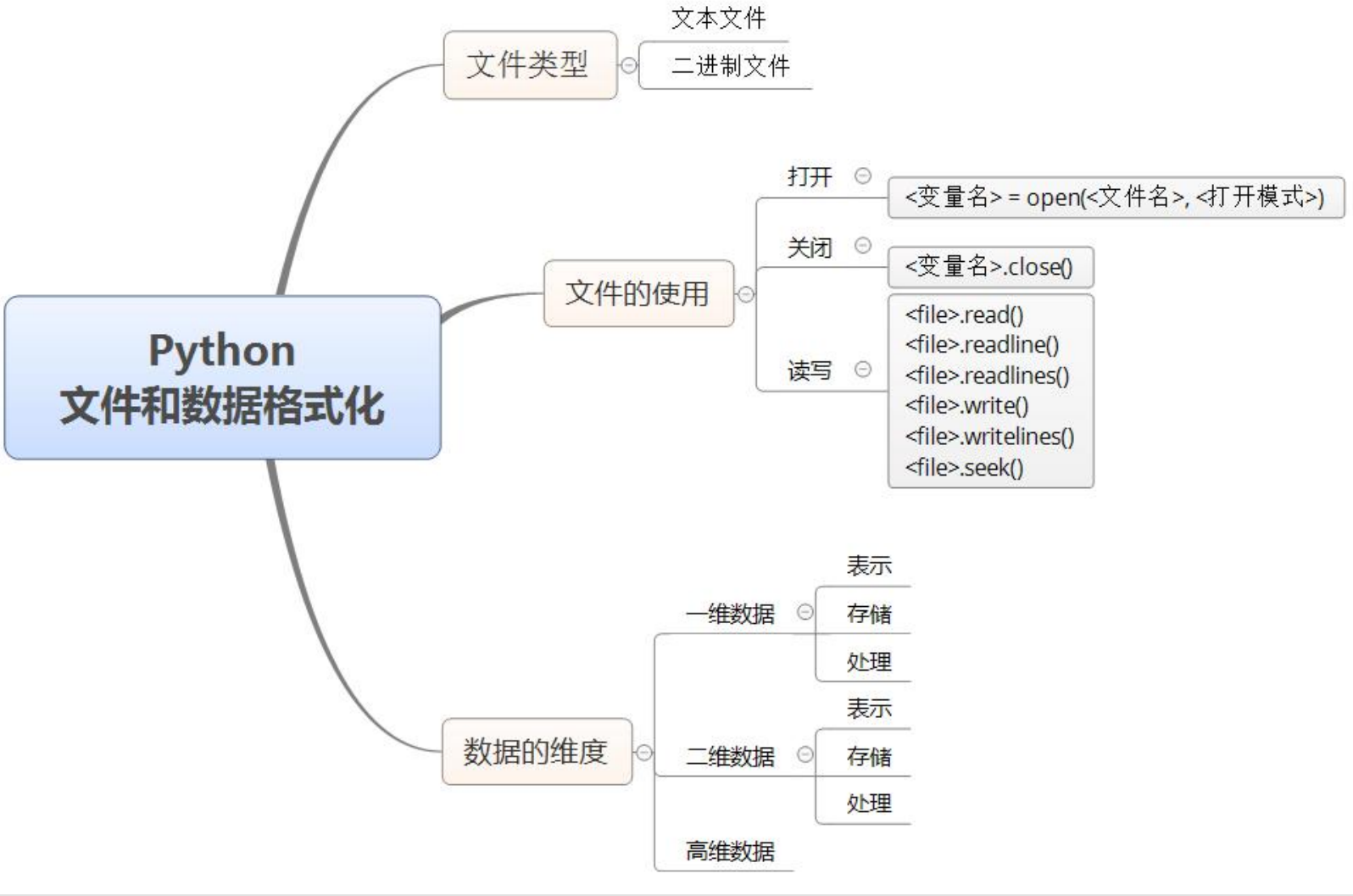
全国计算机等级考试二级教程
Python语言程序设计
(2018年版)

