pydash Documentation

Release 1.1.0

Derrick Gilland

August 19, 2014

Contents

Python port of the Lo-Dash Javascript library.

CHAPTER 1

Links

- Project: https://github.com/dgilland/pydash
- Documentation: http://pydash.readthedocs.org
- PyPi: https://pypi.python.org/pypi/pydash/
- TravisCI: https://travis-ci.org/dgilland/pydash

CHAPTER 2

Quickstart

```
>>> import pydash as pyd

# Arrays
>>> pyd.flatten([1, 2, [3, [4, 5, [6, 7]]]])
[1, 2, 3, 4, 5, 6, 7]

# Collections
>>> pyd.pluck([{'name': 'moe', 'age': 40}, {'name': 'larry', 'age': 50}], 'name')
['moe', 'larry']

# Functions
>>> curried = pyd.curry(lambda a, b, c: a + b + c)
>>> curried(1, 2)(3)
6

# Objects
>>> pyd.omit({'name': 'moe', 'age': 40}, 'age')
{'name': 'moe'}

# Utilities
>>> pyd.times(3, lambda index: index)
[0, 1, 2]

# Chaining
>>> pyd.chain([1, 2, 3, 4]).without(2, 3).reject(lambda x, *args: x > 1).value()
[1]
```

See also:

For further details consult *API Reference*.

CHAPTER 3

Guide

3.1 Installation

pydash requires Python >= 2.6 or >= 3.2. It has no external dependencies.

To install from PyPi:

```
pip install pydash
```

3.2 Quickstart

```
>>> import pydash as pyd

# Arrays
>>> pyd.flatten([1, 2, [3, [4, 5, [6, 7]]]])
[1, 2, 3, 4, 5, 6, 7]

# Collections
>>> pyd.pluck([{'name': 'moe', 'age': 40}, {'name': 'larry', 'age': 50}], 'name')
['moe', 'larry']

# Functions
>>> curried = pyd.curry(lambda a, b, c: a + b + c)
>>> curried(1, 2)(3)
6

# Objects
>>> pyd.omit({'name': 'moe', 'age': 40}, 'age')
{'name': 'moe'}

# Utilities
>>> pyd.times(3, lambda index: index)
[0, 1, 2]

# Chaining
>>> pyd.chain([1, 2, 3, 4]).without(2, 3).reject(lambda x, *args: x > 1).value()
[1]
```

See also:

For further details consult *API Reference*.

3.3 Lo-Dash Differences

- Function names use `snake_case` instead of `camelCase`.
- Any Lo-Dash function that shares its name with a reserved Python keyword will have an _ appended after it (e.g. `filter` in Lo-Dash would be `filter_` in pydash.
- Extra callback args must be explictly handled. In Javascript, it's perfectly fine to pass in extra arguments to a function that aren't explictly accepted by that function (e.g. `function foo(a1){}; foo(1, 2, 3);`). In Python, those extra arguments must be explictly handled (e.g. `def foo(a1, *args): ...; foo(1, 2, 3)`). Therefore, callbacks passed to `pydash` functions must use named args or a catch-all like `*args` since each callbacks will be passed arguments like `(value, index|key, array)`. **NOTE:** *Future versions of pydash may attempt to infer the number of arguments a callback can handle and only pass what's supported.*
- Lo-Dash's `toArray()` is pydash's `to_list()`.
- In addition to `property_()`, pydash has `prop()` as an alias.
- In addition to `escape_reg_exp()` and `is_reg_exp()`, pydash has `escape_re()` and `is_re()` as aliases.
- pydash's `memoize()` uses all passed in arguments as the cache key by default instead of only using the first argument like Lo-Dash does.
- pydash doesn't have `template()`. See *Templating* for more details.

3.4 Templating

Currenly, pydash doesn't support templating. Having a custom templating engine was never a goal of pydash even though Lo-Dash includes one. There already exist many mature and battle-tested templating engines like Jinja2 and Mako which would be much more suited to handling templating needs. However, how best to implement templating in pydash remains an open issue.

Here are a few options to consider:

1. Provide a thin wrapper around String Format and facilitate HTML escaping but provide nothing beyond that; no code execution, no interpolate delimiters, no imports.
2. Provide a thin wrapper around one of the templating libraries (i.e. choose one and make it a dependency) and defer everything to it.
3. Provide an interface to a templating library so that you get `pydash.template()` but you set the engine yourself.
4. All of the above.
5. Leave templating out of pydash.

No decision has been made yet, but one will likely be made by `v1.2.0` or `v1.3.0`.

3.5 Versioning

This project follows Semantic Versioning with the following caveats:

- Only the public API (i.e. the objects imported into the `pydash` module) will maintain backwards compatibility between MINOR version bumps.

- Objects within any other parts of the library are not guaranteed to not break between MINOR version bumps.

With that in mind, it is recommended to only use or import objects from the main module, `pydash`.

CHAPTER 4

API Reference

Includes links to source code.

4.1 API Reference

All public functions are available from the main module.

```
import pydash

pydash.<function>
```

4.1.1 Arrays

Functions that operate on lists.

New in version 1.0.0.

pydash.api.arrays.**chunk**(*array*, *size=1*)

Creates a list of elements split into groups the length of *size*. If *array* can't be split evenly, the final chunk will be the remaining elements.

Parameters

- **array** (*list*) – List to chunk.
- **size** (*int, optional*) – Chunk size. Defaults to `1`.

Returns New list containing chunks of *array*.

Return type list

New in version 1.1.0.

pydash.api.arrays.**compact**(*array*)

Creates a list with all falsey values of array removed.

Parameters **array** (*list*) – List to compact.

Returns Compacted list.

Return type list

New in version 1.0.0.

pydash.api.arrays.**difference**(*array*, **lists*)

Creates a list of list elements not present in the other lists.

Parameters

- **array** (*list*) – List to process.
- **lists** (*list*) – Lists to check.

Returns Difference of the lists.

Return type list

New in version 1.0.0.

pydash.api.arrays.**drop**(*array*, *n*)

Creates a slice of *array* with *n* elements dropped from the beginning.

Parameters

- **array** (*list*) – List to process.
- **n** (*int*) – Number of elements to drop.

Returns Dropped list.

Return type list

New in version 1.0.0.

Changed in version 1.1.0: Added `n` argument and removed as alias of `rest()`.

pydash.api.arrays.**drop_right**(*array*, *n*)

Creates a slice of *array* with *n* elements dropped from the end.

Parameters

- **array** (*list*) – List to process.
- **n** (*int*) – Number of elements to drop.

Returns Dropped list.

Return type list

New in version 1.1.0.

pydash.api.arrays.**drop_right_while**(*array*, *callback=None*)

Creates a slice of *array* excluding elements dropped from the end. Elements are dropped until the *callback* returns falsey. The *callback* is invoked with three arguments: `(value, index, array)`.

Parameters

- **array** (*list*) – List to process.
- **callback** (*mixed*) – Callback called per iteration

Returns Dropped list.

Return type list

New in version 1.1.0.

pydash.api.arrays.**drop_while**(*array*, *callback=None*)

Creates a slice of *array* excluding elements dropped from the beginning. Elements are dropped until the *callback* returns falsey. The *callback* is invoked with three arguments: `(value, index, array)`.

Parameters

- **array** (*list*) – List to process.
- **callback** (*mixed*) – Callback called per iteration

Returns Dropped list.

Return type list

New in version 1.1.0.

pydash.api.arrays.**find_index**(*array*, *callback=None*)

This method is similar to pydash.api.collections.find(), except that it returns the index of the element that passes the callback check, instead of the element itself.

Parameters

- **array** (*list*) – List to process.
- **callback** (*mixed, optional*) – Callback applied per iteration.

Returns Index of found item or `-1` if not found.

Return type int

New in version 1.0.0.

pydash.api.arrays.**find_last_index**(*array*, *callback=None*)

This method is similar to find_index(), except that it iterates over elements from right to left.

Parameters

- **array** (*list*) – List to process.
- **callback** (*mixed, optional*) – Callback applied per iteration.

Returns Index of found item or `-1` if not found.

Return type int

New in version 1.0.0.

pydash.api.arrays.**first**(*array*)

Return the first element of *array*.

Parameters **array** (*list*) – List to process.

Returns First element of list.

Return type mixed

See also:

- first() (main definition)
- head() (alias)
- take() (alias)

New in version 1.0.0.

pydash.api.arrays.**flatten**(*array*, *callback=None*, *_depth=0*)

Flattens a nested array (the nesting can be to any depth). If callback is True, array will only be flattened a single level. If callback is passed, each element of array is passed through a callback before flattening.

Parameters

- **array** (*list*) – List to process.

- **callback** (*mixed, optional*) – Callback applied per iteration. If `True` then flatten shallowly.

Returns Flattened list.

Return type list

New in version 1.0.0.

`pydash.api.arrays.`**`head`**(*array*)
Return the first element of *array*.

Parameters **array** (*list*) – List to process.

Returns First element of list.

Return type mixed

See also:

- `first()` (main definition)
- `head()` (alias)
- `take()` (alias)

New in version 1.0.0.

`pydash.api.arrays.`**`index_of`**(*array*, *value*, *from_index=0*)
Gets the index at which the first occurrence of value is found.

Parameters

- **array** (*list*) – List to search.
- **value** (*mixed*) – Value to search for.
- **from_index** (*int, optional*) – Index to search from.

Returns Index of found item or `-1` if not found.

Return type int

New in version 1.0.0.

`pydash.api.arrays.`**`initial`**(*array*)
Return all but the last element of *array*.

New in version 1.0.0.

