >> cat >> ctrl_provenance >> ctrl_orca
- dax_webservcommon >> dax_dbserv & dax_metaserv
- 3rd party to check: mysqlpython (no), healpy, lmfit, mpi4py, esutil, scisql, flask
- psfex and galsim might be tricky

Porting Guides

- https://docs.python.org/2/howto/pyporting.html
- http://python-future.org
- http://python3porting.com