【第7章】
文件和数据格式化

考纲考点

- 文件的使用: 文件打开、关闭和读写
- 数据组织的维度：一维数据和二维数据
- 一维数据的处理：表示、存储和处理
- 二维数据的处理：表示、存储和处理
- 采用CSV格式对一二维数据文件的读写

知识导图


Python
文件和数据格式化
文件类型
文本文件
二进制文件
文件的使用
打开
<变量名> = open(<文件名>, <打开模式>)
关闭
<变量名>.close()
读写
<file>.read()
<file>.readline()
<file>.readlines()
<file>.write()
<file>.writelines()
<file>.seek()
数据的维度
一维数据
表示
存储
处理
二维数据
表示
存储
处理
高维数据

文件的使用

文件

- 文件是存储在辅助存储器上的一组数据序列，可以包含任何数据内容。概念上，文件是数据的集合和抽象。文件包括两种类型：**文本文件和二进制文件**。

文件的类型

- 文本文件一般由单一特定编码的字符组成，如UTF-8编码，内容容易统一展示和阅读。
- 二进制文件直接由比特0和比特1组成，文件内部数据的组织格式与文件用途有关。二进制是信息按照非字符但特定格式形成的文件，例如，png格式的图片文件、avi格式的视频文件。

文件的类型

- 二进制文件和文本文件最主要的区别在于是否有统一的字符编码。
- 无论文件创建为文本文件或者二进制文件，都可以用“文本文件方式”和“二进制文件方式”打开，但打开后的操作不同。

```
1  f = open("a.txt","rt")    #t表示文本文件方式
2  print(f.readline())
3  f.close()
```

```
>>>
全国计算机等级考试
```

文件的类型

- 文本文件a.txt，采用二进制方式打开

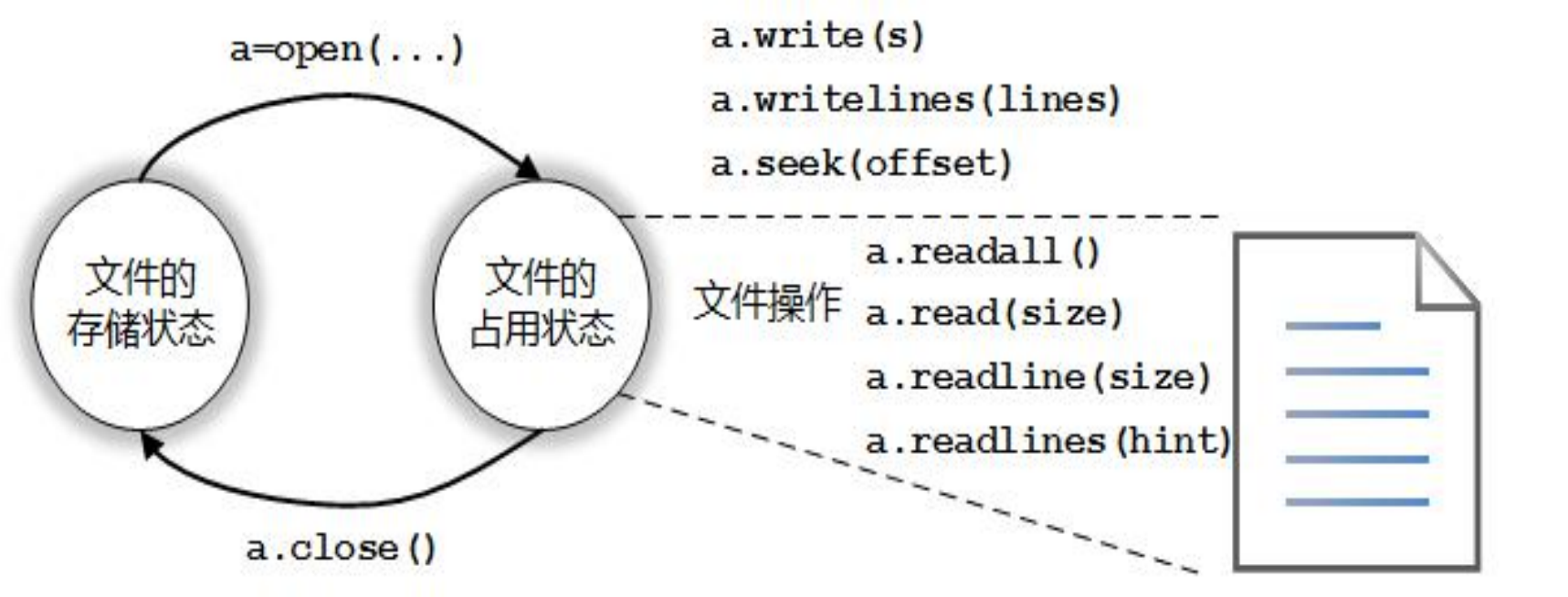
```
1  f = open("a.txt","rb")    #b表示二进制文件方式
2  print(f.readline())
3  f.close()
```

```
>>>
b'\xc8\xab\xb9\xfa\xbc\xc6\xcb\xe3\xbb\xfa\xb5\xc8\xbc\xb6
\xbf\xbc\xca\xd4'
```

