4.5 反向迭代¶

问题¶

你想反方向迭代一个序列

解决方案¶

使用内置的 `reversed()` 函数，比如：

```
>>> a = [1, 2, 3, 4]
>>> for x in reversed(a):
...     print(x)
...
4
3
2
1
```

反向迭代仅仅当对象的大小可预先确定或者对象实现了 `__reversed__()` 的特殊方法时才能生效。 如果两者都不符合，那你必须先将对象转换为一个列表才行，比如：

```
# Print a file backwards
f = open('somefile')
for line in reversed(list(f)):
    print(line, end='')
```

要注意的是如果可迭代对象元素很多的话，将其预先转换为一个列表要消耗大量的内存。

讨论¶

很多程序员并不知道可以通过在自定义类上实现 `__reversed__()` 方法来实现反向迭代。比如：

```
class Countdown:
    def __init__(self, start):
        self.start = start

    # Forward iterator
    def __iter__(self):
        n = self.start
        while n > 0:
            yield n
            n -= 1

    # Reverse iterator
    def __reversed__(self):
        n = 1
        while n <= self.start:
            yield n
            n += 1

for rr in reversed(Countdown(30)):
    print(rr)
```

```
for rr in Countdown(30):
    print(rr)
```

定义一个反向迭代器可以使得代码非常的高效， 因为它不再需要将数据填充到一个列表中然后再去反向迭代这个列表。