FlatDict Documentation

Release 1.1.1

Gavin M. Roy

Mar 07, 2018

Contents

FlatDict is a Python module for interacting with nested dicts as a single level dict with delimited keys. FlatDict supports Python 2.7+ and 3.4+.

Jump to *Installation*, *Example Use*, *Class Documentation*, or license.

For example:

```
foo = {'foo': {'bar': 'baz', 'qux': 'corge'}}
```

is represented as:

```
{'foo:bar': 'baz',
 'foo:qux': 'corge'}
```

And can still be accessed as:

```
foo['foo']['bar']
```

and

```
foo['foo:bar']
```

Additionally, lists and tuples are also converted into dicts using enumerate().

For example:

```
d = {'list': ['a', 'b', 'c',]}
```

Will be flattened as follows:

```
flat = {'list:0': 'a', 'list:1': 'b', 'list:2': 'c'}
```

CHAPTER 1

Installation

```
$ pip install flatdict
```

CHAPTER 2

Example Use

```
import flatdict

values = {'foo': {'bar': {'baz': 0,
                          'qux': 1,
                          'corge': 2},
                  'grault': {'baz': 3,
                             'qux': 4,
                             'corge': 5}},
          'garply': {'foo': 0, 'bar': 1, 'baz': 2, 'qux': {'corge': 3}}}

flat = flatdict.FlatDict(values)

print(flat['foo:bar:baz'])

flat['test:value:key'] = 10

del flat['test']

for key in flat:
    print(key)

for values in flat.itervalues():
    print(value)

print(repr(flat.as_dict()))

print(flat == flat.as_dict())
```

CHAPTER 3

Class Documentation

FlatDict is a dict object that allows for single level, delimited key/value pair mapping of nested dictionaries.

class flatdict.**FlatDict**(*value=None*, *delimiter=':'*)

FlatDict is a dictionary object that allows for single level, delimited key/value pair mapping of nested dictionaries. The default delimiter value is `:` but can be changed in the constructor or by calling *FlatDict.set_delimiter()*.

as_dict()

Return the *FlatDict* as a `dict`

Return type dict

clear()

Remove all items from the flat dictionary.

copy()

Return a shallow copy of the flat dictionary.

Return type *flatdict.FlatDict*

fromkeys(*S*[, *v*]) → New dict with keys from S and values equal to v.
v defaults to None.

get(*key*, *d=None*)

Return the value for key if key is in the flat dictionary, else default. If default is not given, it defaults to `None`, so that this method never raises `KeyError`.

Parameters

- **key** (*mixed*) – The key to get
- **d** (*mixed*) – The default value

Return type mixed

has_key(*k*) → True if D has a key k, else False

items()

Return a copy of the flat dictionary's list of `(key, value)` pairs.

Note: CPython implementation detail: Keys and values are listed in an arbitrary order which is non-random, varies across Python implementations, and depends on the flat dictionary's history of insertions and deletions.

Return type list

iteritems()
Return an iterator over the flat dictionary's (key, value) pairs. See the note for *flatdict.FlatDict.items()*.

Using `iteritems()` while adding or deleting entries in the flat dictionary may raise `RuntimeError` or fail to iterate over all entries.

Return type Iterator

Raises RuntimeError

iterkeys()
Iterate over the flat dictionary's keys. See the note for *flatdict.FlatDict.items()*.

Using `iterkeys()` while adding or deleting entries in the flat dictionary may raise `RuntimeError` or fail to iterate over all entries.

Return type Iterator

Raises RuntimeError

itervalues()
Return an iterator over the flat dictionary's values. See the note *flatdict.FlatDict.items()*.

Using `itervalues()` while adding or deleting entries in the flat dictionary may raise a `RuntimeError` or fail to iterate over all entries.

Return type Iterator

Raises RuntimeError

keys()
Return a copy of the flat dictionary's list of keys. See the note for *flatdict.FlatDict.items()*.

Return type list

pop(*key*, *default=None*)
If key is in the flat dictionary, remove it and return its value, else return default. If default is not given and key is not in the dictionary, `KeyError` is raised.

Parameters

- **key** (*mixed*) – The key name
- **default** (*mixed*) – The default value

Return type mixed

popitem() → (k, v), remove and return some (key, value) pair as a
2-tuple; but raise KeyError if D is empty.

set_delimiter(*delimiter*)
Override the default or passed in delimiter with a new value. If the requested delimiter already exists in a key, a `ValueError` will be raised.

Parameters **delimiter** (*str*) – The delimiter to use

Raises ValueError

setdefault(*key*, *default*)
If key is in the flat dictionary, return its value. If not, insert key with a value of default and return default. default defaults to `None`.

