# 实验一 文件及高维数据格式

## 学习目标

1. 理解高维数据格式的含义。
2. 掌握CSV文件的读取、写入和分析处理方法。
3. 掌握JSON格式数据的分析和处理方法。

## 概述

高维数据指一维以上的数据。如表格，就可看成是由行与列组成的二维数据。相比一维数据，高维数据的分析与处理难度更高，维数太高的话通常需要先进行降维处理。

本实验主要针对两种常用的文本数据格式CSV和JSON，这些格式在信息的表示、传输、存储中十分常用，为大量的应用程序所使用和支持，同时这些格式也内含对高维数据的支持。

## 1．CSV文件的读取与分析

本节的任务在python 交互界面完成，素材文件请先存放在一个固定的目录下，例如c:\sample文件夹中，文件名使用绝对路径描述。

### 1.1 使用常规文件操作方法创建和读取CSV文件

CSV文件与普通文件没有什么差别，都是文本文件，只是CSV文件采用一种特殊的格式存放数据，文件中各行用换行符分隔，行中列与列之间用逗号分隔。所以CSV文件的创建、读取和写入可与普通文件一致。

1. 在c盘sample文件夹中创建test1.csv文件（sample文件夹已存在），并写入标题及2条记录。

|  |
| --- |
| #filename: sample7-1-1.py  f = open("c:\\sample\\test1.csv","w")  head = "name,age,sex,phone"  line1 = "张三,29,男,11111111"  line2 = "李四,21,男,22222222"  print(head,file=f) #将head内容写入参数file指定的文件  print(line1,file=f)  print(line2,file=f)  f.close() |

写入成功的csv文件可用记事本查看，也可用Excel软件打开，打开后还可另存为xls/xlsx格式。

2. 读取test1.csv文件的内容，并在标准输出上显示。

|  |
| --- |
| #filename: sample7-1-2.py  f = open("c:\\sample\\test1.csv")  text = f.read()  print(text)  f.close() |

显示结果：

|  |
| --- |
| >>>  name,age,sex,phone  张三,29,男,11111111  李四,21,男,22222222  >>> |

### 1.2 使用CSV模块创建和读取CSV文件

除了以普通文件方式操作csv文件外，python还提供了CSV模块，可以很方便地创建、修改一个CSV文件。其中，CSV模块的reader和writer对象可用于读取和写入序列。

1. 使用CSV模块创建csv文件。

|  |
| --- |
| #filename: sample7-1-3.py  import csv  csvfile = open("c:\\sample\\test2.csv","w",newline='') #打开文件写  try:  writer = csv.writer(csvfile) #构造writer对象  writer.writerow(("name","age","sex","phone")) #写入标题行  writer.writerow(("张三","29","男","11111111")) #写入数据行  writer.writerow(("李四","21","男","22222222"))  finally:  csvfile.close() #关闭文件 |
|  |

上例中，通过csv模块的writer方法构造一个writer对象，并使用write对象的writerow方法来写入数据到文件，注意该方法的参数是一个元组或列表，写入时元组或列表中的每个元素将作为一行中的一个字段，字段间默认用逗号分隔。如要使用其他行分隔符，在构造writer对象时使用delimiter参数指定。

注意，程序在使用open函数打开文件时，需指定参数newline=''，如不指定，用writerow写入时，会添加空行。

程序运行后，就会在“c:\sample\”目录下，生成与test1.csv相同内容的test2.csv文件。

2. 使用CSV模块读取csv文件并显示。

|  |
| --- |
| #filename: sample7-1-4.py  import csv  csvfile = open("c:\\sample\\test2.csv")  lines = csv.reader(csvfile) #生成reader对象  for line in lines: #从reader对象中遍历各行  print(line) #line对应row对象，是包含行中各字段的列表  csvfile.close() |

这里，用csv模块的reader方法构造reader对象，该对象是一个包含文件各行内容的列表，类似用文件对象的readlines方法得到的列表，但与之不同的是列表中的元素不再是行对应的字符串，而是包含一行中各个字段的列表，称为row对象。程序会循环遍历reader对象中的各个low对象并在标准输出上打印。

程序运行结果：

|  |
| --- |
| ['name', 'age', 'sex', 'phone']  ['张三', '29', '男', '11111111']  ['李四', '21', '男', '22222222'] |

### 1.3 CSV文件操作综合实验

**1.问题描述**

北方甲市和南方乙市某一年1-12月的平均气温统计表如下图（数据单位为摄氏度）：

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **月份**  **城市** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **10** | **11** | **12** |
| **北方甲市** | -18 | -15 | 0 | 10 | 24 | 28 | 30 | 30 | 25 | 12 | 5 | -10 |
| **南方乙市** | 5 | 16 | 20 | 25 | 30 | 35 | 38 | 38 | 35 | 30 | 20 | 15 |