`pydash.api.arrays.`**`intersection`**(**arrays*)
Computes the intersection of all the passed-in arrays.

Parameters **arrays** (*list*) – Lists to process.

Returns Intersection of provided lists.

Return type list

New in version 1.0.0.

`pydash.api.arrays.`**`last`**(*array*)
Return the last element of *array*.

New in version 1.0.0.

`pydash.api.arrays.`**`last_index_of`**(*array*, *value*, *from_index=None*)
Gets the index at which the last occurrence of value is found.

Parameters

- **array** (*list*) – List to search.
- **value** (*mixed*) – Value to search for.
- **from_index** (*int, optional*) – Index to search from.

Returns Index of found item or `False` if not found.

Return type int

New in version 1.0.0.

`pydash.api.arrays.`**`object_`**(*keys*, *values=None*)

Creates a dict composed from lists of keys and values. Pass either a single two dimensional list, i.e. `[[key1, value1], [key2, value2]]`, or two lists, one of keys and one of corresponding values.

Parameters

- **keys** (*list*) – either a list of keys or a list of `[key, value]` pairs
- **values** (*list, optional*) – list of values to zip

Returns Zipped dict.

Return type dict

See also:

- `zip_object()` (main definition)
- `object_()` (alias)

New in version 1.0.0.

`pydash.api.arrays.`**`pull`**(*array*, **values*)

Removes all provided values from the given array.

Parameters

- **array** (*list*) – List to pull from.
- **values** (*mixed*) – Values to remove.

Returns Modified *array*.

Return type list

Warning: *array* is modified in place.

New in version 1.0.0.

`pydash.api.arrays.`**`pull_at`**(*array*, **indexes*)

Removes elements from *array* corresponding to the specified indexes and returns a list of the removed elements. Indexes may be specified as a list of indexes or as individual arguments.

Parameters **array** (*list*) – List to pull from.

Returns Modified *array*.

Return type list

Warning: *array* is modified in place.

New in version 1.1.0.

pydash.api.arrays.**remove**(*array*, *callback=None*)

Removes all elements from a list that the callback returns truthy for and returns an array of removed elements.

Parameters

- **array** (*list*) – List to remove elements from.
- **callback** (*mixed, optional*) – Callback applied per iteration.

Returns Removed elements of *array*.

Return type list

Warning: *array* is modified in place.

New in version 1.0.0.

pydash.api.arrays.**rest**(*array*)

Return all but the first element of *array*.

Parameters **array** (*list*) – List to process.

Returns Rest of the list.

Return type list

See also:

- `rest()` (main definition)
- `tail()` (alias)
- `drop()` (alias)

New in version 1.0.0.

pydash.api.arrays.**slice_**(*array*, *start*, *end*)

Slices *array* from the *start* index up to, but not including, the *end* index.

Parameters

- **array** (*list*) – Array to slice.
- **start** (*int*) – Start index.
- **end** (*int*) – End index.

Returns Sliced list.

Return type list

New in version 1.1.0.

pydash.api.arrays.**sorted_index**(*array*, *value*, *callback=None*)

Determine the smallest index at which the value should be inserted into array in order to maintain the sort order of the sorted array. If callback is passed, it will be executed for value and each element in array to compute their sort ranking. The callback is invoked with one argument: `(value)`. If a property name is passed for callback, the created `pydash.api.collections.pluck()` style callback will return the property value of the given element. If an object is passed for callback, the created `pydash.api.collections.where()` style callback will return `True` for elements that have the properties of the given object, else `False`.

Parameters

- **array** (*list*) – List to inspect.
- **value** (*mixed*) – Value to evaluate.
- **callback** (*mixed, optional*) – Callback to determine sort key.

Returns Smallest index.

Return type int

New in version 1.0.0.

pydash.api.arrays.**sorted_last_index**(*array*, *value*, *callback=None*)

This method is like `sorted_index()` except that it returns the highest index at which a value should be inserted into a given sorted array in order to maintain the sort order of the array.

Parameters

- **array** (*list*) – List to inspect.
- **value** (*mixed*) – Value to evaluate.
- **callback** (*mixed, optional*) – Callback to determine sort key.

Returns Highest index.

Return type int

New in version 1.1.0.

pydash.api.arrays.**tail**(**args*, ***kargs*)

Return all but the first element of *array*.

New in version 1.0.0.

Deprecated since version 1.1.0: Use `rest()` instead.

pydash.api.arrays.**take**(*array*, *n*)

Creates a slice of *array* with *n* elements taken from the beginning.

Parameters

- **array** (*list*) – List to process.
- **n** (*int*) – Number of elements to take.

Returns Taken list.

Return type list

New in version 1.0.0.

Changed in version 1.1.0: Added `n` argument and removed as alias of `first()`.

pydash.api.arrays.**take_right**(*array*, *n*)

Creates a slice of *array* with *n* elements taken from the end.

Parameters

- **array** (*list*) – List to process.
- **n** (*int*) – Number of elements to take.

Returns Taken list.

Return type list

New in version 1.1.0.

pydash.api.arrays.**take_right_while**(*array*, *callback=None*)

Creates a slice of *array* with elements taken from the end. Elements are taken until the *callback* returns falsey. The *callback* is invoked with three arguments: `(value, index, array)`.

Parameters

- **array** (*list*) – List to process.
- **callback** (*mixed*) – Callback called per iteration

Returns Dropped list.

Return type list

New in version 1.1.0.

pydash.api.arrays.**take_while**(*array*, *callback=None*)

Creates a slice of *array* with elements taken from the beginning. Elements are taken until the *callback* returns falsey. The *callback* is invoked with three arguments: `(value, index, array)`.

Parameters

- **array** (*list*) – List to process.
- **callback** (*mixed*) – Callback called per iteration

Returns Taken list.

Return type list

New in version 1.1.0.

pydash.api.arrays.**union**(**arrays*)

Computes the union of the passed-in arrays.

Parameters **arrays** (*list*) – Lists to unionize.

Returns Unionized list.

Return type list

New in version 1.0.0.

pydash.api.arrays.**uniq**(*array*, *callback=None*)

Creates a duplicate-value-free version of the array. If callback is passed, each element of array is passed through a callback before uniqueness is computed. The callback is invoked with three arguments: `(value, index, array)`. If a property name is passed for callback, the created `pydash.api.collections.pluck()` style callback will return the property value of the given element. If an object is passed for callback, the created `pydash.api.collections.where()` style callback will return `True` for elements that have the properties of the given object, else `False`.

Parameters

- **array** (*list*) – List to process.
- **callback** (*mixed, optional*) – Callback applied per iteration.

Returns Unique list.

Return type list

See also:

- `uniq()` (main definition)
- `unique()` (alias)

New in version 1.0.0.

pydash.api.arrays.**unique**(*array*, *callback=None*)

Creates a duplicate-value-free version of the array. If callback is passed, each element of array is passed through a callback before uniqueness is computed. The callback is invoked with three arguments: `(value, index, array)`. If a property name is passed for callback, the created `pydash.api.collections.pluck()` style callback will return the property value of the given element. If an object is passed for callback, the created `pydash.api.collections.where()` style callback will return `True` for elements that have the properties of the given object, else `False`.

Parameters

- **array** (*list*) – List to process.
- **callback** (*mixed, optional*) – Callback applied per iteration.

Returns Unique list.

Return type list

See also:

- `uniq()` (main definition)
- `unique()` (alias)

New in version 1.0.0.

pydash.api.arrays.**without**(*array*, **values*)

Creates an array with all occurrences of the passed values removed.

Parameters

- **array** (*list*) – List to filter.
- **values** (*mixed*) – Values to remove.

Returns Filtered list.

Return type list

New in version 1.0.0.

pydash.api.arrays.**xor**(*array*, **lists*)

Creates a list that is the symmetric difference of the provided lists.

New in version 1.0.0.

pydash.api.arrays.**zip_**(**arrays*)

Groups the elements of each array at their corresponding indexes. Useful for separate data sources that are coordinated through matching array indexes.

Parameters **arrays** (*list*) – Lists to process.

Returns Zipped list.

Return type list

New in version 1.0.0.

pydash.api.arrays.**unzip**(*array*)

The inverse of `zip_()`, this method splits groups of elements into lists composed of elements from each group at their corresponding indexes.

Parameters **array** (*list*) – List to process.

Returns Unzipped list.

Return type list

New in version 1.0.0.

`pydash.api.arrays.`**`zip_object`**(*keys*, *values=None*)

Creates a dict composed from lists of keys and values. Pass either a single two dimensional list, i.e. `[[key1, value1], [key2, value2]]`, or two lists, one of keys and one of corresponding values.

Parameters

- **keys** (*list*) – either a list of keys or a list of `[key, value]` pairs
- **values** (*list, optional*) – list of values to zip

Returns Zipped dict.

Return type dict

See also:

- `zip_object()` (main definition)
- `object_()` (alias)

New in version 1.0.0.

4.1.2 Chaining

Chaining

New in version 1.0.0.

`pydash.api.chaining.`**`chain`**(*value*)

Creates a `Chain` object which wraps the given value to enable intuitive method chaining.

Parameters **value** (*mixed*) – Value to initialize chain operations with.

Returns `Chain`: Instance of `Chain` initialized with *value*.

New in version 1.0.0.

`pydash.api.chaining.`**`tap`**(*value*, *interceptor*)

Invokes interceptor with the value as the first argument and then returns value. The purpose of this method is to "tap into" a method chain in order to perform operations on intermediate results within the chain.

Parameters

- **value** (*mixed*) – Current value of chain operation.
- **interceptor** (*function*) – Function called on *value*.

Returns *value* after *interceptor* call.

Return type mixed

New in version 1.0.0.

