

---

# Dictdiffer Documentation

*Release 0.5.0.post1*

**Fatih Erikli**

January 04, 2016



<b>1</b>	<b>Installation</b>	<b>3</b>
<b>2</b>	<b>Usage</b>	<b>5</b>
<b>3</b>	<b>API</b>	<b>7</b>
<b>4</b>	<b>Changes</b>	<b>9</b>
<b>5</b>	<b>Contributing</b>	<b>11</b>
<b>6</b>	<b>License</b>	<b>13</b>
<b>7</b>	<b>Authors</b>	<b>15</b>
	<b>Python Module Index</b>	<b>17</b>



Dictdiffer is a helper module that helps you to diff and patch dictionaries.



---

## Installation

---

Dictdiffer is on PyPI so all you need is:

```
$ pip install dictdiffer
```





---

## Usage

---

Let's start with an example on how to find the diff between two dictionaries using `diff()` method:

```
from dictdiffer import diff, patch, swap, revert

first = {
    "title": "hello",
    "fork_count": 20,
    "stargazers": ["/users/20", "/users/30"],
    "settings": {
        "assignees": [100, 101, 201],
    }
}

second = {
    "title": "hellooo",
    "fork_count": 20,
    "stargazers": ["/users/20", "/users/30", "/users/40"],
    "settings": {
        "assignees": [100, 101, 202],
    }
}

result = diff(first, second)

assert list(result) == [
    ('change', ['settings', 'assignees', 2], (201, 202)),
    ('add', 'stargazers', [(2, '/users/40')]),
    ('change', 'title', ('hello', 'hellooo'))]
```

Now we can apply the diff result with `patch()` method:

```
result = diff(first, second)
patched = patch(result, first)

assert patched == second
```

Also we can swap the diff result with `swap()` method:

```
result = diff(first, second)
swapped = swap(result)

assert list(swapped) == [
    ('change', ['settings', 'assignees', 2], (202, 201)),
```

```
('remove', 'stargazers', [(2, '/users/40')]),  
('change', 'title', ('hellooo', 'hello'))]
```

Let's revert the last changes:

```
result = diff(first, second)  
reverted = revert(result, patched)  
assert reverted == first
```

A tolerance can be used to consider closed values as equal. The tolerance parameter only applies for int and float.

Let's try with a tolerance of 10% with the values 10 and 10.5:

```
first = {'a': 10.0}  
second = {'a': 10.5}  
  
result = diff(first, second, tolerance=0.1)  
  
assert list(result) == []
```

Now with a tolerance of 1%:

```
result = diff(first, second, tolerance=0.01)  
  
assert list(result) == ('change', 'a', (10.0, 10.5))
```

Dictdiffer is a helper module to diff and patch dictionaries.

`dictdiffer.diff` (*first*, *second*, *node=None*, *ignore=None*, *path\_limit=None*, *expand=False*, *tolerance=2.220446049250313e-16*)

Compare two dictionary/list/set objects, and returns a diff result.

Return iterator with differences between two dictionaries.

```
>>> result = diff({'a': 'b'}, {'a': 'c'})
>>> list(result)
[('change', 'a', ('b', 'c'))]
```

**PathLimit:**

```
>>> list(diff({}, {'a': {'b': 'c'}}))
[('add', '', [('a', {'b': 'c'})])]
```

```
>>> from dictdiffer.utils import PathLimit
>>> list(diff({}, {'a': {'b': 'c'}}), path_limit=PathLimit())
[('add', '', [('a', {})]), ('add', 'a', [('b', 'c')])]
```

```
>>> from dictdiffer.utils import PathLimit
>>> list(diff({}, {'a': {'b': 'c'}}), path_limit=PathLimit(['a', ]))
[('add', '', [('a', {'b': 'c'})])]
```

```
>>> from dictdiffer.utils import PathLimit
>>> list(diff({}, {'a': {'b': 'c'}},
...         path_limit=PathLimit(['a', 'b'])))
[('add', '', [('a', {})]), ('add', 'a', [('b', 'c')])]
```

**Expand:**

```
... list(diff({}, {'foo': 'bar', 'apple': 'mango'})) [('add', '', [('foo', 'bar'), ('apple', 'mango')])]
```

```
... list(diff({}, {'foo': 'bar', 'apple': 'mango'}, expand=True)) [('add', '', [('foo', 'bar')]), ('add', '',
[('apple', 'mango')])]
```

**Parameters**

- **first** – original dictionary, list or set
- **second** – new dictionary, list or set
- **node** – key for comparison that can be used in `dot_lookup()`
- **ignore** – list of keys that should not be checked

- **path\_limit** – List of path limit tuples or dictdiffer.utils.Pathlimit object to limit the diff recursion depth
- **expand** – Expands the patches
- **tolerance** – threshold to consider 2 values identical

Changed in version 0.3: Added *ignore* parameter.

Changed in version 0.4: Arguments *first* and *second* can now contain a set.

Changed in version 0.5: Added *path\_limit* parameter. Added *expand* paramter. Added *tolerance* parameter.

`dictdiffer.patch(diff_result, destination)`

Patch the diff result to the old dictionary.

`dictdiffer.swap(diff_result)`

Swap the diff result.

