# 1.20 合并多个字典或映射¶

## 问题¶

现在有多个字典或者映射，你想将它们从逻辑上合并为一个单一的映射后执行某些操作， 比如查找值或者检查某些键是否存在。

## 解决方案¶

假如你有如下两个字典:

```
a = {'x': 1, 'z': 3 }
b = {'y': 2, 'z': 4 }
```

现在假设你必须在两个字典中执行查找操作（比如先从 a 中找，如果找不到再在 b 中找）。 一个非常简单的解决方案就是使用 collections 模块中的 ChainMap 类。比如：

```
from collections import ChainMap
c = ChainMap(a,b)
print(c['x']) # Outputs 1 (from a)
print(c['y']) # Outputs 2 (from b)
print(c['z']) # Outputs 3 (from a)
```

## 讨论¶

一个 ChainMap 接受多个字典并将它们在逻辑上变为一个字典。 然后，这些字典并不是真的合并在一起了， ChainMap 类只是在内部创建了一个容纳这些字典的列表 并重新定义了一些常见的字典操作来遍历这个列表。大部分字典操作都是可以正常使用的，比如：

```
>>> len(c)
3
>>> list(c.keys())
['x', 'y', 'z']
>>> list(c.values())
[1, 2, 3]
>>>
```

如果出现重复键，那么第一次出现的映射值会被返回。 因此，例子程序中的 c['z'] 总是会返回字典 a 中对应的值，而不是 b 中对应的值。

对于字典的更新或删除操作总是影响的是列表中第一个字典。比如：

```
>>> c['z'] = 10
>>> c['w'] = 40
>>> del c['x']
>>> a
{'w': 40, 'z': 10}
>>> del c['y']
Traceback (most recent call last):
...
KeyError: "Key not found in the first mapping: 'y'"
>>>
```

ChainMap 对于编程语言中的作用范围变量（比如 globals , locals 等）是非常有用的。 事实上，有一些方法可以使它变得简单：

```
>>> values = ChainMap()
>>> values['x'] = 1
>>> # Add a new mapping
>>> values = values.new_child()
>>> values['x'] = 2
>>> # Add a new mapping
>>> values = values.new_child()
```

```
>>> values['x'] = 3
>>> values
ChainMap({'x': 3}, {'x': 2}, {'x': 1})
>>> values['x']
3
>>> # Discard last mapping
>>> values = values.parents
>>> values['x']
2
>>> # Discard last mapping
>>> values = values.parents
>>> values['x']
1
>>> values
ChainMap({'x': 1})
>>>
```

作为 ChainMap 的替代，你可能会考虑使用 update() 方法将两个字典合并。比如：

```
>>> a = {'x': 1, 'z': 3 }
>>> b = {'y': 2, 'z': 4 }
>>> merged = dict(b)
>>> merged.update(a)
>>> merged['x']
1
>>> merged['y']
2
>>> merged['z']
3
>>>
```

这样也能行得通，但是它需要你创建一个完全不同的字典对象（或者是破坏现有字典结构）。 同时，如果原字典做了更新，这种改变不会反应到新的合并字典中去。比如：

```
>>> a['x'] = 13
>>> merged['x']
1
```

ChainMap 使用原来的字典，它自己不创建新的字典。所以它并不会产生上面所说的结果，比如：

```
>>> a = {'x': 1, 'z': 3 }
>>> b = {'y': 2, 'z': 4 }
>>> merged = ChainMap(a, b)
>>> merged['x']
1
>>> a['x'] = 42
>>> merged['x'] # Notice change to merged dicts
42
>>>
```