4.1.3 Collections

Functions that operate on lists and dicts.

New in version 1.0.0.

pydash.api.collections.**all_**(*collection*, *callback=None*)

Checks if the callback returns a truthy value for all elements of a collection. The callback is invoked with three arguments: `(value, index|key, collection)`. If a property name is passed for callback, the created `pluck()` style callback will return the property value of the given element. If an object is passed for callback, the created `where()` style callback will return `True` for elements that have the properties of the given object, else `False`.

Parameters

- **collection** (*list|dict*) – Collection to iterate over.
- **callback** (*mixed, optional*) – Callback applied per iteration.

Returns Whether all elements are truthy.

Return type bool

See also:

- `every()` (main definition)
- `all_()` (alias)

pydash.api.collections.**any_**(*collection*, *callback=None*)

Checks if the callback returns a truthy value for any element of a collection. The callback is invoked with three arguments: `(value, index|key, collection)`. If a property name is passed for callback, the created `pluck()` style callback will return the property value of the given element. If an object is passed for callback, the created `where()` style callback will return `True` for elements that have the properties of the given object, else `False`.

Parameters

- **collection** (*list|dict*) – Collection to iterate over.
- **callbacked** (*mixed, optional*) – Callback applied per iteration.

Returns Whether any of the elements are truthy.

Return type bool

See also:

- `some()` (main definition)
- `any_()` (alias)

pydash.api.collections.**at**(*collection*, **indexes*)

Creates a list of elements from the specified indexes, or keys, of the collection. Indexes may be specified as individual arguments or as arrays of indexes.

Parameters

- **collection** (*list|dict*) – Collection to iterate over.
- **indexes** (*mixed*) – The indexes of *collection* to retrieve, specified as individual indexes or arrays of indexes.

Returns filtered list

Return type list

pydash.api.collections.**collect**(*collection*, *callback=None*)

Creates an array of values by running each element in the collection through the callback. The callback is invoked with three arguments: `(value, index|key, collection)`. If a property name is passed for callback, the created `pluck()` style callback will return the property value of the given element. If an object is passed for callback, the created `where()` style callback will return `True` for elements that have the properties of the given object, else `False`.

Parameters

- **collection** (*list|dict*) – Collection to iterate over.
- **callback** (*mixed, optional*) – Callback applied per iteration.

Returns Mapped list.

Return type list

See also:

- `map_()` (main definition)
- `collect()` (alias)

pydash.api.collections.**contains**(*collection*, *target*, *from_index=0*)

Checks if a given value is present in a collection. If *from_index* is negative, it is used as the offset from the end of the collection.

Parameters

- **collection** (*list|dict*) – Collection to iterate over.
- **target** (*mixed*) – Target value to compare to.
- **from_index** (*int, optional*) – Offset to start search from.

Returns Whether *target* is in *collection*.

Return type bool

See also:

- `contains()` (main definition)
- `include()` (alias)

pydash.api.collections.**count_by**(*collection*, *callback=None*)

Creates an object composed of keys generated from the results of running each element of *collection* through the callback.

Parameters

- **collection** (*list|dict*) – Collection to iterate over.
- **callback** (*mixed, optional*) – Callback applied per iteration.

Returns Dict containing counts by key.

Return type dict

pydash.api.collections.**detect**(*collection*, *callback=None*)

Iterates over elements of a collection, returning the first element that the callback returns truthy for.

Parameters

- **collection** (*list|dict*) – Collection to iterate over.
- **callback** (*mixed, optional*) – Callback applied per iteration.

Returns First element found or `None`.

Return type mixed

See also:

- `find()` (main definition)
- `detect()` (alias)
- `find_where()` (alias)

pydash.api.collections.**each**(*collection*, *callback=None*)

Iterates over elements of a collection, executing the callback for each element.

Parameters

- **collection** (*list|dict*) – Collection to iterate over.
- **callback** (*mixed, optional*) – Callback applied per iteration.

Returns list|dict: *collection*

See also:

- `for_each()` (main definition)
- `each()` (alias)

pydash.api.collections.**each_right**(*collection*, *callback*)

This method is like `for_each()` except that it iterates over elements of a *collection* from right to left.

Parameters

- **collection** (*list|dict*) – Collection to iterate over.
- **callback** (*mixed, optional*) – Callback applied per iteration.

Returns list|dict: *collection*

See also:

- `for_each_right()` (main definition)
- `each_right()` (alias)

pydash.api.collections.**every**(*collection*, *callback=None*)

Checks if the callback returns a truthy value for all elements of a collection. The callback is invoked with three arguments: `(value, index|key, collection)`. If a property name is passed for callback, the created `pluck()` style callback will return the property value of the given element. If an object is passed for callback, the created `where()` style callback will return `True` for elements that have the properties of the given object, else `False`.

Parameters

- **collection** (*list|dict*) – Collection to iterate over.
- **callback** (*mixed, optional*) – Callback applied per iteration.

Returns Whether all elements are truthy.

Return type bool

See also:

- `every()` (main definition)
- `all_()` (alias)

`pydash.api.collections.`**`filter_`**`(collection, callback=None)`

Iterates over elements of a collection, returning an list of all elements the callback returns truthy for.

Parameters

- **collection** (*list|dict*) – Collection to iterate over.
- **callback** (*mixed, optional*) – Callback applied per iteration.

Returns Filtered list.

Return type list

See also:

- `select()` (main definition)
- `filter_()` (alias)

`pydash.api.collections.`**`find`**`(collection, callback=None)`

Iterates over elements of a collection, returning the first element that the callback returns truthy for.

Parameters

- **collection** (*list|dict*) – Collection to iterate over.
- **callback** (*mixed, optional*) – Callback applied per iteration.

Returns First element found or `None`.

Return type mixed

See also:

- `find()` (main definition)
- `detect()` (alias)
- `find_where()` (alias)

`pydash.api.collections.`**`find_last`**`(collection, callback=None)`

This method is like `find()` except that it iterates over elements of a *collection* from right to left.

Parameters

- **collection** (*list|dict*) – Collection to iterate over.
- **callback** (*mixed, optional*) – Callback applied per iteration.

Returns Last element found or `None`.

Return type mixed

`pydash.api.collections.`**`find_where`**`(collection, callback=None)`

Iterates over elements of a collection, returning the first element that the callback returns truthy for.

Parameters

- **collection** (*list|dict*) – Collection to iterate over.
- **callback** (*mixed, optional*) – Callback applied per iteration.

Returns First element found or `None`.

Return type mixed

See also:

- `find()` (main definition)
- `detect()` (alias)
- `find_where()` (alias)

pydash.api.collections.**foldl**(*collection*, *callback=None*, *accumulator=None*)

Reduces a collection to a value which is the accumulated result of running each element in the collection through the callback, where each successive callback execution consumes the return value of the previous execution.

Parameters

- **collection** (*list|dict*) – Collection to iterate over.
- **callback** (*mixed, optional*) – Callback applied per iteration.
- **accumulator** (*mixed, optional*) – Object that stores result of reduction. Default is to use the result of the first iteration.

Returns Accumulator object containing results of reduction.

Return type mixed

See also:

- `reduce_()` (main definition)
- `foldl()` (alias)
- `inject()` (alias)

pydash.api.collections.**foldr**(*collection*, *callback=None*, *accumulator=None*)

This method is like `reduce_()` except that it iterates over elements of a *collection* from right to left.

Parameters

- **collection** (*list|dict*) – Collection to iterate over.
- **callback** (*mixed, optional*) – Callback applied per iteration.
- **accumulator** (*mixed, optional*) – Object that stores result of reduction. Default is to use the result of the first iteration.

Returns Accumulator object containing results of reduction.

Return type mixed

See also:

- `reduce_right()` (main definition)
- `foldr()` (alias)

pydash.api.collections.**for_each**(*collection*, *callback=None*)

Iterates over elements of a collection, executing the callback for each element.

Parameters

- **collection** (*list|dict*) – Collection to iterate over.
- **callback** (*mixed, optional*) – Callback applied per iteration.

Returns list|dict: *collection*

See also:

- `for_each()` (main definition)
- `each()` (alias)

`pydash.api.collections.`**`for_each_right`**(*collection*, *callback*)

This method is like `for_each()` except that it iterates over elements of a *collection* from right to left.

Parameters

- **collection** (*list|dict*) – Collection to iterate over.
- **callback** (*mixed, optional*) – Callback applied per iteration.

Returns list|dict: *collection*

See also:

- `for_each_right()` (main definition)
- `each_right()` (alias)

`pydash.api.collections.`**`group_by`**(*collection*, *callback=None*)

Creates an object composed of keys generated from the results of running each element of a *collection* through the callback.

Parameters

- **collection** (*list|dict*) – Collection to iterate over.
- **callback** (*mixed, optional*) – Callback applied per iteration.

Returns Results of grouping by *callback*.

Return type dict

`pydash.api.collections.`**`include`**(*collection*, *target*, *from_index=0*)

Checks if a given value is present in a collection. If *from_index* is negative, it is used as the offset from the end of the collection.

Parameters

- **collection** (*list|dict*) – Collection to iterate over.
- **target** (*mixed*) – Target value to compare to.
- **from_index** (*int, optional*) – Offset to start search from.

Returns Whether *target* is in *collection*.

Return type bool

See also:

- `contains()` (main definition)
- `include()` (alias)

pydash.api.collections.**index_by**(*collection*, *callback=None*)

Creates an object composed of keys generated from the results of running each element of the collection through the given callback.

Parameters

- **collection** (*list|dict*) – Collection to iterate over.
- **callback** (*mixed, optional*) – Callback applied per iteration.

