

# Package ‘foldr’

June 4, 2013

**Type** Package

**Version** 0.1-1

**Date** 2013-05-29

**Title** A collection of Python-esque data types

**Author** Greg Lamp and Austin Ogilvie

**Maintainer** Greg Lamp <greg@yhathq.com>

**Depends** R (>= 2.12.0)

**Imports** plyr, digest, methods

**Suggests** testthat (>= 0.2)

**Description** foldr provides Python-like data types (list and dict) in R

**License** FreeBSD

**URL** <https://github.com/yhat/foldr>

**BugReports** <https://github.com/yhat/foldr/issues>

**Collate** ‘pydict.R’ ‘pylist.R’ ‘utils.r’

## R topics documented:

dict.py . . . . .	2
dict_repl . . . . .	2
encapsulate . . . . .	2
foldr . . . . .	3
list.py . . . . .	3
merge.list . . . . .	4
zip.dict . . . . .	4
zip.tuple . . . . .	5

<b>Index</b>	<b>6</b>
--------------	----------

---

dict.py	<i>Creates an instance of a dict</i>
---------	--------------------------------------

---

### Description

This is a wrapper function around the `pydict$new` that is a little more R friendly.

### Usage

```
dict.py(...)
```

### Arguments

... a series of key/value pairs in the form `key=value`

### Examples

```
(x <- dict.py("a"=1, "b"=2, "c"=3))
#{a: 1, b: 2, c: 3}
```

---

dict_repl	<i>Function for representing hashed objects as strings</i>
-----------	--

---

### Description

Purely visual.

### Usage

```
dict_repl(object, obj_name)
```

### Arguments

object	an arbitrary thing
obj_name	name of the variable as defined by the user (not currently being used)

---

encapsulate	<i>Helper function for making character vectors have quotes around each item when printed to the console.</i>
-------------	---

---

### Description

Helper function for making character vectors have quotes around each item when printed to the console.

### Usage

```
encapsulate(values)
```

### Arguments

values	a vector of values
--------	--------------------

---

foldr*dicts and lists in R*

---

**Description**

foldr provides functionality similar to dicts and lists in Python and Ruby. Everything in foldr is object oriented and copies much of the API from Python and Ruby.

**Details**

Creating a list

blah blah blah

Merging lists

use the + operator

Creating a dict

blah blah blah

Iteration

blah bla blah; pitfalls, shortcomings, features...

When not to use them

Don't use them with large sets of data...

**References**

blog post by Yhat <http://blog.yhathq.com/a-blog-post/>.

---

list.py*Creates an instance of a list*

---

**Description**

This is a wrapper function around the `pylist$new` that is a little more R friendly.

**Usage**

```
list.py(...)
```

**Arguments**

... a series of values seperated by a comma. NOTE: a vector will be treated as an individual item. i.e. `list.py(1:100)` will yield a list with 1 item, whereas `list.py(1, 2, 3, 4)` will yield a list with 4 items

**Examples**

```
x <- list.py(1, 2, 3, 4)
#[1, 2, 3, 4]
```

---

`merge.list`*Function that takes 2 lists and merges them fairly effeciently*

---

**Description**

Function that takes 2 lists and merges them fairly effeciently

**Usage**

```
merge.list(x, y = NULL, mergeUnnamed = TRUE, ...)
```

**Arguments**

<code>x</code>	a list
<code>y</code>	a second list
<code>mergeUnnamed</code>	boolean for whether or not to include list items with no names
<code>...</code>	whatever else you've got

---

`zip.dict`*Combine 2 lists into a dict of key/values*

---

**Description**

Takes 2 lists and converts them into a key => value mapping, which takes the form of a [dict.py](#).

**Usage**

```
zip.dict(x, y)
```

**Arguments**

<code>x</code>	a list, vector, or list.py
<code>y</code>	a second list, vector, or list.py

**Examples**

```
x <- list.py(1, 2, 3)
y <- list.py("a", "b", "c")
zip.dict(x, y)
#{1: 'a', 2: 'b', 3: 'c'}
zip.dict(y, x)
#{'a': 1, 'b': 2, 'c': 3}
```

---

zip.tuple

---

*Combine 2 lists into a list of lists*

---

**Description**

Return a list of 2 item lists, where each list contains the i-th element from each of the argument sequences. The returned list is truncated in length to the length of the shortest argument sequence.

**Usage**

```
zip.tuple(x, y)
```

**Arguments**

x	a list, vector, or list.py
y	a second list, vector, or list.py

**Examples**

```
x <- list.py(1, 2, 3)
y <- list.py(4, 5, 6)
zip.tuple(x, y)
#[[1, 4], [2, 5], [3, 6]]
y <- list.py("a", "b", "c")
zip.tuple(x, y)
#[[1, 'a'], [2, 'b'], [3, 'c']]
```

# Index

- \*Topic **dict**,
  - dict.py, [2](#)
  - zip.dict, [4](#)
- \*Topic **dict.py**,
  - dict.py, [2](#)
- \*Topic **key/value**
  - dict.py, [2](#)
- \*Topic **list**,
  - list.py, [3](#)
- \*Topic **list.py**
  - list.py, [3](#)
- \*Topic **lists**,
  - zip.tuple, [5](#)
- \*Topic **lists**
  - zip.dict, [4](#)
- \*Topic **zip**,
  - zip.dict, [4](#)
- \*Topic **zip**
  - zip.tuple, [5](#)

dict.py, [2](#), [4](#)  
dict\_repl, [2](#)

encapsulate, [2](#)

foldr, [3](#)  
foldr-package (foldr), [3](#)

list.py, [3](#)

merge.list, [4](#)

zip.dict, [4](#)  
zip.tuple, [5](#)