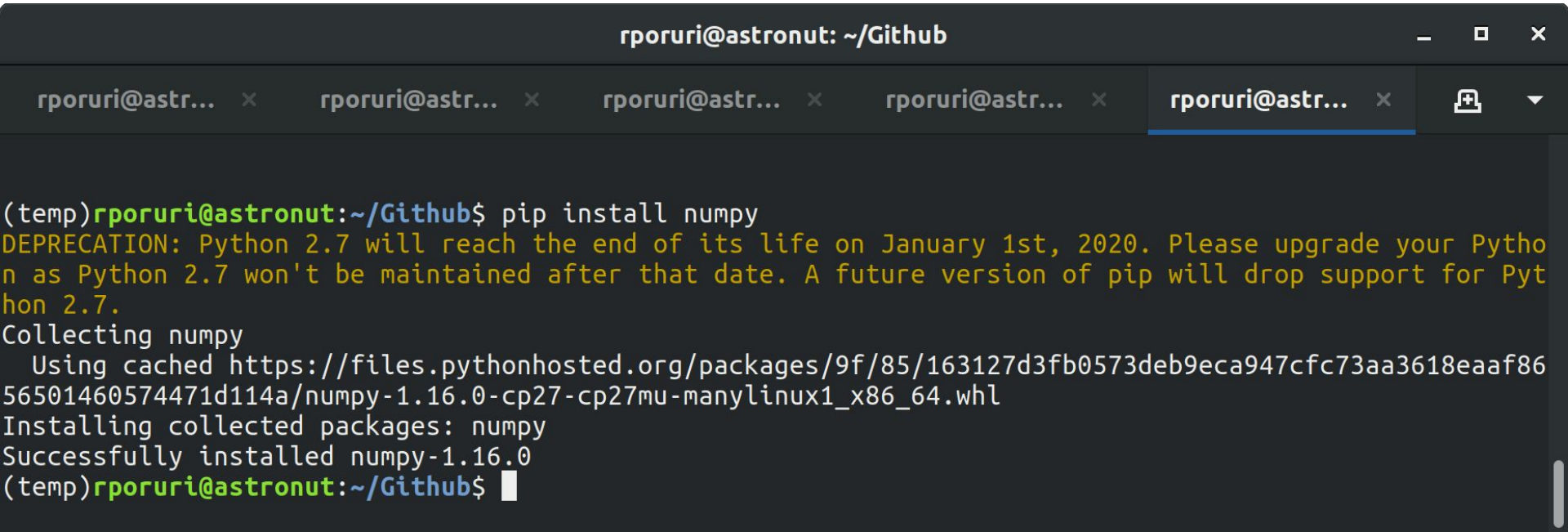
Porting Python applications to support Python 2 and 3

Poruri Sai Rahul,
Scientific Software Developer,
Enthought Inc.

Why?
What?
How?

Why?

Python 2 EoL 2020



Python 2 EoL 2020

- Python 2.7.X will be the last version of Python 2 which contained new features
- Python 2 will not be officially supported after 2020
  - This means no bug fix releases and no critical security fixes
- Support for Python 2 is being dropped by popular open source packages
  - Pandas and Numpy already dropped support for Python 2 on Dec 21 2018
  - See https://github.com/numpy/numpy/blob/master/doc/neps/nep-0014-dropping-python2.7-proposal.rst and http://pandas.pydata.org/pandas-docs/stable/install.html#plan-for-dropping-python-2-7 for detailed info

What?

Python 2 VS Python 3

- Syntax changes
  - print statement on Python 2 vs print function on Python 3
- Behavioral changes in builtins
- Changes in the Python standard library
- APIs not available on Python 3

Python 2 VS Python 3

- Syntax changes
- Behavioral changes in builtins
  - long integer type doesn't exist in Python 3
  - The return value of dictionary methods (e.g. dict.keys()) on Python 2 and 3
  - ASCII-encoded strings vs Unicode strings vs Byte strings
  - 2 / 3 on Python 2 vs Python 3
- Changes in the Python standard library
- APIs not available on Python 3

Python 2 VS Python 3

- Syntax changes
- Behavioral changes in builtins
- Changes in the Python standard library
  - cStringIO/StringIO on Python 2 vs io on Python 3
- APIs not available on Python 3

Python 2 VS Python 3

- Syntax changes
- Behavioral changes in builtins
- Changes in the Python standard library
- APIs not available on Python 3
    - Methods on unittest.TestCase : assertItemsEqual on Python 2 vs assertCountEqual on Python 3

How?