Returns Results of indexing by *callback*.

Return type dict

pydash.api.collections.**inject**(*collection*, *callback=None*, *accumulator=None*)

Reduces a collection to a value which is the accumulated result of running each element in the collection through the callback, where each successive callback execution consumes the return value of the previous execution.

Parameters

- **collection** (*list|dict*) – Collection to iterate over.
- **callback** (*mixed, optional*) – Callback applied per iteration.
- **accumulator** (*mixed, optional*) – Object that stores result of reduction. Default is to use the result of the first iteration.

Returns Accumulator object containing results of reduction.

Return type mixed

See also:

- `reduce_()` (main definition)
- `foldl()` (alias)
- `inject()` (alias)

pydash.api.collections.**invoke**(*collection*, *method_name*, **args*, ***kargs*)

Invokes the method named by *method_name* on each element in the *collection* returning a list of the results of each invoked method.

Parameters

- **collection** (*list|dict*) – Collection to iterate over.
- **method_name** (*str*) – Name of method to invoke.

Returns List of results of invoking method of each item.

Return type list

pydash.api.collections.**map_**(*collection*, *callback=None*)

Creates an array of values by running each element in the collection through the callback. The callback is invoked with three arguments: `(value, index|key, collection)`. If a property name is passed for callback, the created `pluck()` style callback will return the property value of the given element. If an object is passed for callback, the created `where()` style callback will return `True` for elements that have the properties of the given object, else `False`.

Parameters

- **collection** (*list|dict*) – Collection to iterate over.
- **callback** (*mixed, optional*) – Callback applied per iteration.

Returns Mapped list.

Return type list

See also:

- `map_()` (main definition)
- `collect()` (alias)

pydash.api.collections.**max_**(*collection*, *callback=None*)

Retrieves the maximum value of a *collection*.

Parameters

- **collection** (*list|dict*) – Collection to iterate over.
- **callback** (*mixed, optional*) – Callback applied per iteration.

Returns Maximum value.

Return type mixed

pydash.api.collections.**min_**(*collection*, *callback=None*)

Retrieves the minimum value of a *collection*.

Parameters

- **collection** (*list|dict*) – Collection to iterate over.
- **callback** (*mixed, optional*) – Callback applied per iteration.

Returns Minimum value.

Return type mixed

pydash.api.collections.**partition**(*collection*, *callback=None*)

Creates an array of elements split into two groups, the first of which contains elements the *callback* returns truthy for, while the second of which contains elements the *callback* returns falsey for. The *callback* is invoked with three arguments: `(value, index|key, collection)`.

If a property name is provided for *callback* the created `pluck()` style callback returns the property value of the given element.

If an object is provided for *callback* the created `where()` style callback returns `True` for elements that have the properties of the given object, else `False`.

Parameters

- **collection** (*list|dict*) – Collection to iterate over.
- **callback** (*mixed, optional*) – Callback applied per iteration.

Returns List of grouped elements.

Return type list

New in version 1.1.0.

pydash.api.collections.**pluck**(*collection*, *key*)

Retrieves the value of a specified property from all elements in the collection.

Parameters

- **collection** (*list|dict*) – list of dicts
- **key** (*str*) – collection's key to pluck

Returns plucked list

Return type list

pydash.api.collections.**reduce_**(*collection*, *callback=None*, *accumulator=None*)

Reduces a collection to a value which is the accumulated result of running each element in the collection through the callback, where each successive callback execution consumes the return value of the previous execution.

Parameters

- **collection** (*list|dict*) – Collection to iterate over.
- **callback** (*mixed, optional*) – Callback applied per iteration.
- **accumulator** (*mixed, optional*) – Object that stores result of reduction. Default is to use the result of the first iteration.

Returns Accumulator object containing results of reduction.

Return type mixed

See also:

- reduce_() (main definition)
- foldl() (alias)
- inject() (alias)

pydash.api.collections.**reduce_right**(*collection*, *callback=None*, *accumulator=None*)

This method is like reduce_() except that it iterates over elements of a *collection* from right to left.

Parameters

- **collection** (*list|dict*) – Collection to iterate over.
- **callback** (*mixed, optional*) – Callback applied per iteration.
- **accumulator** (*mixed, optional*) – Object that stores result of reduction. Default is to use the result of the first iteration.

Returns Accumulator object containing results of reduction.

Return type mixed

See also:

- reduce_right() (main definition)
- foldr() (alias)

pydash.api.collections.**reject**(*collection*, *callback=None*)

The opposite of filter_() this method returns the elements of a collection that the callback does **not** return truthy for.

Parameters

- **collection** (*list|dict*) – Collection to iterate over.
- **callback** (*mixed, optional*) – Callback applied per iteration.

Returns Rejected elements of *collection*.

Return type list

pydash.api.collections.**sample**(*collection*, *n=None*)
Retrieves a random element or *n* random elements from a *collection*.

Parameters

- **collection** (*list|dict*) – Collection to iterate over.
- **n** (*int, optional*) – Number of random samples to return.

Returns list|mixed: List of sampled collection value if *n* is provided, else single value from collection if *n* is `None`.

pydash.api.collections.**select**(*collection*, *callback=None*)
Iterates over elements of a collection, returning an list of all elements the callback returns truthy for.

Parameters

- **collection** (*list|dict*) – Collection to iterate over.
- **callback** (*mixed, optional*) – Callback applied per iteration.

Returns Filtered list.

Return type list

See also:

- `select()` (main definition)
- `filter_()` (alias)

pydash.api.collections.**shuffle**(*collection*)
Creates a list of shuffled values, using a version of the Fisher-Yates shuffle.

Parameters **collection** (*list|dict*) – Collection to iterate over.

Returns Shuffled list of values.

Return type list

pydash.api.collections.**size**(*collection*)
Gets the size of the *collection* by returning *len(collection)* for iterable objects.

Parameters **collection** (*list|dict*) – Collection to iterate over.

Returns Collection length.

Return type int

pydash.api.collections.**some**(*collection*, *callback=None*)
Checks if the callback returns a truthy value for any element of a collection. The callback is invoked with three arguments: `(value, index|key, collection)`. If a property name is passed for callback, the created `pluck()` style callback will return the property value of the given element. If an object is passed for callback, the created `where()` style callback will return `True` for elements that have the properties of the given object, else `False`.

Parameters

- **collection** (*list|dict*) – Collection to iterate over.
- **callbacked** (*mixed, optional*) – Callback applied per iteration.

Returns Whether any of the elements are truthy.

Return type bool

See also:

- some() (main definition)
- any_() (alias)

pydash.api.collections.**sort_by**(*collection*, *callback=None*)

Creates a list of elements, sorted in ascending order by the results of running each element in a *collection* through the callback.

Parameters

- **collection** (*list|dict*) – Collection to iterate over.
- **callback** (*mixed, optional*) – Callback applied per iteration.

Returns Sorted list.

Return type list

pydash.api.collections.**to_list**(*collection*)

Converts the collection to a list.

Parameters **collection** (*list|dict*) – Collection to iterate over.

Returns Collection converted to list.

Return type list

pydash.api.collections.**where**(*collection*, *properties*)

Examines each element in a collection, returning an array of all elements that have the given properties.

Parameters

- **collection** (*list|dict*) – Collection to iterate over.
- **properties** (*dict*) – property values to filter by

Returns filtered list

Return type list

4.1.4 Functions

Functions that wrap other functions.

New in version 1.0.0.

pydash.api.functions.**after**(*n*, *func*)

Creates a function that executes *func*, with the arguments of the created function, only after being called *n* times.

Parameters

- **n** (*int*) – Number of times *func* must be called before it is executed.
- **func** (*function*) – Function to execute.

Returns Function wrapped in an After context.

Return type After

New in version 1.0.0.

pydash.api.functions.**before**(*n*, *func*)

Creates a function that executes *func*, with the arguments of the created function, until it has been called *n* times.

Parameters

- **n** (*int*) – Number of times *func* may be executed.

- **func** (*function*) – Function to execute.

Returns Function wrapped in an `Before` context.

Return type Before

New in version 1.1.0.

`pydash.api.functions.`**`compose`**`(*funcs)`

Creates a function that is the composition of the provided functions, where each function consumes the return value of the function that follows. For example, composing the functions `f()`, `g()`, and `h()` produces `f(g(h()))`.

Returns Function(s) wrapped in a `Compose` context.

Return type Compose

New in version 1.0.0.

`pydash.api.functions.`**`curry`**`(func, arity=None)`

Creates a function which accepts one or more arguments of *func* that when invoked either executes *func* returning its result, if all *func* arguments have been provided, or returns a function that accepts one or more of the remaining *func* arguments, and so on.

Parameters

- **func** (*function*) – Function to curry.
- **arity** (*int, optional*) – Number of function arguments that can be accepted by curried function. Default is to use the number of arguments that are accepted by *func*.

Returns Function wrapped in a `Curry` context.

Return type Curry

New in version 1.0.0.

`pydash.api.functions.`**`curry_right`**`(func, arity=None)`

This method is like `curry()` except that arguments are applied to *func* in the manner of `partial_right()` instead of `partial()`.

Parameters

- **func** (*function*) – Function to curry.
- **arity** (*int, optional*) – Number of function arguments that can be accepted by curried function. Default is to use the number of arguments that are accepted by *func*.

Returns Function wrapped in a `CurryRight` context.

Return type CurryRight

New in version 1.1.0.

`pydash.api.functions.`**`debounce`**`(func, wait, max_wait=False)`

Creates a function that will delay the execution of *func* until after *wait* milliseconds have elapsed since the last time it was invoked. Subsequent calls to the debounced function will return the result of the last *func* call.

