4.10 序列上索引值迭代¶

问题¶

你想在迭代一个序列的同时跟踪正在被处理的元素索引。

解决方案¶

内置的 `enumerate()` 函数可以很好的解决这个问题：

```
>>> my_list = ['a', 'b', 'c']
>>> for idx, val in enumerate(my_list):
...     print(idx, val)
...
0 a
1 b
2 c
```

为了按传统行号输出(行号从1开始)，你可以传递一个开始参数：

```
>>> my_list = ['a', 'b', 'c']
>>> for idx, val in enumerate(my_list, 1):
...     print(idx, val)
...
1 a
2 b
3 c
```

这种情况在你遍历文件时想在错误消息中使用行号定位时候非常有用：

```
def parse_data(filename):
    with open(filename, 'rt') as f:
        for lineno, line in enumerate(f, 1):
            fields = line.split()
            try:
                count = int(fields[1])
                ...
            except ValueError as e:
                print('Line {}: Parse error: {}'.format(lineno, e))
```

`enumerate()` 对于跟踪某些值在列表中出现的位置是很有用的。 所以，如果你想将一个文件中出现的单词映射到它出现的行号上去，可以很容易的利用 `enumerate()` 来完成：

```
word_summary = defaultdict(list)

with open('myfile.txt', 'r') as f:
    lines = f.readlines()

for idx, line in enumerate(lines):
    # Create a list of words in current line
    words = [w.strip().lower() for w in line.split()]
    for word in words:
        word_summary[word].append(idx)
```

如果你处理完文件后打印 `word_summary`，会发现它是一个字典(准确来讲是一个 `defaultdict`)， 对于每个单词有一个 `key`，每个 `key` 对应的值是一个由这个单词出现的行号组成的列表。 如果某个单词在一行中出现过两次，那么这个行号也会出现两次， 同时也可以作为文本的一个简单统计。

讨论¶

当你想额外定义一个计数变量的时候，使用 `enumerate()` 函数会更加简单。你可能会像下面这样写代码：

```
lineno = 1
for line in f:
    # Process line
    ...
    lineno += 1
```

但是如果使用 `enumerate()` 函数来代替就显得更加优雅了：

```
for lineno, line in enumerate(f):
    # Process line
    ...
```

`enumerate()` 函数返回的是一个 `enumerate` 对象实例， 它是一个迭代器，返回连续的包含一个计数和一个值的元组， 元组中的值通过在传入序列上调用 `next()` 返回。

还有一点可能并不很重要，但是也值得注意， 有时候当你在一个已经解压后的元组序列上使用 `enumerate()` 函数时很容易调入陷阱。 你得像下面正确的方式这样写：

```
data = [ (1, 2), (3, 4), (5, 6), (7, 8) ]

# Correct!
for n, (x, y) in enumerate(data):
    ...
# Error!
for n, x, y in enumerate(data):
    ...
```