假设上表中的数据已放入两个字典变量，格式类似：

city1 = {1:-18,2:-15,3:0,4:10,5:24,6:28,7:30,8:30,9:25,10:12,11:5,12:-10}

city2 = {1:5,2:16,3:20,4:25,5:30,6:35,7:38,8:38,9:35,10:30,11:20,12:15}

使用python编程，完成以下功能：

1. 编写程序文件sy1-1.py，将这两个城市的气温数据写入一个CSV文件，文件名为city\_temp.csv，格式类似：

城市,月份1,月份2,…月份12

北方甲市,-18,-15,…,-10

南方乙市,5,16,…,15

1. 利用步骤（1）中CSV文件中的数据，编写程序文件sy1-2.py，求这两个城市的月平均最高和最低气温分别出现在几月份？分别是多少摄氏度？
2. 利用步骤（1）中CSV文件中的数据，编写程序文件sy1-3.py，求这两个城市那个月的温差最大？差多少摄氏度？
3. 假如某一农作物，生长周期7个月，要求月平均气温20摄氏度以上（含20），利用步骤（1）中CSV文件中的数据，编写程序文件sy1-4.py，判断这两个城市是否满足该农作物的生长要求。

## 2．JSON格式数据的处理

### 2.1 JSON格式数据

JSON作为一种轻量级的数据交换格式，经常被用于数据交换和输出处理。JSON的主要类型包括字符串、值、对象、和数组，JSON字符串需用双引号包括，而JSON对象类似Python的字典，由键值对的元素组成，但不同的是JSON对象中元素的键要求是字符串。

1. 一个JSON对象的例子。它包括3个元素，其中最后一个元素（序号为2）的值是一个数组。

|  |
| --- |
| {          "name":"网站",          "num":3,          "sites":["Google", "Runoob", "Taobao"]  } |

2. 一个JSON对象的例子。该JSON对象包含一个元素，键是“employees”,值是一个数组，该数组包含3个元素，每个元素均是JSON对象，其包含2个元素。

|  |
| --- |
| {"employees":[{"firstName":"John","lastName":"Doe"},  { "firstName":"Anna","lastName":"Smith"},  {"firstName":"Peter","lastName":"Jones"}]} |

### 2.2 使用json模块处理JSON格式数据

Python使用JSON模块来处理JSON格式的数据。JSON模块包含在python标准库中，不需要额外安装。使用时需用import语句载入JSON模块。

1. 使用python的json模块来处理JSON格式数据：

|  |
| --- |
| #filename: sample7-2-1.py  import json  jsonString = '{"arrayOfNums":[{"number":0},{"number":1},{"number":2}],\  "arrayOfFruits":[{"fruit":"apple"},{"fruit":"banana"},{"fruit":"pear"}]}'  jsonObj = json.loads(jsonString) #载入JSON格式字符串，返回字典对象  print(jsonObj.get("arrayOfNums")) #获取字典中key为arrayOfNums的值（该值为数组）  #获取字典中key为arrayOfNums的值（数组）的第1个元素（该元素为字典对象）  print(jsonObj.get("arrayOfNums")[1])  #获取字典中key为arrayOfNums的值（数组）的第1个元素中（字典对象）中key为"number"的值  print(jsonObj.get("arrayOfNums")[1].get("number")) |

这里loads方法返回的是一个由参数类型决定的python对象（指去除最外层引号后的实际对象类型，这里是字典对象），该对象由JSON串中的键/值对构成，可通过键名来获得相应的值。

注意，载入字符串用loads方法，载入文件则用load方法，该方法的参数是一个JSON格式的文件对象。

1. JSON文件的读取

JSON格式的文件读取有两种方法：

方法一：先将文件（文件名为stus.json，在实验素材文件夹中）读取出来到一个字符串中，再用loads方法载入字符串并生成JSON对象。

|  |
| --- |
| #filename: sample7-2-2.py  import json  f = open('stus.json', encoding='utf-8') #该JSON文件的编码格式是utf-8，指定用这种编码格式来打开文件  content = f.read() #使用loads（）方法需要先读文件  jsonObj = json.loads(content)  print(jsonObj ) #打印JSON对象（jsonObj）的字符串表示 |

方法二：直接用load方法载入文件并生成JSON对象。

|  |
| --- |
| #filename: sample7-2-3.py  import json  f = open('stus.json', encoding="utf-8")  jsonObj = json.load(f) #直接载入文件对象  print(jsonObj ) |

### 2.3 JSON格式数据处理综合实验

**1.问题描述**

北方甲市和南方乙市某一年1-12月的平均气温统计表如下图（数据单位为摄氏度）：

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **月份**  **城市** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **10** | **11** | **12** |
| **北方甲市** | -18 | -15 | 0 | 10 | 24 | 28 | 30 | 30 | 25 | 12 | 5 | -10 |
| **南方乙市** | 5 | 16 | 20 | 25 | 30 | 35 | 38 | 38 | 35 | 30 | 20 | 15 |

假设在上述1.3的实验（1）中已生成csv文件city\_temp.csv，格式类似：

城市,月份1,月份2,…月份12

北方甲市,-18,-15,…,-10

南方乙市,5,16,…,15

请编写python程序sy1-5，读取该CSV文件，将数据构造成为JSON字符串的形式并在标准输出显示，每个城市对应一个JSON字符串。

输出类似：

北方甲市：{"1": -18, "2": -15, "3": 0, "4": 10, "5": 24, "6": 28, "7": 30, "8": 30, "9": 25, "10": 12, "11": 5, "12": -10}

南方乙市：{"1": 5, "2": 16, "3": 20, "4": 25, "5": 30, "6": 35, "7": 38, "8": 38, "9": 35, "10": 30, "11": 20, "12": 15}