

Package ‘foldr’

June 21, 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:

as.character	2
cos	2
cumsum	2
dict.py	3
dict_repl	3
encapsulate	3
hist	4
is.dict.py	4
is.list.py	4
lapply	5
length	5
list.py	5
merge.list	6
paste	6

plot	6
sapply	6
sign	7
sin	7
sum	7
summary	7
toString	7
zip.dict	8
zip.tuple	8

Index	9
--------------	----------

as.character	<i>Turns a list into a character vector.</i>
--------------	--

Description

Generic function that calls the string method for a list.

Arguments

- x a list
- ... named args

Examples

```
as.character(list.py(1, 2, 3, 4))
```

cos	<i>Calculates the cos of the items in a list</i>
-----	--

Description

Generic function for caculating the cos of the items in a list. If an item is not numeric an error occurs.

cumsum	<i>Calculates the cumsum of the items in a list</i>
--------	---

Description

Generic function for caculating the cumsum of the items in a list. If an item is not numeric an error occurs.

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

hist	<i>Plots a histogram of the items of a list.</i>
------	--

Description

Generic function that plots a histogram of the items in a list.

is.dict.py	<i>Determines whether or not an object is an instance of a dictionary.</i>
------------	--

Description

Determines the class of an object and checks to see if it's a dictionary.

Usage

```
is.dict.py(object)
```

Arguments

object	any object
--------	------------

Examples

```
x <- dict.py("a"=1)
is.dict.py(x)
#TRUE
x <- list(1, 2, 3, 4)
is.dict.py(x)
#FALSE
```

is.list.py	<i>Determines whether or not an object is an instance of a list</i>
------------	---

Description

Determines the class of an object and checks to see if it's a list

Usage

```
is.list.py(object)
```

Arguments

object	any object
--------	------------

Examples

```
x <- list.py("a")
is.list.py(x)
#TRUE
x <- 1:10
is.dict.py(x)
#FALSE
```

lapply

*Wrapper around lapply.***Description**

Automatically invokes lapply on the items in the list.

length

*Function for getting the number of items in a dictionary.***Description**

Use much like length(list()) or length(c(1, 2, 3)).

Use much like length(list(1, 2, 3)) or length(c(1, 2, 3)).

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

`paste`*Turns a list into a printable string*

Description

Generic function that calls the toString method for a list.

`plot`*Plots a scatterplot of the items of a list.*

Description

Generic function that plots a scatterplot of the items in a list.

`supply`*Wrapper around supply.*

Description

Automatically invokes supply on the items in the list.

sign	<i>Calculates the sign of the items in a list</i>
------	---

Description

Generic function for calculating the sign of the items in a list. If an item is not numeric an error occurs.

sin	<i>Calculates the sin of the items in a list</i>
-----	--

Description

Generic function for calculating the sin of the items in a list. If an item is not numeric an error occurs.

sum	<i>Calculates the sum of the items in a list</i>
-----	--

Description

Generic function for calculating the sum of the items in a list. If an item is not numeric an error occurs.

summary	<i>Creates a summary of the items in a list.</i>
---------	--

Description

Summarizes the list by data type. Each data type gets its own summary with the results put into a native R list.

toString	<i>Turns a list into a string.</i>
----------	------------------------------------

Description

Generic function that calls the string method for a list.

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

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("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, 3
 - zip.dict, 8
- *Topic **dict.py**,
 - dict.py, 3
- *Topic **key/value**
 - dict.py, 3
- *Topic **list**,
 - list.py, 5
- *Topic **list.py**
 - list.py, 5
- *Topic **lists**,
 - zip.tuple, 8
- *Topic **lists**
 - zip.dict, 8
- *Topic **zip**,
 - zip.dict, 8
- *Topic **zip**
 - zip.tuple, 8
- as.character, 2
- cos, 2
- cumsum, 2
- dict.py, 3, 8
- dict_repl, 3
- encapsulate, 3
- hist, 4
- is.dict.py, 4
- is.list.py, 4
- lapply, 5
- length, 5
- list.py, 5
- merge.list, 6
- paste, 6
- plot, 6
- sapply, 6
- sign, 7
- sin, 7
- sum, 7
- summary, 7
- toString, 7
- zip.dict, 8
- zip.tuple, 8