Parameters

- **func** (*function*) – Function to execute.
- **wait** (*int*) – Milliseconds to wait before executing *func*.
- **max_wait** (*optional*) – Maximum time to wait before executing *func*.

Returns Function wrapped in a `Debounce` context.

Return type Debounce

New in version 1.0.0.

pydash.api.functions.**delay**(*func*, *wait*, **args*, ***kargs*)

Executes the *func* function after *wait* milliseconds. Additional arguments will be provided to *func* when it is invoked.

Parameters

- **func** (*function*) – Function to execute.
- **wait** (*int*) – Milliseconds to wait before executing *func*.

Returns Return from *func*.

Return type mixed

New in version 1.0.0.

pydash.api.functions.**negate**(*func*)

Creates a function that negates the result of the predicate *func*. The *func* function is executed with the arguments of the created function.

Parameters **func** (*function*) – Function to negate execute.

Returns Function wrapped in a `Negate` context.

Return type Negate

New in version 1.1.0.

pydash.api.functions.**once**(*func*)

Creates a function that is restricted to execute func once. Repeat calls to the function will return the value of the first call.

Parameters **func** (*function*) – Function to execute.

Returns Function wrapped in a `Once` context.

Return type Once

New in version 1.0.0.

pydash.api.functions.**partial**(*func*, **args*)

Creates a function that, when called, invokes *func* with any additional partial arguments prepended to those provided to the new function.

Parameters **func** (*function*) – Function to execute.

Returns Function wrapped in a `Partial` context.

Return type Partial

New in version 1.0.0.

pydash.api.functions.**partial_right**(*func*, **args*)

This method is like `partial()` except that partial arguments are appended to those provided to the new function.

Parameters **func** (*function*) – Function to execute.

Returns Function wrapped in a `Partial` context.

Return type Partial

New in version 1.0.0.

pydash.api.functions.**throttle**(*func*, *wait*)

Creates a function that, when executed, will only call the *func* function at most once per every *wait* milliseconds. Subsequent calls to the throttled function will return the result of the last *func* call.

Parameters

- **func** (*function*) – Function to throttle.
- **wait** (*int*) – Milliseconds to wait before calling *func* again.

Returns Results of last *func* call.

Return type mixed

New in version 1.0.0.

pydash.api.functions.**wrap**(*value*, *func*)

Creates a function that provides value to the wrapper function as its first argument. Additional arguments provided to the function are appended to those provided to the wrapper function.

Parameters

- **value** (*mixed*) – Value provided as first argument to function call.
- **func** (*function*) – Function to execute.

Returns Function wrapped in a `Partial` context.

Return type Partial

New in version 1.0.0.

4.1.5 Objects

Functions that operate on lists, dicts, and other objects.

New in version 1.0.0.

pydash.api.objects.**assign**(*obj*, **sources*, ***kargs*)

Assigns own enumerable properties of source object(s) to the destination object.

Parameters

- **obj** (*dict*) – Destination object whose properties will be modified.
- **callback** (*mixed, optional*) – Callback applied per iteration.

Returns Modified *obj*.

Return type dict

Warning: *obj* is modified in place.

See also:

- `assign()` (main definition)
- `extend()` (alias)

New in version 1.0.0.

pydash.api.objects.**clone**(*value*, *is_deep=False*, *callback=None*)

Creates a clone of *value*. If *is_deep* is `True` nested valueects will also be cloned, otherwise they will be assigned by reference. If a callback is provided it will be executed to produce the cloned values. The callback is invoked with one argument: `(value)`.

Parameters

- **value** (*list|dict*) – Object to clone.
- **is_deep** (*bool, optional*) – Whether to perform deep clone.
- **callback** (*mixed, optional*) – Callback applied per iteration.

Returns list|dict: Cloned object.

New in version 1.0.0.

pydash.api.objects.**clone_deep**(*value*, *callback=None*)

Creates a deep clone of *value*. If a callback is provided it will be executed to produce the cloned values. The callback is invoked with one argument: `(value)`.

Parameters

- **value** (*list|dict*) – Object to clone.
- **callback** (*mixed, optional*) – Callback applied per iteration.

Returns list|dict: Cloned object.

New in version 1.0.0.

pydash.api.objects.**defaults**(*obj*, **sources*)

Assigns own enumerable properties of source object(s) to the destination object for all destination properties that resolve to undefined.

Parameters **obj** (*dict*) – Destination object whose properties will be modified.

Returns Modified *obj*.

Return type dict

Warning: *obj* is modified in place.

New in version 1.0.0.

pydash.api.objects.**extend**(*obj*, **sources*, ***kargs*)

Assigns own enumerable properties of source object(s) to the destination object.

Parameters

- **obj** (*dict*) – Destination object whose properties will be modified.
- **callback** (*mixed, optional*) – Callback applied per iteration.

Returns Modified *obj*.

Return type dict

Warning: *obj* is modified in place.

See also:

- `assign()` (main definition)
- `extend()` (alias)

New in version 1.0.0.

pydash.api.objects.**find_key**(*obj*, *callback=None*)

This method is like pydash.api.arrays.find_index() except that it returns the key of the first element that passes the callback check, instead of the element itself.

Parameters

- **obj** (*list|dict*) – Object to search.
- **callback** (*mixed*) – Callback applied per iteration.

Returns Found key or None.

Return type mixed

See also:

- find_key() (main definition)
- find_last_key() (alias)

New in version 1.0.0.

pydash.api.objects.**find_last_key**(*obj*, *callback=None*)

This method is like pydash.api.arrays.find_index() except that it returns the key of the first element that passes the callback check, instead of the element itself.

Parameters

- **obj** (*list|dict*) – Object to search.
- **callback** (*mixed*) – Callback applied per iteration.

Returns Found key or None.

Return type mixed

See also:

- find_key() (main definition)
- find_last_key() (alias)

New in version 1.0.0.

pydash.api.objects.**for_in**(*obj*, *callback=None*)

Iterates over own and inherited enumerable properties of *obj*, executing *callback* for each property.

Parameters

- **obj** (*list|dict*) – Object to process.
- **callback** (*mixed*) – Callback applied per iteration.

Returns list|dict: *obj*.

See also:

- for_in() (main definition)
- for_own() (alias)

New in version 1.0.0.

pydash.api.objects.**for_in_right**(*obj*, *callback=None*)
This function is like for_in() except it iterates over the properties in reverse order.

Parameters

- **obj** (*list|dict*) – Object to process.
- **callback** (*mixed*) – Callback applied per iteration.

Returns list|dict: *obj*.

See also:

- for_in_right() (main definition)
- for_own_right() (alias)

New in version 1.0.0.

pydash.api.objects.**for_own**(*obj*, *callback=None*)
Iterates over own and inherited enumerable properties of *obj*, executing *callback* for each property.

Parameters

- **obj** (*list|dict*) – Object to process.
- **callback** (*mixed*) – Callback applied per iteration.

Returns list|dict: *obj*.

See also:

- for_in() (main definition)
- for_own() (alias)

New in version 1.0.0.

pydash.api.objects.**for_own_right**(*obj*, *callback=None*)
This function is like for_in() except it iterates over the properties in reverse order.

Parameters

- **obj** (*list|dict*) – Object to process.
- **callback** (*mixed*) – Callback applied per iteration.

Returns list|dict: *obj*.

See also:

- for_in_right() (main definition)
- for_own_right() (alias)

New in version 1.0.0.

pydash.api.objects.**functions**(*obj*)
Creates a list of keys of an object that are callable.

Parameters **obj** (*list|dict*) – Object to inspect.

Returns All keys whose values are callable.

Return type list

See also:

- `functions()` (main definition)
- `methods()` (alias)

New in version 1.0.0.

`pydash.api.objects.has(obj, key)`
Checks if *key* exists as a key of *obj*.

Parameters

- **obj** (*mixed*) – Object to test.
- **key** (*mixed*) – Key to test for.

Returns Whether *obj* has *key*.

Return type bool

New in version 1.0.0.

`pydash.api.objects.invert(obj)`
Creates an object composed of the inverted keys and values of the given object.

Parameters **obj** (*dict*) – dict to invert

Returns Inverted dict

Return type dict

Note: Assumes *dict* values are hashable as *dict* keys.

New in version 1.0.0.

`pydash.api.objects.is_boolean(value)`
Checks if *value* is a boolean value.

Parameters **value** (*mixed*) – Value to check.

Returns Whether *value* is a boolean.

Return type bool

New in version 1.0.0.

`pydash.api.objects.is_date(value)`
Check if *value* is a date object.

Parameters **value** (*mixed*) – Value to check.

Returns Whether *value* is a date object.

Return type bool

Note: This will also return `True` for datetime objects.

New in version 1.0.0.

`pydash.api.objects.is_empty(value)`
Checks if *value* is empty.

Parameters **value** (*mixed*) – Value to check.

Returns Whether *value* is empty.

Return type bool

Note: Returns `True` for booleans and numbers.

New in version 1.0.0.

`pydash.api.objects.`**`is_equal`**(*a*, *b*, *callback=None*)

Performs a comparison between two values to determine if they are equivalent to each other. If a callback is provided it will be executed to compare values. If the callback returns `None`, comparisons will be handled by the method instead. The callback is invoked with two arguments: `(a, b)`.

Parameters

- **a** (*list|dict*) – Object to compare.
- **b** (*list|dict*) – Object to compare.
- **callback** (*mixed, optional*) – Callback used to compare values from *a* and *b*.

Returns Whether *a* and *b* are equal.

Return type bool

New in version 1.0.0.

`pydash.api.objects.`**`is_error`**(*value*)

Checks if *value* is an `Exception`.

