的页面，输入你想搜索的任何查询（例如，夏天），然后单击搜索相关按钮。你将看到在“与夏天相关”标题下返回了一些结果，第一个结果是相关性最高（每个结果左边的数字）。单击图形上方的“散点图”选项可查看散点图，其中 x 轴标记为夏天，y 轴用顶部结果标记。忽略在两个轴上绘制的精确数字，因为我们只对相关性和散点图感兴趣。

在散点图上方，单击“导出数据为 CSV”，文件下载将开始。将此文件保存在与程序相同的目录中。

此 CSV 文件与我们之前看到的文件略有不同。在文件的开头，你会看到一些空行和带有'#'符号的行，直到最后你会看到标题和数据。这些行对我们来说是没有用的，使用能打开 CSV 文件的任何软件，手动删除它们，使得文件的第一行是标题。你还需要删除文件末尾的空行。然后保存文件。在这个步骤中，我们清理了文件以便能更简单地使用 Python 执行操作，此步骤通常称为预处理数据。

标题有几列，第一列包含每行中数据的日期（每行的数据对应于此列中日期开始的周数），第二列是你输入的搜索查询，第三列显示与你的搜索查询相关性最高的搜索查询，其他列包含与你输入的搜索查询按相关性降序排列的其他多个搜索查询。这些列中的数字是相应搜索查询的 z 分数。z 分数表示在特定周期间搜索词语的次数与该词每周的总平均搜索次数之间的差异。正的 z 分数值表示搜索次数高于该周搜索次数的均值，负的 z 分数值表示低于均值。

现在，我们只处理第二和第三列。你可以使用 read_csv()函数来读取这些列：

```
def read_csv(filename):

    with open(filename) as f:
        reader = csv.reader(f)
        next(reader)

        summer = []
        highest_correlated = []
❶       for row in reader:
            summer.append(float(row[1]))
            highest_correlated.append(float(row[2]))

    return summer, highest_correlated
```

这很像早期版本的 read_csv 函数，这里的主要变化是我们如何将值附加到从❶处开始的每个列表中：我们现在读取每行的第二个元素和第三个元素，并将它们存储为浮点数。

以下程序使用此函数计算你输入的搜索查询与其相关性最高的查询两者之间相关性的值。它还创建了这些值的散点图：

```
import matplotlib.pyplot as plt
import csv

if __name__ == '__main__':
❶    summer, highest_correlated = read_csv('correlate-summer.csv')
```