6.1 读写CSV数据¶

问题¶

你想读写一个CSV格式的文件。

解决方案¶

对于大多数的CSV格式的数据读写问题，都可以使用 `csv` 库。 例如：假设你在一个名叫stocks.csv文件中有一些股票市场数据，就像这样：

```
Symbol,Price,Date,Time,Change,Volume
"AA",39.48,"6/11/2007","9:36am",-0.18,181800
"AIG",71.38,"6/11/2007","9:36am",-0.15,195500
"AXP",62.58,"6/11/2007","9:36am",-0.46,935000
"BA",98.31,"6/11/2007","9:36am",+0.12,104800
"C",53.08,"6/11/2007","9:36am",-0.25,360900
"CAT",78.29,"6/11/2007","9:36am",-0.23,225400
```

下面向你展示如何将这些数据读取为一个元组的序列：

```
import csv
with open('stocks.csv') as f:
    f_csv = csv.reader(f)
    headers = next(f_csv)
    for row in f_csv:
        # Process row
        ...
```

在上面的代码中， `row` 会是一个列表。因此，为了访问某个字段，你需要使用下标，如 `row[0]` 访问Symbol， `row[4]` 访问Change。

由于这种下标访问通常会引起混淆，你可以考虑使用命名元组。例如：

```
from collections import namedtuple
with open('stock.csv') as f:
    f_csv = csv.reader(f)
    headings = next(f_csv)
    Row = namedtuple('Row', headings)
    for r in f_csv:
        row = Row(*r)
        # Process row
        ...
```

它允许你使用列名如 `row.Symbol` 和 `row.Change` 代替下标访问。 需要注意的是这个只有在列名是合法的Python标识符的时候才生效。如果不是的话， 你可能需要修改下原始的列名(如将非标识符字符替换成下划线之类的)。

另外一个选择就是将数据读取到一个字典序列中去。可以这样做：

```
import csv
with open('stocks.csv') as f:
    f_csv = csv.DictReader(f)
    for row in f_csv:
```

```
    # process row
    ...
```

在这个版本中，你可以使用列名去访问每一行的数据了。比如， `row['Symbol']` 或者 `row['Change']`

为了写入CSV数据，你仍然可以使用csv模块，不过这时候先创建一个 `writer` 对象。例如:

```
headers = ['Symbol','Price','Date','Time','Change','Volume']
rows = [('AA', 39.48, '6/11/2007', '9:36am', -0.18, 181800),
        ('AIG', 71.38, '6/11/2007', '9:36am', -0.15, 195500),
        ('AXP', 62.58, '6/11/2007', '9:36am', -0.46, 935000),
       ]

with open('stocks.csv','w') as f:
    f_csv = csv.writer(f)
    f_csv.writerow(headers)
    f_csv.writerows(rows)
```

如果你有一个字典序列的数据，可以像这样做：

```
headers = ['Symbol', 'Price', 'Date', 'Time', 'Change', 'Volume']
rows = [{'Symbol':'AA', 'Price':39.48, 'Date':'6/11/2007',
        'Time':'9:36am', 'Change':-0.18, 'Volume':181800},
        {'Symbol':'AIG', 'Price': 71.38, 'Date':'6/11/2007',
        'Time':'9:36am', 'Change':-0.15, 'Volume': 195500},
        {'Symbol':'AXP', 'Price': 62.58, 'Date':'6/11/2007',
        'Time':'9:36am', 'Change':-0.46, 'Volume': 935000},
        ]

with open('stocks.csv','w') as f:
    f_csv = csv.DictWriter(f, headers)
    f_csv.writeheader()
    f_csv.writerows(rows)
```

讨论¶

你应该总是优先选择csv模块分割或解析CSV数据。例如，你可能会像编写类似下面这样的代码：

```
with open('stocks.csv') as f:
for line in f:
    row = line.split(',')
    # process row
    ...
```

使用这种方式的一个缺点就是你仍然需要去处理一些棘手的细节问题。 比如，如果某些字段值被引号包围，你不得不去除这些引号。 另外，如果一个被引号包围的字段碰巧含有一个逗号，那么程序就会因为产生一个错误大小的行而出错。

默认情况下， `csv` 库可识别Microsoft Excel所使用的CSV编码规则。 这或许也是最常见的形式，并且也会给你带来最好的兼容性。 然而，如果你查看csv的文档，就会发现有很多种方法将它应用到其他编码格式上(如修改分割字符等)。 例如，如果你想读取以tab分割的数据，可以这样做：

```
# Example of reading tab-separated values
with open('stock.tsv') as f:
    f_tsv = csv.reader(f, delimiter='\t')
    for row in f_tsv:
```

```
    # Process row
    ...
```

如果你正在读取CSV数据并将它们转换为命名元组，需要注意对列名进行合法性认证。 例如，一个CSV格式文件有一个包含非法标识符的列头行，类似下面这样：

```
Street Address,Num-Premises,Latitude,Longitude 5412 N CLARK,10,41.980262,-87.668452
```

这样最终会导致在创建一个命名元组时产生一个 `ValueError` 异常而失败。 为了解决这问题，你可能不得不先去修正列标题。 例如，可以像下面这样在非法标识符上使用一个正则表达式替换：

```
import re
with open('stock.csv') as f:
  f_csv = csv.reader(f)
  headers = [ re.sub('[^a-zA-Z_]', '_', h) for h in next(f_csv) ]
  Row = namedtuple('Row', headers)
  for r in f_csv:
    row = Row(*r)
    # Process row
    ...
```

还有重要的一点需要强调的是，csv产生的数据都是字符串类型的，它不会做任何其他类型的转换。 如果你需要做这样的类型转换，你必须自己手动去实现。 下面是一个在CSV数据上执行其他类型转换的例子：

```
col_types = [str, float, str, str, float, int]
with open('stocks.csv') as f:
  f_csv = csv.reader(f)
  headers = next(f_csv)
  for row in f_csv:
    # Apply conversions to the row items
    row = tuple(convert(value) for convert, value in zip(col_types, row))
    ...
```

另外，下面是一个转换字典中特定字段的例子：

```
print('Reading as dicts with type conversion')
field_types = [ ('Price', float),
        ('Change', float),
        ('Volume', int) ]

with open('stocks.csv') as f:
  for row in csv.DictReader(f):
    row.update((key, conversion(row[key]))
        for key, conversion in field_types)
    print(row)
```

通常来讲，你可能并不想过多去考虑这些转换问题。 在实际情况中，CSV文件都或多或少有些缺失的数据，被破坏的数据以及其它一些让转换失败的问题。 因此，除非你的数据确实有保障是准确无误的，否则你必须考虑这些问题(你可能需要增加合适的错误处理机制)。

最后，如果你读取CSV数据的目的是做数据分析和统计的话， 你可能需要看一看 `Pandas` 包。 `Pandas` 包含了一个非常方便的函数叫 `pandas.read_csv()` ， 它可以加载CSV数据到一个 `DataFrame` 对象中去。 然后利用这个对象你就可以生成各种形式的统计、过滤数据以及执行其他高级操作了。 在6.13小节中会有这样一个例子。