Parameters **value** (*mixed*) – Value to check.

Returns Whether *value* is an exception.

Return type bool

New in version 1.1.0.

`pydash.api.objects.`**`is_function`**(*value*)

Checks if *value* is a function.

Parameters **value** (*mixed*) – Value to check.

Returns Whether *value* is callable.

Return type bool

New in version 1.0.0.

`pydash.api.objects.`**`is_list`**(*value*)

Checks if *value* is a list.

Parameters **value** (*mixed*) – Value to check.

Returns Whether *value* is a list.

Return type bool

New in version 1.0.0.

`pydash.api.objects.`**`is_nan`**(*value*)

Checks if *value* is not a number.

Parameters **value** (*mixed*) – Value to check.

Returns Whether *value* is not a number.

Return type bool

New in version 1.0.0.

pydash.api.objects.**is_none**(*value*)
Checks if *value* is *None*.

Parameters **value** (*mixed*) – Value to check.

Returns Whether *value* is `None`.

Return type bool

New in version 1.0.0.

pydash.api.objects.**is_number**(*value*)
Checks if *value* is a number.

Parameters **value** (*mixed*) – Value to check.

Returns Whether *value* is a number.

Return type bool

Note: Returns `True` for `int`, `long` (PY2), `float`, and `decimal.Decimal`.

New in version 1.0.0.

pydash.api.objects.**is_object**(*value*)
Checks if *value* is a `list` or `dict`.

Parameters **value** (*mixed*) – Value to check.

Returns Whether *value* is `list` or `dict`.

Return type bool

New in version 1.0.0.

pydash.api.objects.**is_plain_object**(*value*)
Checks if *value* is a `dict`.

Parameters **value** (*mixed*) – Value to check.

Returns Whether *value* is a `dict`.

Return type bool

New in version 1.0.0.

pydash.api.objects.**is_re**(*value*)
Checks if *value* is a `RegExp` object.

Parameters **value** (*mxied*) – Value to check.

Returns Whether *value* is a RegExp object.

Return type bool

See also:

- `is_reg_exp()` (main definition)
- `is_re()` (alias)

New in version 1.1.0.

pydash.api.objects.**is_reg_exp**(*value*)
Checks if *value* is a `RegExp` object.

Parameters **value** (*mxied*) – Value to check.

Returns Whether *value* is a RegExp object.

Return type bool

See also:

- `is_reg_exp()` (main definition)
- `is_re()` (alias)

New in version 1.1.0.

`pydash.api.objects.`**`is_string`**`(value)`
Checks if *value* is a string.

Parameters **value** (*mixed*) – Value to check.

Returns Whether *value* is a string.

Return type bool

New in version 1.0.0.

`pydash.api.objects.`**`keys`**`(obj)`
Creates a list composed of the keys of *obj*.

Parameters **obj** (*mixed*) – Object to extract keys from.

Returns List of keys.

Return type list

See also:

- `keys()` (main definition)
- `keys_in()` (alias)

New in version 1.0.0.

Changed in version 1.1.0: Added `keys_in()` as alias.

`pydash.api.objects.`**`keys_in`**`(obj)`
Creates a list composed of the keys of *obj*.

Parameters **obj** (*mixed*) – Object to extract keys from.

Returns List of keys.

Return type list

See also:

- `keys()` (main definition)
- `keys_in()` (alias)

New in version 1.0.0.

Changed in version 1.1.0: Added `keys_in()` as alias.

`pydash.api.objects.`**`map_values`**`(obj, callback=None)`
Creates an object with the same keys as *obj* and values generated by running each property of *obj* through the *callback*. The callback is invoked with three arguments: `(value, key, object)`. If a property name is provided for *callback* the created `pydash.api.collections.pluck()` style callback

will return the property value of the given element. If an object is provided for callback the created pydash.api.collections.where() style callback will return True for elements that have the properties of the given object, else False.

Parameters

- **obj** (*list|dict*) – Object to map.
- **callback** (*mixed*) – Callback applied per iteration.

Returns list|dict: Results of running *obj* through *callback*.

New in version 1.0.0.

pydash.api.objects.**merge**(*obj*, **sources*, ***kargs*)

Recursively merges own enumerable properties of the source object(s) that don't resolve to undefined into the destination object. Subsequent sources will overwrite property assignments of previous sources. If a callback is provided it will be executed to produce the merged values of the destination and source properties. If the callback returns undefined merging will be handled by the method instead. The callback is invoked with two arguments: (obj_value, source_value).

Parameters **obj** – Destination object to merge source(s) into.

Keyword Arguments **callback** (*function, optional*) – Callback function to handle merging (must be passed in as keyword argument).

Returns Merged object.

Return type dict

Warning: *obj* is modified in place.

New in version 1.0.0.

pydash.api.objects.**methods**(*obj*)

Creates a list of keys of an object that are callable.

Parameters **obj** (*list|dict*) – Object to inspect.

Returns All keys whose values are callable.

Return type list

See also:

- functions() (main definition)
- methods() (alias)

New in version 1.0.0.

pydash.api.objects.**omit**(*obj*, *callback=None*, **properties*)

Creates a shallow clone of object excluding the specified properties. Property names may be specified as individual arguments or as lists of property names. If a callback is provided it will be executed for each property of object omitting the properties the callback returns truthy for. The callback is invoked with three arguments: (value, key, object).

Parameters

- **obj** (*mixed*) – Object to process.
- **callback** (*mixed, optional*) – Callback used to determine whic properties to omit.

Returns Results of omitting properties.

Return type dict

New in version 1.0.0.

pydash.api.objects.**pairs**(*obj*)

Creates a two dimensional list of an object's key-value pairs, i.e. `[[key1, value1], [key2, value2]]`.

Parameters **obj** (*mixed*) – Object to process.

Returns Two dimensional list of object's key-value pairs.

Return type list

New in version 1.0.0.

pydash.api.objects.**parse_int**(*value*, *radix=None*)

Converts the given *value* into an integer of the specified *radix*. If *radix* is falsey a radix of `10` is used unless the *value* is a hexadecimal, in which case a radix of 16 is used.

Parameters

- **value** (*mixed*) – Value to parse.
- **radix** (*int, optional*) – Base to convert to.

Returns Integer if parsable else `None`.

Return type mixed

New in version 1.0.0.

pydash.api.objects.**pick**(*obj*, *callback=None*, **properties*)

Creates a shallow clone of object composed of the specified properties. Property names may be specified as individual arguments or as lists of property names. If a callback is provided it will be executed for each property of object picking the properties the callback returns truthy for. The callback is invoked with three arguments: `(value, key, object)`.

Parameters

- **obj** (*list|dict*) – Object to pick from.
- **callback** (*mixed, optional*) – Callback used to determine whic properties to pick.

Returns Results of picking properties.

Return type dict

New in version 1.0.0.

pydash.api.objects.**transform**(*obj*, *callback=None*, *accumulator=None*)

An alternative to `pydash.api.collections.reduce()`, this method transforms *obj* to a new accumulator object which is the result of running each of its properties through a callback, with each callback execution potentially mutating the accumulator object. The callback is invoked with four arguments: `(accumulator, value, key, object)`. Callbacks may exit iteration early by explicitly returning `False`.

Parameters

- **obj** (*list|dict*) – Object to process.
- **callback** (*mixed*) – Callback applied per iteration.
- **accumulator** (*mixed, optional*) – Accumulated object. Defaults to `list`.

Returns Accumulated object.

Return type mixed

New in version 1.0.0.

pydash.api.objects.**update**(*obj*, *source*, *callback=None*)

Update properties of *obj* with *source*. If a callback is provided, it will be executed to produce the updated values of the destination and source properties. The callback is invoked with two arguments: `(obj_value, source_value)`.

Parameters

- **obj** (*dict*) – destination object to merge source(s) into
- **source** (*dict*) – source object to merge from
- **callback** (*function, optional*) – callback function to handle merging

Returns merged object

Return type mixed

Warning: *obj* is modified in place.

New in version 1.0.0.

pydash.api.objects.**values**(*obj*)

Creates a list composed of the values of *obj*.

Parameters **obj** (*mixed*) – Object to extract values from.

Returns List of values.

Return type list

See also:

- `values()` (main definition)
- `values_in()` (alias)

New in version 1.0.0.

Changed in version 1.1.0: Added `values_in()` as alias.

pydash.api.objects.**values_in**(*obj*)

Creates a list composed of the values of *obj*.

Parameters **obj** (*mixed*) – Object to extract values from.

Returns List of values.

Return type list

See also:

- `values()` (main definition)
- `values_in()` (alias)

New in version 1.0.0.

Changed in version 1.1.0: Added `values_in()` as alias.

4.1.6 Strings

String functions.

New in version 1.1.0.

pydash.api.strings.**camel_case**(*text*)
Converts *text* to camel case.

Parameters **text** (*str*) – String to convert.

Returns String converted to camel case.

Return type str

New in version 1.1.0.

pydash.api.strings.**capitalize**(*text*)
Capitalizes the first character of *text*.

Parameters **text** (*str*) – String to capitalize.

Returns Capitalized string.

Return type str

New in version 1.1.0.

pydash.api.strings.**ends_with**(*text*, *target*, *position=None*)
Checks if *text* ends with a given target string.

Parameters

- **text** (*str*) – String to check.
- **target** (*str*) – String to check for.
- **position** (*int, optional*) – Position to search from. Defaults to end of *text*.

Returns Whether *text* ends with *target*.

Return type bool

New in version 1.1.0.

pydash.api.strings.**escape**(*text*)
Converts the characters `&`, `<`, `>`, `"`, `'`, and `` ` `` in *text* to their corresponding HTML entities.