- 采用文本方式读入文件，文件经过编码形成字符串，打印出有含义的字符；采用二进制方式打开文件，文件被解析为字节流。

文件的打开和关闭

- Python对文本文件和二进制文件采用统一的操作步骤，即“**打开-操作-关闭**”



文件的打开和关闭

- Python通过open()函数打开一个文件，并返回一个操作这个文件的变量，语法形式如下：

<变量名> = open(<文件路径及文件名>, <打开模式>)

打开模式	含义
'r'	只读模式，如果文件不存在，返回异常FileNotFoundError，默认值
'w'	覆盖写模式，文件不存在则创建，存在则完全覆盖源文件
'x'	创建写模式，文件不存在则创建，存在则返回异常FileExistsError
'a'	追加写模式，文件不存在则创建，存在则在原文件最后追加内容
'b'	二进制文件模式
't'	文本文件模式，默认值
'+'	与r/w/x/a一同使用，在原功能基础上增加同时读写功能

文件的打开和关闭

- 打开模式使用字符串方式表示，根据字符串定义，单引号或者双引号均可。上述打开模式中，'r'、'w'、'x'、'b'可以和'b'、't'、'+'组合使用，形成既表达读写又表达文件模式的方式。

文件的打开和关闭

- 文件使用结束后要用close()方法关闭，释放文件的使用授权，语法形式如下：

<变量名>.close()

文件的打开和关闭

- 新建一个文本文件a.txt，其内容为“全国计算机等级考试”，保存在目录PATH中，假设此时路径PATH是Windows系统的D盘根目录。打开并关闭该文件的操作过程如下。

```
>>>PATH = "D:\\"
>>>f = open(PATH + "a.txt", "rt")
>>>print(f.readline())
国家计算机等级考试
>>>f.close()
>>>print(f.readline())
Traceback (most recent call last):
  File "<pyshell#81>", line 1, in <module>
    print(f.readline())
ValueError: I/O operation on closed file.
```

文件的读写

- 根据打开方式不同，文件读写也会根据文本文件或二进制打开方式有所不同。

方法	含义
f.read(size=-1)	从文件中读入整个文件内容。参数可选，如果给出，读入前size长度的字符串或字节流
f.readline(size = -1)	从文件中读入一行内容。参数可选，如果给出，读入该行前size长度的字符串或字节流
f.readlines(hint=-1)	从文件中读入所有行，以每行为元素形成一个列表。参数可选，如果给出，读入hint行
f.seek(offset)	改变当前文件操作指针的位置，offset的值： 0：文件开头； 2: 文件结尾

