Eight Documentation

Release 0.0.1

Andrey Kislyuk

May 10, 2017

Contents

Eight is a Python module that provides a minimalist compatibility layer between Python 3 and 2. Eight lets you write code for Python 3.3+ while providing limited compatibility with Python 2.7 with no code changes. Eight is inspired by six, nine, and python-future, but provides better internationalization (i18n) support, is more lightweight, easier to use, and unambiguously biased toward Python 3 code: if you remove eight from your code, it will continue to function exactly as it did with eight on Python 3.

To write code for Python 3 that is portable to Python 2, you may also want to read Armin Ronacher's excellent Python 3 porting guide, as well as the official porting guide.

Writing `from eight import *` in your code is a no-op in Python 3. In Python 2, it binds a bunch of Python 3 names to their Python 2 equivalents. Also, if you need to import a module or module member that was renamed in Python 3, writing `from eight import <module>` will do the right thing (equivalent to `import <module>` on Python 3 and `import <old_name> as <module>` on Python 2). Finally, eight can optionally wrap your standard streams and environment variable I/O to use text, not bytes (see below).

CHAPTER 1

Installation

```
pip install eight
```

CHAPTER 2

Synopsis

```
from eight import *
from eight import queue
from eight.collections import UserList, deque
```

If you use `print`, division, non-ASCII literals, or relative imports, you should also add this future import at the top of each source file:

```
from __future__ import (print_function, division, unicode_literals, absolute_import)
```

CHAPTER 3

Wrapping stdio

Eight provides wrappers for `sys.stdin`, `sys.stdout`, and `sys.stderr` to make them (and methods that use them) behave like they do on Python 3. Specifically, in Python 3 these streams accept text data, and their `.buffer` attributes refer to the underlying streams that accept bytes. Eight uses the io module to do the same for you, but subclasses the TextIOWrapper class for `sys.stdout` and `sys.stderr` to coerce non-unicode input to unicode on Python 2 (otherwise, because of the Python 2 semantics, things like exception printing cease to work).

To enable stdio wrapping, use the following:

```
import eight
eight.wrap_stdio()
```

To revert the effects of this on any of the streams, use the detach method, e.g. `sys.stdin = sys.stdin.detach()` (but remember to condition this on `eight.USING_PYTHON2`). See the io module documentation for more information.

CHAPTER 4

Decoding command-line arguments

Eight provides a utility function to decode the contents of `sys.argv` on Python 2 (as Python 3 does). It uses `sys.stdin.encoding` as the encoding to do so:

```
import eight
eight.decode_command_line_args()
```

The call to `decode_command_line_args()` replaces `sys.argv` with its decoded contents and returns the new contents. On Python 3, the call is a no-op (it returns `sys.argv` and leaves it intact).

CHAPTER 5

Wrapping environment variable getters and setters

Eight provides utility wrappers to help bring Python 2 environment variable access and assignment in line with Python 3: encode the input to `os.putenv` (which is used for statements like `os.environ[x] = y`) and decode the output of `os.getenv` (used for `x = os.environ[y]`). Use `wrap_os_environ_io()` to monkey-patch these wrappers into the `os` module:

```
import eight
eight.wrap_os_environ_io()
```

On Python 3, the call is a no-op.

CHAPTER 6

Selecting from the buffet

You can see what `from eight import *` will do by running IPython and typing `import eight`, then `eight.<TAB>`. Here is a full list of what's available:

- `ascii`
- `bytes`
- `chr`
- `filter`
- `hex`
- `input`
- `int`
- `map`
- `oct`
- `open`
- `range`
- `round`
- `str`
- `super`
- `zip`

You can import these symbols by listing them explicitly. If for any reason you see an issue with importing them all (which is recommended), you can of course import a subset.

In addition to names imported by `from eight import *`, the following modules are available and should be imported by name using `from eight import <name>` when needed:

- `queue` (old name: `Queue`)
- `builtins` (old name: `__builtin__`)

- `copyreg` (old name: `copy_reg`)
- `configparser` (old name: `ConfigParser`)
- `reprlib` (old name: `repr`)
- `winreg` (old name: `_winreg`)
- `_thread` (old name: `thread`)
- `_dummy_thread` (old name: `dummy_thread`)

The following modules have attributes which resided elsewhere in Python 2: TODO

CHAPTER 7

Acknowledgments

Python-future for doing a bunch of heavy lifting on backports of Python 3 features.

CHAPTER 8

Links

- Project home page (GitHub)
- Documentation (Read the Docs)
- Package distribution (PyPI)

Bugs

Please report bugs, issues, feature requests, etc. on GitHub.

CHAPTER 9

License

Licensed under the terms of the Apache License, Version 2.0.

CHAPTER 10

API documentation

CHAPTER 11

Table of Contents