hiker

Aug 28, 2020

Contents:

CHAPTER 1

hiker package

1.1 Submodules

1.2 hiker.hiker module

Some Utility functions, that make yur life easier but don't fit in any better catorgory than util.

exception hiker.hiker.**KeyNotFoundError**(*cause*, *keys=None*, *visited=None*)
Bases: Exception

hiker.hiker.**contains_key**(*nested_thing*, *key*, *splitval='/'*, *expand=True*)
Tests if the path like key can find an object in the nested_thing.

hiker.hiker.**get_leaf_names**(*nested_thing*)

hiker.hiker.**pop_keypath**(*current_item: Union[callable, list, dict], key: str, splitval: str = '/', default: object = None, expand: bool = True, pass_success: bool = False*)
Given a nested list or dict structure, pop the desired value at key expanding callable nodes if necessary and `expand` is `True`. The expansion is done in-place.

Parameters

- **current_item** (*list or dict*) – Possibly nested list or dictionary.
- **key** (*str*) – key/to/value, path like string describing all keys necessary to consider to get to the desired value. List indices can also be passed here.
- **splitval** (*str*) – String that defines the delimiter between keys of the different depth levels in *key*.
- **default** (*obj*) – Value returned if `key` is not found.
- **expand** (*bool*) – Whether to expand callable nodes on the path or not.

Returns

- The desired value or if `default` is not `None` and the

- `key` is not found returns `default`.

Raises

- Exception if `key` not in `list_or_dict` and `default` is
- `None`.

`hiker.hiker.`**`retrieve`**(*list_or_dict*, *key*, *splitval='/'*, *default=None*, *expand=True*, *pass_success=False*)

Given a nested list or dict return the desired value at key expanding callable nodes if necessary and `expand` is `True`. The expansion is done in-place.

Parameters

- **`list_or_dict`** (*`list or dict`*) – Possibly nested list or dictionary.
- **`key`** (*`str`*) – key/to/value, path like string describing all keys necessary to consider to get to the desired value. List indices can also be passed here.
- **`splitval`** (*`str`*) – String that defines the delimiter between keys of the different depth levels in *key*.
- **`default`** (*`obj`*) – Value returned if `key` is not found.
- **`expand`** (*`bool`*) – Whether to expand callable nodes on the path or not.

Returns

- The desired value or if `default` is not `None` and the
- `key` is not found returns `default`.

Raises

- Exception if `key` not in `list_or_dict` and `default` is
- `None`.

`hiker.hiker.`**`set_default`**(*list_or_dict*, *key*, *default*, *splitval='/'*)

Combines *`retrieve()`* and *`set_value()`* to create the behaviour of pythons `dict.setdefault`: If `key` is found in `list_or_dict`, return its value, otherwise return `default` and add it to `list_or_dict` at `key`.

Parameters

- **`list_or_dict`** (*`list or dict`*) – Possibly nested list or dictionary. splitval (str): String that defines the delimiter between keys of the different depth levels in *key*.
- **`key`** (*`str`*) – key/to/value, path like string describing all keys necessary to consider to get to the desired value. List indices can also be passed here.
- **`default`** (*`object`*) – Value to be returned if `key` not in list_or_dict and set to be at `key` in this case.
- **`splitval`** (*`str`*) – String that defines the delimiter between keys of the different depth levels in `key`.

Returns

- The retrieved value or if the `key` is not found returns
- `default`.

`hiker.hiker.`**`set_value`**(*list_or_dict*, *key*, *val*, *splitval='/'*)

Sets a value in a possibly nested list or dict object.

Parameters

- **key** (*str*) – key/to/value, path like string describing all keys necessary to consider to get to the desired value. List indices can also be passed here.
- **value** (*object*) – Anything you want to put behind key
- **list_or_dict** (*list or dict*) – Possibly nested list or dictionary.
- **splitval** (*str*) – String that defines the delimiter between keys of the different depth levels in key.

