OrderedDict — Remember the Order Keys are Added to a Dictionary

An OrderedDict is a dictionary subclass that remembers the order in which its contents are added.

```
# collections_ordereddict_iter.py

import collections

print('Regular dictionary:')
d = {}
d['a'] = 'A'
d['b'] = 'B'
d['c'] = 'C'

for k, v in d.items():
    print(k, v)

print('\nOrderedDict:')
d = collections.OrderedDict()
d['a'] = 'A'
d['b'] = 'B'
d['c'] = 'C'

for k, v in d.items():
    print(k, v)
```

Before Python 3.6 a regular dict did not track the insertion order, and iterating over it produced the values in order based on how the keys are stored in the hash table, which is in turn influenced by a random value to reduce collisions. In an OrderedDict, by contrast, the order in which the items are inserted is remembered and used when creating an iterator.

```
$ python3.5 collections_ordereddict_iter.py

Regular dictionary:
c C
b B
a A

OrderedDict:
a A
b B
c C
```

Under Python 3.6, the built-in dict does track insertion order, although this behavior is a side-effect of an implementation change and should not be relied on.

```
$ python3.6 collections_ordereddict_iter.py

Regular dictionary:
a A
b B
c C

OrderedDict:
a A
b B
c C
```

Equality

A regular dict looks at its contents when testing for equality. An OrderedDict also considers the order in which the items were added.

```
# collections_ordereddict_equality.py

import collections

print('dict       :', end=' ')
d1 = {}
d1['a'] = 'A'
d1['b'] = 'B'
d1['c'] = 'C'

d2 = {}
d2['c'] = 'C'
d2['b'] = 'B'
d2['a'] = 'A'

print(d1 == d2)

print('OrderedDict:', end=' ')

d1 = collections.OrderedDict()
d1['a'] = 'A'
d1['b'] = 'B'
d1['c'] = 'C'

d2 = collections.OrderedDict()
d2['c'] = 'C'
d2['b'] = 'B'
d2['a'] = 'A'

print(d1 == d2)
```

In this case, since the two ordered dictionaries are created from values in a different order, they are considered to be different.

```
$ python3 collections_ordereddict_equality.py

dict       : True
OrderedDict: False
```

Reordering

It is possible to change the order of the keys in an OrderedDict by moving them to either the beginning or the end of the sequence using move_to_end().

```
# collections_ordereddict_move_to_end.py

import collections

d = collections.OrderedDict(
    [('a', 'A'), ('b', 'B'), ('c', 'C')]
)

print('Before:')
for k, v in d.items():
    print(k, v)

d.move_to_end('b')

print('\nmove_to_end():')
for k, v in d.items():
    print(k, v)

d.move_to_end('b', last=False)

print('\nmove_to_end(last=False):')
for k, v in d.items():
        print(k, v)
```

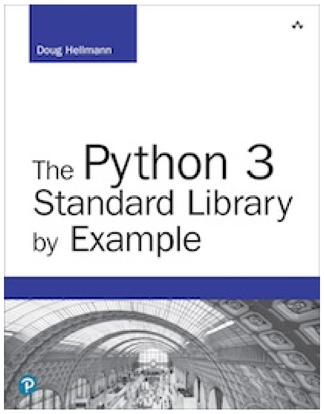
The `last` argument tells `move_to_end()` whether to move the item to be the last item in the key sequence (when `True`) or the first (when `False`).

```
$ python3 collections_ordereddict_move_to_end.py

Before:
a A
b B
c C

move_to_end():
a A
c C
b B

move_to_end(last=False):
b B
a A
c C
```

See also

- PYTHONHASHSEED – Environment variable to control the random seed value added to the hash algorithm for key locations in the dictionary.

This page was last updated 2018-03-18.



The output from all the example programs from PyMOTW-3 has been generated with Python 3.7.1, unless otherwise noted. Some of the features described here may not be available in earlier versions of Python.

Looking for examples for Python 2?