文件的读写

- 如果文件不大，可以一次性将文件内容读入，保存到程序内部变量中。**f.read()**是最常用的一次性读入文件的函数，其结果是一个字符串。

```
>>>f = open("D://b.txt", "r")
>>>s = f.read()
>>>print(s)
新年都未有芳华，二月初惊见草芽。
白雪却嫌春色晚，故穿庭树作飞花。
>>>f.close()
```

文件的读写

- **f.readlines()**也是一次性读入文件的函数，其结果是一个列表，每个元素是文件的一行。

```
>>>f = open("D://b.txt", "r")
>>>ls = f.readlines()
>>>print(ls)
['新年都未有芳华，二月初惊见草芽。\n', '白雪却嫌春色晚，故穿庭树作飞花。\n']
>>>f.close()
```

文件的读写

- 文件打开后，对文件的读写有一个读取指针，当从文件中读入内容后，读取指针将向前进，再次读取的内容将从指针的新位置开始。

```
>>>f = open("D://b.txt", "r")
>>>s = f.read()
>>>print(s)
新年都未有芳华，二月初惊见草芽。
白雪却嫌春色晚，故穿庭树作飞花。
>>>ls = f.readlines()
>>>print(ls)
[]
>>>f.close()
```

文件的读写

- 结合读取指针理解，上述代码中ls返回值为空，因为之前f.read()方法已经读取了文件全部内容，读取指针在文件末尾，再次调用f.readlines()方法已经无法从当前读取指针读入内容，因此返回结果为空。

文件的读写

- **f.seek()**方法能够移动读取指针的位置，f.seek(0)将读取指针移动到文件开头，f.seek(2)将读取指针移动到文件结尾。

```
>>>f = open("D://b.txt", "r")
>>>s = f.read()
>>>print(s)
新年都未有芳华，二月初惊见草芽。
白雪却嫌春色晚，故穿庭树作飞花。
>>>f.seek(0)   # 将读取指针重置到文件开头
>>>ls = f.readlines()
>>>print(ls)
['新年都未有芳华，二月初惊见草芽。\n', '白雪却嫌春色晚，故穿庭树作飞花。\n']
>>>f.close()
```

文件的读写

- 从文本文件中逐行读入内容并进行处理是一个基本的文件操作需求。文本文件可以看成是由行组成的组合类型，因此，可以使用遍历循环逐行遍历文件，使用方法如下：

```
f = open(<文件路径及名称>, "r")
for line in f:
        # 处理一行数据
f.close()
```

文件的读写

```
1 f = open("D://b.txt", "r")
2 for line in f:
3     print(line)
4 f.close()
```

```
>>
新年都未有芳华，二月初惊见草芽。


白雪却嫌春色晚，故穿庭树作飞花。
```

文件的读写

方法	含义
f.write(s)	向文件写入一个字符串或字节流
f.writelines(lines)	将一个元素为字符串的列表写入文件

- **f.write(s)**向文件写入字符串s，每次写入后，将会记录一个写入指针。该方法可以反复调用，将在写入指针后分批写入内容，直至文件被关闭。

```
>>>f = open("D://c.txt", "w")
>>>f.write('新年都未有芳华\n')
>>>f.write('二月初惊见草芽\n')
>>>f.write('白雪却嫌春色晚\n')
>>>f.write('故穿庭树作飞花\n')
>>>f.close()
```

文件的读写

- 上述语句运行后将在D盘目录下生成一个文件c.txt，内容如下。

```
新年都未有芳华
二月初惊见草芽
白雪却嫌春色晚
故穿庭树作飞花
```

- 使用f.write(s)时，要显式的使用'\n'对写入文本进行分行，如果不进行分行，每次写入的字符串会被连接起来。

文件的读写

- **f.writelines(lines)**直接将列表类型的各元素连接起来写入文件f。

```
>>>ls = ['新年都未有芳华\n', '二月初惊见草芽\n','白雪却嫌春色晚
\n','故穿庭树作飞花\n']
>>>f = open("D://c.txt", "w")
>>>f.writelines(ls)
>>>f.close()
```