Parameters

- **key** (*mixed*) – The key name
- **default** (*mixed*) – The default value

Return type mixed

update(*other=None*, ***kwargs*)
Update the flat dictionary with the key/value pairs from other, overwriting existing keys.

`update()` accepts either another flat dictionary object or an iterable of key/value pairs (as tuples or other iterables of length two). If keyword arguments are specified, the flat dictionary is then updated with those key/value pairs: `d.update(red=1, blue=2)`.

Parameters **other** (*iterable*) – Iterable of key, value pairs

Return type None

values()
Return a copy of the flat dictionary's list of values. See the note for *flatdict.FlatDict.items()*.

Return type list

viewitems() → a set-like object providing a view on D's items

viewkeys() → a set-like object providing a view on D's keys

viewvalues() → an object providing a view on D's values

class flatdict.**FlatterDict**(*value=None*, *delimiter=':'*)
Like *FlatDict* but also coerces lists and sets to child-dict instances with the offset as the key. Alternative to the implementation added in v1.2 of FlatDict.

as_dict()
Return the *FlatDict* as a `dict`, or if the class was originally populated as a list or tuple, return it as such.

Return type dict

clear()
Remove all items from the flat dictionary.

copy()
Return a shallow copy of the flat dictionary.

Return type *flatdict.FlatDict*

fromkeys(*S*[, *v*]) → New dict with keys from S and values equal to v.
v defaults to None.

get(*key*, *d=None*)
Return the value for key if key is in the flat dictionary, else default. If default is not given, it defaults to `None`, so that this method never raises `KeyError`.

Parameters

- **key** (*mixed*) – The key to get
- **d** (*mixed*) – The default value

Return type mixed

has_key(*k*) → True if D has a key k, else False

items()
Return a copy of the flat dictionary's list of `(key, value)` pairs.

Note: CPython implementation detail: Keys and values are listed in an arbitrary order which is non-random, varies across Python implementations, and depends on the flat dictionary's history of insertions and deletions.

Return type list

iteritems()
Return an iterator over the flat dictionary's (key, value) pairs. See the note for *flatdict.FlatDict.items()*.

Using `iteritems()` while adding or deleting entries in the flat dictionary may raise `RuntimeError` or fail to iterate over all entries.

Return type Iterator

Raises RuntimeError

iterkeys()
Iterate over the flat dictionary's keys. See the note for *flatdict.FlatDict.items()*.

Using `iterkeys()` while adding or deleting entries in the flat dictionary may raise `RuntimeError` or fail to iterate over all entries.

Return type Iterator

Raises RuntimeError

itervalues()
Return an iterator over the flat dictionary's values. See the note *flatdict.FlatDict.items()*.

Using `itervalues()` while adding or deleting entries in the flat dictionary may raise a `RuntimeError` or fail to iterate over all entries.

Return type Iterator

Raises RuntimeError

keys()
Return a copy of the flat dictionary's list of keys. See the note for *flatdict.FlatDict.items()*.

Return type list

pop(*key*, *default=None*)
If key is in the flat dictionary, remove it and return its value, else return default. If default is not given and key is not in the dictionary, `KeyError` is raised.

Parameters

- **key** (*mixed*) – The key name
- **default** (*mixed*) – The default value

Return type mixed

popitem() → (k, v), remove and return some (key, value) pair as a
2-tuple; but raise KeyError if D is empty.

set_delimiter(*delimiter*)
Override the default or passed in delimiter with a new value. If the requested delimiter already exists in a key, a `ValueError` will be raised.

Parameters **delimiter** (*str*) – The delimiter to use

Raises ValueError

setdefault(*key*, *default*)
If key is in the flat dictionary, return its value. If not, insert key with a value of default and return default. default defaults to `None`.

Parameters

- **key** (*mixed*) – The key name
- **default** (*mixed*) – The default value

Return type mixed

update(*other=None*, ***kwargs*)
Update the flat dictionary with the key/value pairs from other, overwriting existing keys.

`update()` accepts either another flat dictionary object or an iterable of key/value pairs (as tuples or other iterables of length two). If keyword arguments are specified, the flat dictionary is then updated with those key/value pairs: `d.update(red=1, blue=2)`.

Parameters **other** (*iterable*) – Iterable of key, value pairs

Return type None

values()
Return a copy of the flat dictionary's list of values. See the note for *flatdict.FlatDict.items()*.

Return type list

viewitems() → a set-like object providing a view on D's items

viewkeys() → a set-like object providing a view on D's keys

viewvalues() → an object providing a view on D's values

Python Module Index

f

Index