Parameters **text** (*str*) – String to escape.

Returns HTML escaped string.

Return type str

New in version 1.0.0.

Changed in version 1.1.0: Moved function to Strings module.

pydash.api.strings.**escape_reg_exp**(*text*)
Escapes the RegExp special characters in *text*.

Parameters **text** (*str*) – String to escape.

Returns RegExp escaped string.

Return type str

New in version 1.1.0.

pydash.api.strings.**escape_re**(*text*)
Escapes the RegExp special characters in *text*.

Parameters **text** (*str*) – String to escape.

Returns RegExp escaped string.

Return type str

New in version 1.1.0.

pydash.api.strings.**kebab_case**(*text*)
Converts *text* to kebab case (a.k.a. spinal case).

Parameters **text** (*str*) – String to convert.

Returns String converted to kebab case.

Return type str

New in version 1.1.0.

pydash.api.strings.**pad**(*text*, *length*, *chars=' '*)
Pads *text* on the left and right sides if it is shorter than the given padding length. The *chars* string may be truncated if the number of padding characters can't be evenly divided by the padding length.

Parameters

- **text** (*str*) – String to pad.
- **length** (*int*) – Amount to pad.
- **chars** (*str, optional*) – Chars to pad with. Defaults to `" "`.

Returns Padded string.

Return type str

New in version 1.1.0.

pydash.api.strings.**pad_left**(*text*, *length*, *chars=' '*)
Pads *text* on the left side if it is shorter than the given padding length. The *chars* string may be truncated if the number of padding characters can't be evenly divided by the padding length.

Parameters

- **text** (*str*) – String to pad.
- **length** (*int*) – Amount to pad.
- **chars** (*str, optional*) – Chars to pad with. Defaults to `" "`.

Returns Padded string.

Return type str

New in version 1.1.0.

pydash.api.strings.**pad_right**(*text*, *length*, *chars=' '*)
Pads *text* on the right side if it is shorter than the given padding length. The *chars* string may be truncated if the number of padding characters can't be evenly divided by the padding length.

Parameters

- **text** (*str*) – String to pad.
- **length** (*int*) – Amount to pad.
- **chars** (*str, optional*) – Chars to pad with. Defaults to `" "`.

Returns Padded string.

Return type str

New in version 1.1.0.

pydash.api.strings.**repeat**(*text*, *n=0*)
Repeats the given string *n* times.

Parameters

- **text** (*str*) – String to repeat.
- **n** (*int, optional*) – Number of times to repeat the string.

Returns Repeated string.

Return type str

New in version 1.1.0.

pydash.api.strings.**snake_case**(*text*)
Converts *text* to snake case.

Parameters **text** (*str*) – String to convert.

Returns String converted to snake case.

Return type str

New in version 1.1.0.

pydash.api.strings.**starts_with**(*text*, *target*, *position=None*)
Checks if *text* starts with a given target string.

Parameters

- **text** (*str*) – String to check.
- **target** (*str*) – String to check for.
- **position** (*int, optional*) – Position to search from. Defaults to beginning of *text*.

Returns Whether *text* starts with *target*.

Return type bool

New in version 1.1.0.

pydash.api.strings.**trim**(*text*, *chars=None*)
Removes leading and trailing whitespace or specified characters from *text*.

Parameters

- **text** (*str*) – String to trim.
- **chars** (*str, optional*) – Specific characters to remove.

Returns Trimmed string.

Return type str

New in version 1.1.0.

pydash.api.strings.**trim_left**(*text*, *chars=None*)
Removes leading whitespace or specified characters from *text*.

Parameters

- **text** (*str*) – String to trim.

- **chars** (*str, optional*) – Specific characters to remove.

Returns Trimmed string.

Return type str

New in version 1.1.0.

pydash.api.strings.**trim_right**(*text*, *chars=None*)
Removes trailing whitespace or specified characters from *text*.

Parameters

- **text** (*str*) – String to trim.
- **chars** (*str, optional*) – Specific characters to remove.

Returns Trimmed string.

Return type str

New in version 1.1.0.

pydash.api.strings.**trunc**(*text*, *length=30*, *omission='...'*, *separator=None*)
Truncates *text* if it is longer than the given maximum string length. The last characters of the truncated string are replaced with the omission string which defaults to `...`.

Parameters

- **text** (*str*) – String to truncate.
- **length** (*int, optional*) – Maximum string length. Defaults to `30`.
- **omission** (*str, optional*) – String to indicate text is omitted.
- **separator** (*mixed, optional*) – Separator pattern to truncate to.

Returns Truncated string.

Return type str

New in version 1.1.0.

pydash.api.strings.**unescape**(*text*)
The inverse of `escape()`. This method converts the HTML entities `&`, `<`, `>`, `&quot;`, `&#39;`, and ``` in *text* to their corresponding characters.

Parameters **text** (*str*) – String to unescape.

Returns HTML unescaped string.

Return type str

New in version 1.0.0.

Changed in version 1.1.0: Moved to Strings module.

4.1.7 Utilities

Utility functions.

New in version 1.0.0.

pydash.api.utilities.**attempt**(*func*, **args*, ***kargs*)
Attempts to execute *func*, returning either the result or the caught error object.

Parameters **func** (*function*) – The function to attempt.

Returns Returns the *func* result or error object.

Return type mixed

New in version 1.1.0.

pydash.api.utilities.**constant**(*value*)
Creates a function that returns *value*.

Parameters **value** (*mixed*) – Constant value to return.

Returns Function that always returns *value*.

Return type function

New in version 1.0.0.

pydash.api.utilities.**callback**(*func*)
Return a callback. If *func* is a property name the created callback will return the property value for a given element. If *func* is an object the created callback will return `True` for elements that contain the equivalent object properties, otherwise it will return `False`.

Parameters **func** (*mixed*) – Object to create callback function from.

Returns Callback function.

Return type function

See also:

- `callback()` (main definition)
- `create_callback()` (alias)

New in version 1.0.0.

pydash.api.utilities.**create_callback**(*func*)
Return a callback. If *func* is a property name the created callback will return the property value for a given element. If *func* is an object the created callback will return `True` for elements that contain the equivalent object properties, otherwise it will return `False`.

Parameters **func** (*mixed*) – Object to create callback function from.

Returns Callback function.

Return type function

See also:

- `callback()` (main definition)
- `create_callback()` (alias)

New in version 1.0.0.

pydash.api.utilities.**identity**(**args*)
Return the first argument provided to it.

Returns First argument or `None`.

Return type mixed

New in version 1.0.0.

pydash.api.utilities.**matches**(*source*)

Creates a pydash.api.collections.where() style predicate function which performs a deep comparison between a given object and the *source* object, returning True if the given object has equivalent property values, else False.

Parameters **source** (*dict*) – Source object used for comparision.

Returns Function that compares a dict to *source* and returns whether the two objects contain the same items.

Return type function

New in version 1.0.0.

pydash.api.utilities.**memoize**(*func*, *resolver=None*)

Creates a function that memoizes the result of *func*. If *resolver* is provided it will be used to determine the cache key for storing the result based on the arguments provided to the memoized function. By default, all arguments provided to the memoized function are used as the cache key. The result cache is exposed as the cache property on the memoized function.

Parameters

- **func** (*function*) – Function to memoize.
- **resolver** (*function, optional*) – Function that returns the cache key to use.

Returns Memoized function.

Return type function

New in version 1.0.0.

pydash.api.utilities.**noop**(**args*, ***kargs*)

A no-operation function.

New in version 1.0.0.

pydash.api.utilities.**now**()

Return the number of milliseconds that have elapsed since the Unix epoch (1 January 1970 00:00:00 UTC).

Returns Milliseconds since Unix epoch.

Return type int

New in version 1.0.0.

pydash.api.utilities.**property_**(*key*)

Creates a pydash.collections.pluck() style function, which returns the key value of a given object.

Parameters **key** (*mixed*) – Key value to fetch from object.

Returns Function that returns object's key value.

Return type function

See also:

- property_() (main definition)
- prop() (alias)

New in version 1.0.0.

pydash.api.utilities.**prop**(*key*)

Creates a pydash.collections.pluck() style function, which returns the key value of a given object.

Parameters **key** (*mixed*) – Key value to fetch from object.

Returns Function that returns object's key value.

Return type function

See also:

- `property_()` (main definition)
- `prop()` (alias)

New in version 1.0.0.

`pydash.api.utilities.`**`random`**(*start=0*, *stop=1*, *floating=False*)

Produces a random number between *start* and *stop* (inclusive). If only one argument is provided a number between 0 and the given number will be returned. If floating is truthy or either *start* or *stop* are floats a floating-point number will be returned instead of an integer.

Parameters

- **start** (*int*) – Minimum value.
- **stop** (*int*) – Maximum value.
- **floating** (*bool, optional*) – Whether to force random value to `float`. Default is `False`.

Returns int|float: Random value.

New in version 1.0.0.

`pydash.api.utilities.`**`range_`**(**args*)

Creates a list of numbers (positive and/or negative) progressing from start up to but not including end. If start is less than stop a zero-length range is created unless a negative step is specified.

Parameters

- **stop** (*int*) – Integer - 1 to stop at. Defaults to `1`.
- **start** (*int, optional*) – Integer to start with. Defaults to `0`.
- **step** (*int, optional*) – If positive the last element is the largest `start + i * step` less than *stop*. If negative the last element is the smallest `start + i * step` greater than *stop*. Defaults to `1`.

Returns List of integers in range

Return type list

New in version 1.0.0.

Changed in version 1.1.0: Moved to Utilities module.