数据组织的维度

数据组织的维度

- 一组数据在被计算机处理前需要进行一定的组织，表明数据之间的基本关系和逻辑，进而形成“数据的维度”。根据数据的关系不同，数据组织可以分为：**一维数据、二维数据和高维数据**。

一维数据

- 一维数据由对等关系的有序或无序数据构成，采用**线性方式组织**，对应于数学中数组的概念。例如：中国的直辖市列表即可表示为一维数据，一维数据具有线性特点。

北京、上海、天津、重庆

二维数据

- 二维数据，也称表格数据，由关联关系数据构成，采用二维表格方式组织，对应于数学中的矩阵，常见的表格都属于二维数据。
- 例如：国家统计局发布的居民消费价格指数是二维数据

二维数据

指标	2014年	2015年	2016年
居民消费价格指数	102	101.4	102
食品	103.1	102.3	104.6
烟酒及用品	99.4	102.1	101.5
衣着	102.4	102.7	101.4
家庭设备用品	101.2	101	100.5
医疗保健和个人用品	101.3	102	101.1
交通和通信	99.9	98.3	98.7
娱乐教育文化	101.9	101.4	101.6
居住	102	100.7	101.6

每个数据为相比上年数据的标准值，即上年指标为100。

高维数据

- 高维数据由键值对类型的数据构成，采用对象方式组织，可以多层嵌套。
- 高维数据在Web系统中十分常用，作为当今Internet组织内容的主要方式，高位数据衍生出HTML、XML、JSON等具体数据组织的语法结构。

高维数据

```
"本书" : [
            "第1章" : "程序设计基本方法",
            "第2章" : "Python语言基本语法元素",
            "第3章" : "基本数据类型",
            "第4章" : "程序的控制结构",
            "第5章" : "函数和代码复用",
            "第6章" : "组合数据类型",
            "第7章" : "文件和数据格式化",
            "第8章" : "Python计算生态",
            "第9章" : "Python标准库概览",
            "第10章" : "Python第三方库概览",
            "第11章" : "Python第三方库纵览",
            "第12章" : "考试指导",
            "附录" : "附录1234567"
        ]
```

一维数据的处理

一维数据的表示

- 一维数据是最简单的数据组织类型，由于是线性结构，在Python语言中主要采用列表形式表示。例如：中国的直辖市数据可以采用一个列表变量表示。

```
>>>ls = ['北京', '上海', '天津', '重庆']
>>>print(ls)
['北京', '上海', '天津', '重庆']
```

一维数据的存储

- 一维数据的文件存储有多种方式，总体思路是采用特殊字符分隔各数据。常用存储方法包括4种。
- （1）采用空格分隔元素，例如：

  北京　上海　天津　重庆
- （2）采用逗号分隔元素，例如：

  北京,上海,天津,重庆

一维数据的存储

- （3）采用换行分隔包括，例如：

北京

上海

天津

重庆

- （4）其他特殊符号分隔，以分号分隔为例，例如：

北京;上海;天津;重庆

一维数据的存储

- 逗号分割的存储格式叫做**CSV格式**（Comma-Separated Values，即逗号分隔值），它是一种通用的、相对简单的文件格式，在商业和科学上广泛应用，大部分编辑器都支持直接读入或保存文件为CSV格式
- 一维数据保存成CSV格式后，各元素采用逗号分隔，形成一行。从Python表示到数据存储，需要将列表对象输出为CSV格式以及将CSV格式读入成列表对象

一维数据的存储

- 列表对象输出为CSV格式文件方法如下，采用字符串的join()方法最为方便。

```
ls = ['北京', '上海', '天津', '重庆']
f = open("city.csv", "w")
f.write(",".join(ls)+ "\n")
f.close()
```

