mergedeep Documentation

Release 1.3.4

Travis Clarke

Sep 14, 2021

Contents

Version 1.3.4

A deep merge function for .

CHAPTER 1

Installation

```
$ pip install mergedeep
```

CHAPTER 2

Usage

```
merge(destination: MutableMapping, *sources: Mapping, strategy: Strategy = Strategy.
↪REPLACE) -> MutableMapping
```

Deep merge without mutating the source dicts.

```
from mergedeep import merge

a = {"keyA": 1}
b = {"keyB": {"sub1": 10}}
c = {"keyB": {"sub2": 20}}

merged = merge({}, a, b, c)

print(merged)
# {"keyA": 1, "keyB": {"sub1": 10, "sub2": 20}}
```

Deep merge into an existing dict.

```
from mergedeep import merge

a = {"keyA": 1}
b = {"keyB": {"sub1": 10}}
c = {"keyB": {"sub2": 20}}

merge(a, b, c)

print(a)
# {"keyA": 1, "keyB": {"sub1": 10, "sub2": 20}}
```

2.1 Merge strategies:

- Replace (*default*)

`Strategy.REPLACE`

```
# When `destination` and `source` keys are the same, replace the `destination` value␣
↪with one from `source` (default).

# Note: with multiple sources, the `last` (i.e. rightmost) source value will be what␣
↪appears in the merged result.

from mergedeep import merge, Strategy

dst = {"key": [1, 2]}
src = {"key": [3, 4]}

merge(dst, src, strategy=Strategy.REPLACE)
# same as: merge(dst, src)

print(dst)
# {"key": [3, 4]}
```

- Additive

  `Strategy.ADDITIVE`

```
# When `destination` and `source` values are both the same additive collection type,␣
↪extend `destination` by adding values from `source`.
# Additive collection types include: `list`, `tuple`, `set`, and `Counter`

# Note: if the values are not additive collections of the same type, then fallback to␣
↪a `REPLACE` merge.

from mergedeep import merge, Strategy

dst = {"key": [1, 2], "count": Counter({"a": 1, "b": 1})}
src = {"key": [3, 4], "count": Counter({"a": 1, "c": 1})}

merge(dst, src, strategy=Strategy.ADDITIVE)

print(dst)
# {"key": [1, 2, 3, 4], "count": Counter({"a": 2, "b": 1, "c": 1})}
```

- Typesafe replace

  `Strategy.TYPESAFE_REPLACE` or `Strategy.TYPESAFE`

```
# When `destination` and `source` values are of different types, raise `TypeError`.␣
↪Otherwise, perform a `REPLACE` merge.

from mergedeep import merge, Strategy

dst = {"key": [1, 2]}
src = {"key": {3, 4}}

merge(dst, src, strategy=Strategy.TYPESAFE_REPLACE) # same as: `Strategy.TYPESAFE`
# TypeError: destination type: <class 'list'> differs from source type: <class 'set'>␣
↪for key: "key"
```

- Typesafe additive

  `Strategy.TYPESAFE_ADDITIVE`

```
# When `destination` and `source` values are of different types, raise `TypeError`.␣
↪Otherwise, perform a `ADDITIVE` merge.

from mergedeep import merge, Strategy

dst = {"key": [1, 2]}
src = {"key": {3, 4}}

merge(dst, src, strategy=Strategy.TYPESAFE_ADDITIVE)
# TypeError: destination type: <class 'list'> differs from source type: <class 'set'>␣
↪for key: "key"
```

CHAPTER 3

License