`pydash.api.utilities.`**`result`**(*obj*, *key*)

Return the value of property *key* on *obj*. If *key* value is a function it will be invoked and its result returned, else the property value is returned. If *obj* is falsey then `None` is returned.

Parameters

- **obj** (*list|dict*) – Object to retrieve result from.
- **key** (*mixed*) – Key or index to get result from.

Returns Result of `obj[key]` or `None`.

Return type mixed

New in version 1.0.0.

pydash.api.utilities.**times**(*n*, *callback*)

Executes the callback *n* times, returning a list of the results of each callback execution. The callback is invoked with one argument: `(index)`.

Parameters

- **n** (*int*) – Number of times to execute *callback*.
- **callback** (*function*) – Function to execute.

Returns A list of results from calling *callback*.

Return type list

New in version 1.0.0.

pydash.api.utilities.**unique_id**(*prefix=None*)

Generates a unique ID. If *prefix* is provided the ID will be appended to it.

Parameters **prefix** (*str, optional*) – String prefix to prepend to ID value.

Returns ID value.

Return type str

New in version 1.0.0.

CHAPTER 5

Project Info

5.1 License

5.2 Changelog

5.2.1 v1.1.0 (2014-08-19)

- Add `attempt()`.
- Add `before()`.
- Add `camel_case()`.
- Add `capitalize()`.
- Add `chunk()`.
- Add `curry_right()`.
- Add `drop_right()`.
- Add `drop_right_while()`.
- Add `drop_while()`.
- Add `ends_with()`.

- Add `escape_reg_exp()` and `escape_re()`.
- Add `is_error()`.
- Add `is_reg_exp()` and `is_re()`.
- Add `kebab_case()`.
- Add `keys_in()` as alias of `keys()`.
- Add `negate()`.
- Add `pad()`.
- Add `pad_left()`.
- Add `pad_right()`.
- Add `partition()`.
- Add `pull_at()`.
- Add `repeat()`.
- Add `slice_()`.
- Add `snake_case()`.
- Add `sorted_last_index()`.
- Add `starts_with()`.
- Add `take_right()`.
- Add `take_right_while()`.
- Add `take_while()`.
- Add `trim()`.
- Add `trim_left()`.
- Add `trim_right()`.
- Add `trunc()`.
- Add `values_in()` as alias of `values()`.
- Create `pydash.api.strings` module.
- Deprecate `tail()`.
- Modify `drop()` to accept `n` argument and remove as alias of `rest()`.
- Modify `take()` to accept `n` argument and remove as alias of `first()`.
- Move `escape()` and `unescape()` from `pydash.api.utilities` to `pydash.api.strings`. **possible breaking change**
- Move `range_()` from `pydash.api.arrays` to `pydash.api.utilities`. **possible breaking change**

5.2.2 v1.0.0 (2014-08-05)

- Add Python 2.6 and Python 3 support.
- Add `after()`.
- Add `assign()` and `extend()`. Thanks nathancahill!

- Add `callback()` and `create_callback()`.
- Add `chain()`.
- Add `clone()`.
- Add `clone_deep()`.
- Add `compose()`.
- Add `constant()`.
- Add `count_by()`. Thanks nathancahill!
- Add `curry()`.
- Add `debounce()`.
- Add `defaults()`. Thanks nathancahill!
- Add `delay()`.
- Add `escape()`.
- Add `find_key()`. Thanks nathancahill!
- Add `find_last()`. Thanks nathancahill!
- Add `find_last_index()`. Thanks nathancahill!
- Add `find_last_key()`. Thanks nathancahill!
- Add `for_each()`. Thanks nathancahill!
- Add `for_each_right()`. Thanks nathancahill!
- Add `for_in()`. Thanks nathancahill!
- Add `for_in_right()`. Thanks nathancahill!
- Add `for_own()`. Thanks nathancahill!
- Add `for_own_right()`. Thanks nathancahill!
- Add `functions_()` and `methods()`. Thanks nathancahill!
- Add `group_by()`. Thanks nathancahill!
- Add `has()`. Thanks nathancahill!
- Add `index_by()`. Thanks nathancahill!
- Add `identity()`.
- Add `inject()`.
- Add `invert()`.
- Add `invoke()`. Thanks nathancahill!
- Add `is_list()`. Thanks nathancahill!
- Add `is_boolean()`. Thanks nathancahill!
- Add `is_empty()`. Thanks nathancahill!
- Add `is_equal()`.
- Add `is_function()`. Thanks nathancahill!
- Add `is_none()`. Thanks nathancahill!

- Add `is_number()`. Thanks nathancahill!
- Add `is_object()`.
- Add `is_plain_object()`.
- Add `is_string()`. Thanks nathancahill!
- Add `keys()`.
- Add `map_values()`.
- Add `matches()`.
- Add `max_()`. Thanks nathancahill!
- Add `memoize()`.
- Add `merge()`.
- Add `min_()`. Thanks nathancahill!
- Add `noop()`.
- Add `now()`.
- Add `omit()`.
- Add `once()`.
- Add `pairs()`.
- Add `parse_int()`.
- Add `partial()`.
- Add `partial_right()`.
- Add `pick()`.
- Add `property_()` and `prop()`.
- Add `pull()`. Thanks nathancahill!
- Add `random()`.
- Add `reduce_()` and `foldl()`.
- Add `reduce_right()` and `foldr()`.
- Add `reject()`. Thanks nathancahill!
- Add `remove()`.
- Add `result()`.
- Add `sample()`.
- Add `shuffle()`.
- Add `size()`.
- Add `sort_by()`. Thanks nathancahill!
- Add `tap()`.
- Add `throttle()`.
- Add `times()`.
- Add `transform()`.

- Add `to_list()`. Thanks nathancahill!
- Add `unescape()`.
- Add `unique_id()`.
- Add `values()`.
- Add `wrap()`.
- Add `xor()`.

5.2.3 v0.0.0 (2014-07-22)

- Add `all_()`.
- Add `any_()`.
- Add `at()`.
- Add `bisect_left()`.
- Add `collect()`.
- Add `collections()`.
- Add `compact()`.
- Add `contains()`.
- Add `detect()`.
- Add `difference()`.
- Add `drop()`.
- Add `each()`.
- Add `each_right()`.
- Add `every()`.
- Add `filter_()`.
- Add `find()`.
- Add `find_index()`.
- Add `find_where()`.
- Add `first()`.
- Add `flatten()`.
- Add `head()`.
- Add `include()`.
- Add `index_of()`.
- Add `initial()`.
- Add `intersection()`.
- Add `last()`.
- Add `last_index_of()`.
- Add `map_()`.

- Add `object_()`.
- Add `pluck()`.
- Add `range_()`.
- Add `rest()`.
- Add `select()`.
- Add `some()`.
- Add `sorted_index()`.
- Add `tail()`.
- Add `take()`.
- Add `union()`.
- Add `uniq()`.
- Add `unique()`.
- Add `unzip()`.
- Add `where()`.
- Add `without()`.
- Add `zip_()`.
- Add `zip_object()`.

5.3 Authors

5.3.1 Lead

- Derrick Gilland, dgilland@gmail.com, dgilland@github

5.3.2 Contributors

- Nathan Cahill, nathan@nathancahill.com, nathancahill@github

5.4 How to Contribute

- Overview
- Guidelines
- Branching
- Continuous Integration
- Project CLI

5.4.1 Overview

1. Fork the repo.
2. Build development environment run tests to ensure a clean, working slate.
3. Improve/fix the code.
4. Add test cases if new functionality introduced or bug fixed (100% test coverage).
5. Ensure tests pass.
6. Add yourself to `AUTHORS.rst`.
7. Push to your fork and submit a pull request to the `develop` branch.

5.4.2 Guidelines

Some simple guidelines to follow when contributing code:

- Adhere to PEP8.
- Clean, well documented code.
- All tests must pass.
- 100% test coverage.

5.4.3 Branching

There are two main development branches: `master` and `develop`. `master` represents the currently released version while `develop` is the latest development work. When submitting a pull request, be sure to submit to `develop`. The originating branch you submit from can be any name though.

5.4.4 Continuous Integration

Integration testing is provided by Travis-CI at https://travis-ci.org/dgilland/pydash.

Test coverage reporting is provided by Coveralls at https://coveralls.io/r/dgilland/pydash.

5.4.5 Project CLI

Some useful CLI commands when working on the project are below. **NOTE:** All commands are run from the root of the project and require `make`.

make build

Run the `clean` and `install` commands.

```
make build
```

make install

Install Python dependencies into virtualenv located at `env/`.

```
make install
```

make clean

Remove build/test related temporary files like `env/`, `.tox`, `.coverage`, and `__pycache__`.

```
make clean
```

make test

Run unittests under the virtualenv's default Python version. Does not test all support Python versions. To test all supported versions, see make test-full.

```
make test
```

make test-full

Run unittest and linting for all supported Python versions. **NOTE:** This will fail if you do not have all Python versions installed on your system. If you are on an Ubuntu based system, the Dead Snakes PPA is a good resource for easily installing multiple Python versions. If for whatever reason you're unable to have all Python versions on your development machine, note that Travis-CI will run full integration tests on all pull requests.

```
make test-full
```

make lint

Run `make pylint` and `make pep8` commands.

```
make lint
```

make pylint

Run `pylint` compliance check on code base.

```
make pylint
```

make pep8

Run PEP8 compliance check on code base.

```
make pep8
```

make docs

Build documentation to `docs/_build/`.

```
make docs
```

5.5 Kudos

Thank you to Lo-Dash for providing such a great library to port.

CHAPTER 6

Indices and Tables

- *genindex*
- *modindex*
- *search*

p