北京,上海,天津,重庆

一维数据的处理

- 对一维数据进行处理首先需要从CSV格式文件读入一维数据，并将其表示为列表对象。

```
1 f = open("city.csv", "r")
2 ls = f.read().strip('\n').split(",")
3 f.close()
4 print(ls)
```

```
>>>
['北京', '上海', '天津', '重庆']
```

二维数据的处理

- 二维数据由多条一维数据构成，可以看成是一维数据的组合形式。因此，二维数据可以采用二维列表来表示，即列表的每个元素对应二维数据的一行，这个元素本身也是列表类型，其内部各元素对应这行中的各列值

二维数据的处理

```
ls = [
        ['指标', '2014年', '2015年', '2016年'],
        ['居民消费价格指数', '102', '101.4', '102'],
        ['食品', '103.1', '102.3', '104.6'],
        ['烟酒及用品', '994', '102.1', '101.5'],
        ['衣着', '102.4', '102.7', '101.4'],
        ['家庭设备用品', '101.2', '101', '100.5'],
        ['医疗保健和个人用品', '101.3', '102', '101.1'],
        ['交通和通信', '99.9', '98.3', '98.7'],
        ['娱乐教育文化', '101.9', '101.4', '101.6'],
        ['居住', '102', '100.7', '101.6'],
    ]
```

二维数据的存储

- 二维数据由一维数据组成，用CSV格式文件存储。CSV文件的每一行是一维数据，整个CSV文件是一个二维数据。
- 二维列表对象输出为CSV格式文件方法如下，采用遍历循环和字符串的**join()方法**相结合。

```
# ls代表二维列表，此处省略
f = open("cpi.csv", "w")
for row in ls:
    f.write(",".join(row)+ "\n")
f.close()
```

二维数据的处理

- 对二维数据进行处理首先需要从CSV格式文件读入二维数据，并将其表示为二维列表对象。借鉴一维数据读取方法，从CSV文件读入数据的方法如下。

```
1  f = open("cpi.csv", "r")
2  ls = []
3  for line in f:
4      ls.append(line.strip('\n').split(","))
5  f.close()
6  print(ls)
```

二维数据的处理

- 程序执行后二维列表对象ls的内容如下。

```
>>>
[['指标', '2014年', '2015年', '2016年'], ['居民消费价格指数', '102', '101.4', '102'], ['食品', '103.1', '102.3', '104.6'], ['烟酒及用品', '994', '102.1', '101.5'], ['衣着', '102.4', '102.7', '101.4'], ['家庭设备用品', '101.2', '101', '100.5'], ['医疗保健和个人用品', '101.3', '102', '101.1'], ['交通和通信', '99.9', '98.3', '98.7'], ['娱乐教育文化', '101.9', '101.4', '101.6'], ['居住', '102', '100.7', '101.6']]
```

二维数据的处理

- 二维数据处理等同于二维列表的操作，与一维列表不同，二维列表一般需要借助循环遍历实现对每个数据的处理，基本代码格式如下：

```
for row in ls:
    for item in row:
        <对第row行第item列元素进行处理>
```

二维数据的处理

- 对二维数据进行格式化输出，打印成表格形状

```
1  # 此处略去从CSV获取数据到二维列表ls
2  for row in ls:
3      line = ""
4      for item in row:
5          line += "{:10}\t".format(item)
6      print(line)
```

```
>>>
指标                2014年          2015年          2016年
居民消费价格指数        102             101.4           102
食品                103.1           102.3           104.6
烟酒及用品           994             102.1           101.5
衣着                102.4           102.7           101.4
家庭设备用品          101.2           101             100.5
医疗保健和个人用品     101.3           102             101.1
交通和通信           99.9            98.3            98.7
娱乐教育文化          101.9           101.4           101.6
居住                 102             100.7           101.6
```