Tools

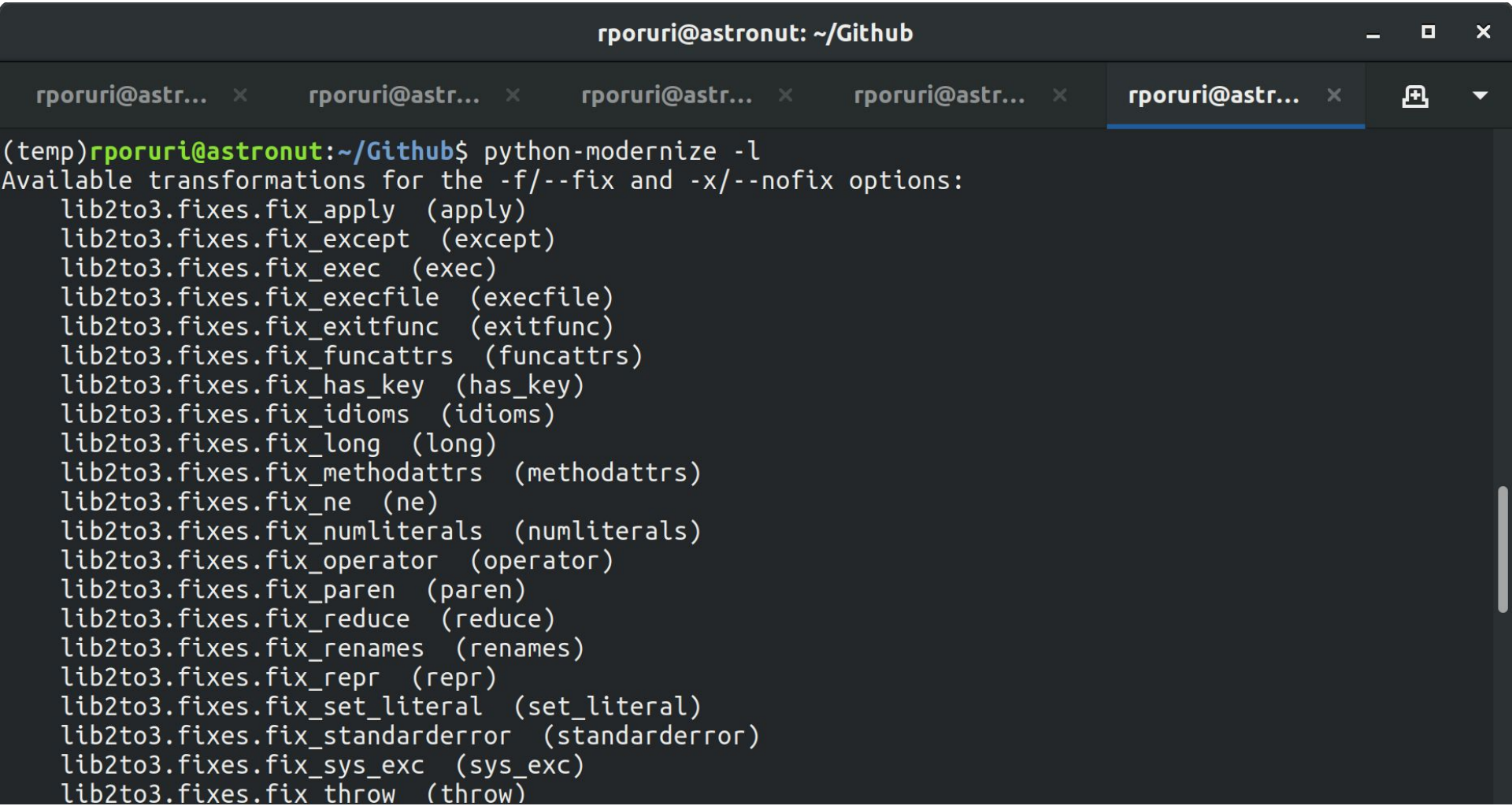
- flake8
  - Used to check for syntax errors on Python 2/3
- python-modernize
  - Automate porting of Python 2 code
- future
  - Live in the future i.e. Python 3
- six
  - Package to access Python 2 / 3 features from a unified API

flake8

python-modernize

Automatically make changes to the codebase

- Update print statements to print function calls
- Update handling exceptions in try/except code blocks
- Update use of dictionary methods


rporuri@astronut: ~/Github
rporuri@astr...
rporuri@astr...
rporuri@astr...
rporuri@astr...
rporuri@astr...
(temp)rporuri@astronut:~/Github$ python-modernize -l
Available transformations for the -f/--fix and -x/--nofix options:
lib2to3.fixes.fix_apply (apply)
lib2to3.fixes.fix_except (except)
lib2to3.fixes.fix_exec (exec)
lib2to3.fixes.fix_execfile (execfile)
lib2to3.fixes.fix_exitfunc (exitfunc)
lib2to3.fixes.fix_funcattrs (funcattrs)
lib2to3.fixes.fix_has_key (has_key)
lib2to3.fixes.fix_idioms (idioms)
lib2to3.fixes.fix_long (long)
lib2to3.fixes.fix_methodattrs (methodattrs)
lib2to3.fixes.fix_ne (ne)
lib2to3.fixes.fix_numliterals (numliterals)
lib2to3.fixes.fix_operator (operator)
lib2to3.fixes.fix_paren (paren)
lib2to3.fixes.fix_reduce (reduce)
lib2to3.fixes.fix_renames (renames)
lib2to3.fixes.fix_repr (repr)
lib2to3.fixes.fix_set_literal (set_literal)
lib2to3.fixes.fix_standarderror (standarderror)
lib2to3.fixes.fix_sys_exc (sys_exc)
lib2to3.fixes.fix_throw (throw)

__future__

- absolute_import
- division
- print_function
- unicode_literals

six

Utility to access Python 2 / 3 APIs in a unified manner

-

six

Name	Python 2 name	Python 3 name
`builtins`	`__builtin__`	`builtins`
`configparser`	`ConfigParser`	`configparser`
`copyreg`	`copy_reg`	`copyreg`
`cPickle`	`cPickle`	`pickle`
`cStringIO`	`cStringIO.StringIO()`	`io.StringIO`

Strategies

- Do not use 2to3
- Do not use python-future
- Make incremental changes across multiple PRs
  - Start with syntax-related changes
  - Move on to fixing/updating standard library imports and APIs not available on Python 3
  - Finally, fix/update I/O related code
- Strive for a Python 3-first approach
- Never break master (branch on git) on Python 2