Python 2 vs. Python 3

Changes, support and porting

Gerald Senarclens de Grancy, gerald@senarclens.eu

Linuxtage, April 2013, Graz

Outline

Which version Should I use?

Short version:

Python 2.x is the status quo.

Python 3.x is the present and future of the language.

- No new major releases for Python 2.x
- Many of the less intuitive constructs have been improved
- Less corner cases
- Recent standard library improvements only available in Python 3
- New features
    - *Clean Unicode/bytes separation*
    - *Exception chaining*
    - *Function annotations*
    - *Syntax for keyword-only arguments*
    - *Extended tuple unpacking*
    - *Non-local variable declarations*
- Main issues
    - *Many libraries are not (yet) Python 3 compatible*
    - *Ported libraries might have less Python 3 compatible extensions*
    - *You may not control your target environment*
    - *Support from alternative implementations is rare*

Libraries

Not available

- gevent
-
- ...

Currently ported

Python 3 porting status

- twisted
- PIL
- ...

Available libraries

Current Linux distributions offer many Python 3 libraries

600+ libraries/modules already ported to Python 3 on PyPI

- cython
- GUI: Tkinter, PyQt4, PySide, PyGObject
- matplotlib >= 1.2
- nose
- numpy
- pip
- Web: django >= 1.5, Pyramid >= 1.3, CherryPy
- ...

```
sudo pip-${PYTHON-VERSION} install ${PACKAGE}
```

Differences by Example

Use a good reference! (available as Android app ;)

- Strings are Unicode by default
  - *Python 2 used byte arrays + character encoding as strings*
  - `u'...'` *is obsolete*
  - *Python 3 is less forgiving with mixing strings and byte arrays*
  - *Potential trouble interacting with C, the OS or the web*
  - *Indexing a single element from a byte array yields an integer (solution: use slicing)*
  - *Regular expressions also differentiate strings and byte arrays*
- `print` is now a function

  `print >>sys.stderr, 1, 2, 3` ⇒ `print(1, 2, 3, file=sys.stderr)`
  - *`sys.stderr.write(.)`*
- `xrange` ⇒ `range`
  - *Use `list()` if required (e.g. in interactive sessions)*
- try...except statement
  - *Python 2:* `except (RuntimeError, ImportError), e`
  - *Python 3:* `except (RuntimeError, ImportError) as e`
- Python 2

```python
# Python 3 differences
a_list = range(0, 10, 2)
squares = [x ** 2 for x in range(10)]

try:
    1 / 0
except (ZeroDivisionError, AttributeError), e:
    pass

a_string = u"text"
```

- Python 3

```python
# Python 3 differences
a_list = range(0, 10, 2)
a_list = list(range(0, 10, 2))

squares = [x ** 2 for x in range(10)]

try:
    1 / 0
except (ZeroDivisionError, AttributeError) as e:
    pass

a_string = "text"
```

- Floating point instead of floor division
- `long` data type removed
  - *sys.maxint ⇒ sys.maxsize*
- Relative imports
  - *All imports now are absolute by default*

    from . import submodule
- `__nonzero__(self) ⇒ __bool__(self)`
  - *Truth test for objects*

```python
class Stack(object):
    """
    Stump of a single stack in a pre-marshalling problem.

    A stack can only be filled or emptied from top. Each stack has a given
    maximum high.

    The highest priority of any container on the stack is 1. Increasing
    numbers mean decreasing priority.
    """

    def __init__(self, max_size):
        """
        Initialize a new empty stack.
        """
        self._containers = []
        self._max_size = max_size
        self._steps = 0

    def __len__(self):
        """
        Use the container list's len.
        """
        return len(self._containers)

    def __bool__(self):
        """
        All existing stacks are True.

        This is important to be able to test for existance. Otherwise empty
        stacks would evaluate to False (as __len__ is the backup for evaluating
        truth values and empty stacks would be False.
        """
        return True


def search_final_stack(container):
    """
    Return stack if found, otherwise None.

    Dummy implementation.
    """
    return stack


stack = Stack(5)
for _i in range(5):
    final_stack = search_final_stack(3)
    if final_stack:
        print("found a stack")
    else:
        print("no final stack available")
```

- `reduce(.)`

    - *moved to* `functools.reduce(.)`
- argparse
    - *successor of optparse*
- `iterator.next()` ⇒ `next(iterator)`

Case Study - Porting an Existing Program

1. Prerequisites
    - *Make sure you have a comprehensive battery of tests*
    - *Ensure that required libraries are available for Python 3*
    - *pip3 and nosetests3 are your friends*
2. Run the tests to see if they pass
3. Commit the current state to a VCS
4. Run 2to3
    1. *Convert multiple files by passing a directory*
    2. `2to3 -w dirname/`
    3. *Later, consider* `2to3 -w -f idioms dirname/`
    4. *Shebangs are **not adjusted***
5. Manual porting "loop"
    1. *Run the tests*
    2. *Fix failures*
    3. *Stop if all tests pass*
6. Run a view manual system tests
    - *If errors occur, create new unit tests*
    - *Fix remaining issues*
7. Commit the result to a VCS (maybe a new branch)

Flowchart
Comprehensive example

Conclusion

- Use Python 3 whenever possible
- Porting is a pain
- #1 problem: difference between strings and byte arrays
- Be sure to use version control before porting
- Unit tests are essential...
- Don't port anything without unit tests
- Unit tests are essential. Really!

Further Reading

MARK PILGRIM
Dive Into Python 3 (2nd edition)
Apress (October 23, 2009)

PYTHON SOFTWARE FOUNDATION
Python Documentation
http://docs.python.org/

PYTHON WIKI
Porting Python Code to 3.0
http://wiki.python.org/moin/PortingPythonToPy3k

PYTHON WIKI
Should I use Python 2 or Python 3 for my development activity?
http://wiki.python.org/moin/Python2orPython3

LENNART REGEBRO
Porting to Python 3: An in-depth guide
Createspace (2. März 2011)

GUIDO VAN ROSSUM
PEP 3000 -- Python 3000
http://www.python.org/dev/peps/pep-3000/

The following list contains all external links in order of appearance in the presentation

- Gerald Senarclens de Grancy,: http://www.senarclens.eu/~gerald/
- Python 3 porting status: http://wiki.python.org/moin/Python3PortingStatus
- twisted: http://twistedmatrix.com/trac/wiki/Plan/Python3
- PIL: http://stackoverflow.com/questions/3896286/image-library-for-python-3
- Current Linux distributions offer many Python 3 libraries: http://stackoverflow.com/a/3127755/104659
- PyPI: https://pypi.python.org/pypi
- Use a good reference!: http://getpython3.com/diveintopython3/porting-code-to-python-3-with-2to3.html
- argparse: http://docs.python.org/3/library/argparse.html
- Comprehensive example: http://getpython3.com/diveintopython3/case-study-porting-chardet-to-python-3.html
- Dive Into Python 3: http://getpython3.com/diveintopython3/
- Python Documentation: http://docs.python.org/
- Porting Python Code to 3.0: http://wiki.python.org/moin/PortingPythonToPy3k

- Should I use Python 2 or Python 3 for my development activity?: http://wiki.python.org/moin/Python2orPython3
- Porting to Python 3: An in-depth guide: http://python3porting.com/
- PEP 3000 -- Python 3000: http://www.python.org/dev/peps/pep-3000/