实例解析：国家财政数据趋势演算

国家财政数据趋势演算

- 国家统计局每年会公开许多数据，比如国民经济核算指标等。国家统计局公布的大部分数据都以二维表格形式展现，然而，藏在这些数据背后的价值要比表格所展现的更多。

国家财政数据趋势演算

- 以国家财政收支的公开数据为例，这里展示如何利用Python挖掘数据变化的规律。将从网上获取的公开信息存为finance.csv文件

指标	2000年	2015年	2016年
全部收入	13395.2	152269.2	159605
中央收入	6989.2	69267.2	72365.6
地方收入	6406.1	83002	87239.4
全部支出	15886.5	175877.8	187755.2
中央支出	5519.9	25542.2	27403.9
地方支出	10366.7	150335.6	160351.4

国家财政数据趋势演算

- 由个别数据预测规律属于数值分析的内容，可以通过线性回归方程建立简单模型，线性回归方程的公式为：$\hat{y} = b\hat{x} + a.$

- X代表年份，Y代表各年份对应的数值。Python实现的国家财政数据趋势演算，根据上述三个数值计算出更多年份的可能数据。

国家财政数据趋势演算

实例7.1

```
# FinancePredict.py
def parseCSV(filename):
    dataNames, data = [], []
    f = open(filename, 'r', encoding='utf-8')
    for line in f:
        splitedLine = line.strip().split(',')
        if '指标' in splitedLine[0]:
            years = [int(x[:-1]) for x in splitedLine[1:]]
        else:
            dataNames.append('{:10}'.format(splitedLine[0]))
            data.append([float(x) for x in splitedLine[1:]])
    f.close()
    return years, dataNames, data

def means(data):
    return sum(data) / len(data)

def linearRegression(xlist, ylist):
    xmeans, ymeans = means(xlist), means(ylist)
    bNumerator = - len(xlist) * xmeans * ymeans
    bDenominator =  - len(xlist) * xmeans ** 2
    for x, y in zip(xlist, ylist):
        bNumerator += x * y
        bDenominator += x ** 2
    b = bNumerator / bDenominator
    a = ymeans - b * xmeans
    return a, b
```

国家财政数据趋势演算

实例7.1

```
28
29  def calNewData(newyears, a, b):
30      return [(a + b * x) for x in newyears]
31
32  def showResults(years, dataNames, newDatas):
33      print('{:^60}'.format('国家财政收支线性估计'))
34      header = '指标        '
35      for year in years:
36          header += '{:10}'.format(year)
37      print(header)
38      for name, lineData in zip(dataNames, newDatas):
39          line = name
40          for data in lineData:
41              line += '{:>10.1f}'.format(data)
42          print(line)
43
44  def main():
45      newyears = [x+2010 for x in range(7)]
46      newDatas = []
47      years, dataNames, datas = parseCSV('finance.csv')
48      for data in datas:
49          a, b = linearRegression(years, data)
50          newDatas.append(calNewData(newyears, a, b))
51      showResults(newyears, dataNames, newDatas)
52
53  main()
```

国家财政数据趋势演算

```
>>>
```

国家财政收支线性估计

指标	2010	2011	2012	2013	2014	2015	2016
全部收入	105359.6	114550.1	123740.6	132931.0	142121.5	151312.0	160502.4
中央收入	48169.1	52283.8	56398.5	60513.2	64627.9	68742.7	72857.4
地方收入	57190.6	62266.3	67342.1	72417.8	77493.6	82569.3	87645.1
全部支出	122936.9	133645.7	144354.5	155063.3	165772.1	176480.9	187189.8
中央支出	19037.5	20390.9	21744.3	23097.7	24451.1	25804.5	27157.9
地方支出	103899.4	113254.8	122610.2	131965.6	141321.0	150676.4	160031.9

本章小结

本章讲解了文件的基本使用方法，包括文件的打开、关闭、读取和写入。进一步围绕数据的维度，讲解了一维数据、二维数据和高维数据的概念，以及一二维数据的表示、存储和处理方法。通过国家财政数据趋势演算的实例帮助读者理解数据处理的基本方法。

一二维数据是社会生活中数据运用的绝对大多数，请思考一下，方圆十米范围内有哪些一二维数据亟待程序的处理？！