It uses following mapping:

- remove -> add
- add -> remove

In addition, swap the changed values for *change* flag.

```
>>> swapped = swap([('add', 'a.b.c', ('a', 'b'))])
>>> next(swapped)
('remove', 'a.b.c', ('a', 'b'))
```

```
>>> swapped = swap([('change', 'a.b.c', ('a', 'b'))])
>>> next(swapped)
('change', 'a.b.c', ('b', 'a'))
```

`dictdiffer.revert(diff_result, destination)`

Call swap function to revert patched dictionary object.

Usage example:

```
>>> first = {'a': 'b'}
>>> second = {'a': 'c'}
>>> revert(diff(first, second), second)
{'a': 'b'}
```

`dictdiffer.dot_lookup(source, lookup, parent=False)`

Allow you to reach dictionary items with string or list lookup.

Recursively find value by lookup key split by `'.'`.

```
>>> dot_lookup({'a': {'b': 'hello'}}, 'a.b')
'hello'
```

If parent argument is True, returns the parent node of matched object.

```
>>> dot_lookup({'a': {'b': 'hello'}}, 'a.b', parent=True)
{'b': 'hello'}
```

If node is empty value, returns the whole dictionary object.

```
>>> dot_lookup({'a': {'b': 'hello'}}, '')
{'a': {'b': 'hello'}}
```

---

## Changes

---

Version 0.5.0 (released 2016-01-04)

- Adds tolerance parameter used when user wants to treat closed values as equals
- Adds support for comparing numerical values and NaN. (#54) (#55)

Version 0.4.0 (released 2015-03-11)

- Adds support for diffing and patching of sets. (#44)
- New tests for diff on the same lists. (#48)
- Fix for exception when dict has unicode keys and ignore parameter is provided. (#50)
- PEP8 improvements.

Version 0.3.0 (released 2014-11-05)

- Adds ignore argument to *diff* function that allows skipping check on specified keys. (#34 #35)
- Fix for diffing of dict or list subclasses. (#37)
- Better instance checking of diffing objects. (#39)

Version 0.2.0 (released 2014-09-29)

- Fix for empty list instructions. (#30)
- Regression test for empty list instructions.

Version 0.1.0 (released 2014-09-01)

- Fix for list removal issues during patching caused by wrong iteration. (#10)
- Fix for issues with multiple value types for the same key. (#10)
- Fix for issues with strings handled as iterables. (#6)
- Fix for integer keys. (#12)
- Regression test for complex dictionaries. (#4)
- Better testing with Travis CI, tox, pytest, code coverage. (#10)
- Initial release of documentation on ReadTheDocs. (#21 #24)
- Support for Python 3. (#15)

Version 0.0.4 (released 2014-01-04)

- List diff behavior treats lists as lists instead of sets. (#3)

- Differed typed objects are flagged as *changed* now.
- Swap function refactored.

Version 0.0.3 (released 2013-05-26)

- Initial public release on PyPI.

---

## Contributing

---

Bug reports, feature requests, and other contributions are welcome. If you find a demonstrable problem that is caused by the code of this library, please:

1. Search for [already reported problems](#).
2. Check if the issue has been fixed or is still reproducible on the latest *master* branch.
3. Create an issue with **a test case**.

If you create a feature branch, you can run the tests to ensure everything is operating correctly:

```
$ ./run-tests.sh

...

Name                               Stmt%  Miss  Cover  Missing
-----
dictdiff/___init___                88      0  100%
dictdiff/version                    2      0  100%
-----
TOTAL                             90      0  100%

...

52 passed, 2 skipped in 0.44 seconds
```





---

### License

---

Dictdiffer is free software; you can redistribute it and/or modify it under the terms of the MIT License quoted below.

Copyright (C) 2013 Fatih Erikli. Copyright (C) 2013, 2014 CERN.

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the “Software”), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED “AS IS”, WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

In applying this license, CERN does not waive the privileges and immunities granted to it by virtue of its status as an Intergovernmental Organization or submit itself to any jurisdiction.



---

### Authors

---

Dictdiffer was originally developed by Fatih Erikli. It is now being developed and maintained by the Invenio collaboration. You can contact us at [info@invenio-software.org](mailto:info@invenio-software.org).

Contributors:

- Fatih Erikli <[fatiharikli@gmail.com](mailto:fatiharikli@gmail.com)>
- Brian Rue <[brianrue@gmail.com](mailto:brianrue@gmail.com)>
- Lars Holm Nielsen <[lars.holm.nielsen@cern.ch](mailto:lars.holm.nielsen@cern.ch)>
- Tibor Simko <[tibor.simko@cern.ch](mailto:tibor.simko@cern.ch)>
- Jiri Kuncar <[jiri.kuncar@cern.ch](mailto:jiri.kuncar@cern.ch)>
- Jason Peddle <[jwpeddle@gmail.com](mailto:jwpeddle@gmail.com)>
- Martin Vesper <[martin.vesper@cern.ch](mailto:martin.vesper@cern.ch)>
- Gilles DAVID <[frodon1@gmail.com](mailto:frodon1@gmail.com)>
- Alexander Mohr <[amohr@farmersbusinessnetwork.com](mailto:amohr@farmersbusinessnetwork.com)>



## d

`dictdiffer`, [7](#)



## D

`dictdiffer` (module), 7

`diff()` (in module `dictdiffer`), 7

`dot_lookup()` (in module `dictdiffer`), 8

## P

`patch()` (in module `dictdiffer`), 8

## R

`revert()` (in module `dictdiffer`), 8

## S

`swap()` (in module `dictdiffer`), 8