
transducer Documentation

Release 0.9

Sixty North

Jun 28, 2017

Contents

1	Contents	3
1.1	Front Matter	3
1.2	Narrative Documentation	4
1.3	Reference Documentation	4
1.4	Detailed Change History	6
1.5	Samples	7
2	Indices and tables	9
	Python Module Index	11

transducer is a Python package for ...

It is licensed under the MIT License.

Front Matter

Copyright

transducer

Copyright © 2014-2015 Sixty North

Official Website

<https://github.com/sixty-north/python-transducers>

License

Copyright (c) 2015 Sixty North AS

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the “Software”), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED “AS IS”, WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

Narrative Documentation

Read this to learn how to use `transducer`.

`transducer` Introduction

Current the best description of `transducer` can be found in our series of articles [Understanding Transducers Through Python](#). The code developed over the course of these articles is substantially the same as in this `transducer` package, although the package uses some further abstractions and tools which are largely irrelevant to understanding how transducers work.

Reference Documentation

Descriptions and examples for every public function, class and method in `transducer`.

API Reference

`transducer`

`transducer.transducers`

Functions for creating transducers.

The functions in this module return transducers.

Blah blah blah

Transducer Factories

Call these functions to obtain transducer objects.

mapping

Create a mapping transducer with the given transform.

mapping (*transform*)

Create a mapping transducer with the given transform.

Parameters `transform` – A single-argument function which will be applied to each input element to produce the corresponding output element.

Returns A mapping transducer: A single argument function which, when passed a reducing function, returns a new reducing function which applies the specified mapping transform function *before* delegating to the original reducer.

Examples

Mapping a squaring function over a list:

```
>>> from transducer.eager import transduce
>>> from transducer.reducers import appending
```



```
>>> from transducer.transducers import mapping
>>> m = mapping(lambda x: x*x)
>>> a = [1, 7, 9, 4, 3, 2]
>>> transduce(m, appending(), a)
[1, 49, 9, 4, 3, 2]
```

filtering (*predicate*)

Create a filtering transducer with the given predicate.

Parameters **predicate** – A single-argument function which will be used to test each element and which must return True or False. Only those elements for which this function returns True will be retained.

Returns A filtering transducer: A single argument function which, when passed a reducing function, returns a new reducing function which passes only those items in the input stream which match satisfy the predicate to the original reducer.

Examples

Filtering even numbers from a list:

```
>>> from transducer.eager import transduce
>>> from transducer.reducers import appending
>>> from transducer.transducers import filtering
>>> f = filtering(lambda x: x % 2 == 0)
>>> a = [1, 7, 9, 4, 3, 2]
>>> transduce(f, appending(), a)
[4, 2]
```

reducing (*reducer, init=UNSET*)

Create a reducing transducer with the given reducer.

Parameters **reducer** – A two-argument function which will be used to combine the partial cumulative result in the first argument with the next item from the input stream in the second argument.

Returns A reducing transducer: A single argument function which, when passed a reducing function, returns a new reducing function which entirely reduces the input stream using ‘reducer’ before passing the result to the reducing function passed to the transducer.

Examples

Reducing a list of numbers to a single value by summing them:

```
>>> from transducer.eager import transduce
>>> from transducer.reducers import expecting_single
>>> from transducer.transducers import reducing
>>> r = reducing(lambda x, y: x + y)
>>> a = [1, 7, 9, 4, 3, 2]
>>> transduce(r, expecting_single(), a)
>>> 26
```

`transducer.eager`

Blah blah blah

transduce

transduce (*transducer, reducer, iterable, init=UNSET*)

Examples

Eager transduction of a mapping over a list:

```
>>> from transducer.eager import transducer
>>> from transducer.reducers import appending
>>> from transducer.transducers import mapping
>>> m = mapping(lambda x: x*x)
>>> a = [1, 7, 9, 4, 3, 2]
>>> transduce(mapping, appending(), a)
[1, 49, 9, 4, 3, 2]
```

Differences from Clojure's transducers

Although `transducer` is inspired by Clojure's transducers, there are inevitably some differences in order to accommodate the variance between Clojure and Python.

Frequently Asked Questions

So far, there are none.

Detailed Change History

Changes

transducer 0.8

Reorganised the code – particularly for the coroutine based event processing: sources, sinks and the react transduce now exist in their own modules.

Clarified the semantics of the sources, which now no longer call `close()` on the targets.

Single-sourced the version from the `transducer/__init__.py` file.

Adds a new reducer for producing sets.

A new reducer for completing regular Python reducing functions into objects supporting the full reducer protocol.

Numerous fixes throughout the code as a result of vastly improved test coverage.

transducer 0.7

Rework to make the code closer to the code in the blog series.

Much clearer reduction to single values for transducers such as `first()` and `last()` in conjunction with the `expecting_single()` reducer.

Simplified and clarified much of the implementation.

Dropped support for Python 3.2.

transducer 0.6

Correctness fixes by moving function locals in the transducer factories to be class attributes of the transducers. This allows transducers returned by the transducer factories to be safely used more than once.

Renamed `transducer.transducer` to `transducer.transducers`.

Relocate reducing functions to `reducers.py`

Move transducer infrastructure to `infrastructure.py`

transducer 0.5

Initial release

Samples

More complex examples of non-trivial usage of `transducer`:

Samples

One day ...

CHAPTER 2

Indices and tables

- `genindex`
- `modindex`
- `search`

t

`transducer.eager`, 6

`transducer.transducers`, 4

F

`filtering()` (in module `transducer.transducers`), 5

M

`mapping()` (in module `transducer.transducers`), 4

R

`reducing()` (in module `transducer.transducers`), 5

T

`transduce()` (in module `transducer.eager`), 6

`transducer.eager` (module), 6

`transducer.transducers` (module), 4