Examples

```
dol = {"a": [1, 2], "b": {"c": {"d": 1}, "e": 2}}

# Change existing entry
set_value(dol, "a/0", 3)
# {'a': [3, 2], 'b': {'c': {'d': 1}, 'e': 2}}

set_value(dol, "b/e", 3)
# {"a": [3, 2], "b": {"c": {"d": 1}, "e": 3}}

set_value(dol, "a/1/f", 3)
# {"a": [3, {"f": 3}], "b": {"c": {"d": 1}, "e": 3}}

# Append to list
dol = {"a": [1, 2], "b": {"c": {"d": 1}, "e": 2}}

set_value(dol, "a/2", 3)
# {"a": [1, 2, 3], "b": {"c": {"d": 1}, "e": 2}}

set_value(dol, "a/5", 6)
# {"a": [1, 2, 3, None, None, 6], "b": {"c": {"d": 1}, "e": 2}}

# Add key
dol = {"a": [1, 2], "b": {"c": {"d": 1}, "e": 2}}
set_value(dol, "f", 3)
# {"a": [1, 2], "b": {"c": {"d": 1}, "e": 2}, "f": 3}

set_value(dol, "b/1", 3)
# {"a": [1, 2], "b": {"c": {"d": 1}, "e": 2, 1: 3}, "f": 3}

# Raises Error:
# Appending key to list
# set_value(dol, 'a/g', 3)  # should raise

# Fancy Overwriting
dol = {"a": [1, 2], "b": {"c": {"d": 1}}, "e": 2}

set_value(dol, "e/f", 3)
# {"a": [1, 2], "b": {"c": {"d": 1}}, "e": {"f": 3}}

set_value(dol, "e/f/1/g", 3)
# {"a": [1, 2], "b": {"c": {"d": 1}}, "e": {"f": [None, {"g": 3}]}}

# Toplevel new key
dol = {"a": [1, 2], "b": {"c": {"d": 1}}, "e": 2}
set_value(dol, "h", 4)
```

(continues on next page)

(continued from previous page)

```
# {"a": [1, 2], "b": {"c": {"d": 1}}, "e": 2, "h": 4}

set_value(dol, "i/j/k", 4)
# {"a": [1, 2], "b": {"c": {"d": 1}}, "e": 2, "h": 4, "i": {"j": {"k": 4}}}

set_value(dol, "j/0/k", 4)
# {"a": [1, 2], "b": {"c": {"d": 1}}, "e": 2, "h": 4, "i": {"j": {"k": 4}}, "j": [
↪{"k": 4}], }

# Toplevel is list new key
dol = [{"a": [1, 2], "b": {"c": {"d": 1}}, "e": 2}, 2, 3]

set_value(dol, "0/k", 4)
# [{"a": [1, 2], "b": {"c": {"d": 1}}, "e": 2, "k": 4}, 2, 3]

set_value(dol, "0", 1)
# [1, 2, 3]
```

hiker.hiker.**strenumerate**(*iterable*)

Works just as enumerate, but the returned index is a string.

Parameters **iterable** (*Iterable*) – An (guess what) iterable object.

hiker.hiker.**update**(*to_update*, *to_update_with*, *splitval='/'*, *mode='lax'*)

Updates the entries in a nested object given another nested object.

Parameters

- **to_update** (*dict or list*) – The object that will be manipulated
- **to_update_with** (*dict or list*) – The object used to manipulate `to_update`.
- **splitval** (*str*) – Path element seperator.
- **mode** (*str*) – One of `['lax', 'medium', 'strict']`. Determines how the update is executed:

  lax Any given key in `to_update_with` will be created in `to_update`.

  medium Any key in `to_update_with` that does not exist in `to_update` will simply be ignored.

  strict The first key in `to_update_with` that does not exist in `to_update` will raise a *KeyNotFoundError*.

Raises *KeyNotFoundError* – If a key in `to_update_with` cannot be found in `to_update`.

hiker.hiker.**walk**(*dict_or_list*, *fn*, *inplace=False*, *pass_key=False*, *prev_key=''*, *splitval='/'*, *walk_np_arrays=False*)

Walk a nested list and/or dict recursively and call fn on all non list or dict objects.

Example

```
dol = {'a': [1, 2], 'b': {'c': 3, 'd': 4}}

def fn(val):
    return val**2

result = walk(dol, fn)
```

(continues on next page)

(continued from previous page)

```
print(result)  # {'a': [1, 4], 'b': {'c': 9, 'd': 16}}
print(dol)  # {'a': [1, 2], 'b': {'c': 3, 'd': 4}}

result = walk(dol, fn, inplace=True)
print(result)  # {'a': [1, 4], 'b': {'c': 9, 'd': 16}}
print(dol)  # {'a': [1, 4], 'b': {'c': 9, 'd': 16}}
```

Parameters

- **dict_or_list** (*dict or list*) – Possibly nested list or dictionary.
- **fn** (*Callable*) – Applied to each leave of the nested list_dict-object.
- **inplace** (*bool*) – If False, a new object with the same structure and the results of fn at the leaves is created. If True the leaves are replaced by the results of fn.
- **pass_key** (*bool*) – Also passes the key or index of the leave element to `fn`.
- **prev_key** (*str*) – If `pass_key == True` keys of parent nodes are passed to calls of `walk` on child nodes to accumulate the keys.
- **splitval** (*str*) – String used to join keys if `pass_key` is `True`.
- **walk_np_arrays** (*bool*) – If `True`, numpy arrays are intepreted as list, ie not as leaves.

Returns

- *The resulting nested list-dict-object with the results of*
- **fn at its leaves.** (*dict or list*)

1.3 Module contents

CHAPTER 2

Indices and tables

- genindex
- modindex
- search

Python Module Index

h

Index