4.9 排列组合的迭代¶

问题¶

你想迭代遍历一个集合中元素的所有可能的排列或组合

解决方案¶

itertools模块提供了三个函数来解决这类问题。 其中一个是 `itertools.permutations()`， 它接受一个集合并产生一个元组序列，每个元组由集合中所有元素的一个可能排列组成。 也就是说通过打乱集合中元素排列顺序生成一个元组，比如：

```
>>> items = ['a', 'b', 'c']
>>> from itertools import permutations
>>> for p in permutations(items):
...     print(p)
...
('a', 'b', 'c')
('a', 'c', 'b')
('b', 'a', 'c')
('b', 'c', 'a')
('c', 'a', 'b')
('c', 'b', 'a')
>>>
```

如果你想得到指定长度的所有排列，你可以传递一个可选的长度参数。就像这样：

```
>>> for p in permutations(items, 2):
...     print(p)
...
('a', 'b')
('a', 'c')
('b', 'a')
('b', 'c')
('c', 'a')
('c', 'b')
>>>
```

使用 `itertools.combinations()` 可得到输入集合中元素的所有的组合。比如：

```
>>> from itertools import combinations
>>> for c in combinations(items, 3):
...     print(c)
...
('a', 'b', 'c')

>>> for c in combinations(items, 2):
...     print(c)
...
('a', 'b')
('a', 'c')
('b', 'c')

>>> for c in combinations(items, 1):
...     print(c)
...
('a',)
('b',)
('c',)
>>>
```

对于 `combinations()` 来讲，元素的顺序已经不重要了。 也就是说，组合 `('a', 'b')` 跟 `('b', 'a')` 其实是一样的(最终只会输出其中一个)。

在计算组合的时候，一旦元素被选取就会从候选中剔除掉(比如如果元素'a'已经被选取了，那么接下来就不会再考虑它

了)。 而函数 itertools.combinations_with_replacement() 允许同一个元素被选择多次，比如：

```
>>> for c in combinations_with_replacement(items, 3):
...     print(c)
...
('a', 'a', 'a')
('a', 'a', 'b')
('a', 'a', 'c')
('a', 'b', 'b')
('a', 'b', 'c')
('a', 'c', 'c')
('b', 'b', 'b')
('b', 'b', 'c')
('b', 'c', 'c')
('c', 'c', 'c')
>>>
```

讨论¶

这一小节我们向你展示的仅仅是 itertools 模块的一部分功能。 尽管你也可以自己手动实现排列组合算法，但是这样做得要花点脑力。 当我们碰到看上去有些复杂的迭代问题时，最好可以先去看看itertools模块。 如果这个问题很普遍，那么很有可能会在里面找到解决方案！