学习目标：

1. 掌握文件的读写方法以及打开和关闭等基本操作
2. 理解数据组织的维度和特点
3. 掌握一二维数据的存储格式和读写方法
4. 运用PIL库进行基本的图像处理
5. 运用json库进行数据的维度转换
6. 了解高维数据的存储格式和读写方法

第七章 文件和数据格式化

目录

CONTENTS

7.1 文件的使用

- 打开文件用open函数，本例中有3个参数，参数1是文件名，参数2是打开方式(r表示读取，t表示文本文件，b表示二进制文件)，参数三为文件的编码格式。
- Readline()函数表示从文件中读取一行

文件概述

- 文件：存储在存储器上的数据序列；
- 文本文件：由单一的特定编码的字符组成；容易展示和阅读，如word，notepad等都能方便的阅读文本文件；
- 二进制文件：直接由0和1组成、没有统一字符编码，如图片、声音、视频等文件。

微实例7.1 理解文本文件和二进制文件的区别

```
textfile=open("7.1.txt","rt",encoding="utf-8")
print(textfile.readline())
textfile.close()
binfile=open("7.1.txt","rb")
print(binfile.readline())
binfile.close()
```

输出结果：

```
中国是个伟大的国家
b'\xe4\xb8\xad\xe5\x9b\xbd\xe6\x98\xaf\xe4\xb8\xaa\xe4\xbc\x9f\xe5\xa4\xa7\xe7\x9a\x84\xe5\x9b\xbd\xe5\xae\xb6'
```

7.1 文件的使用

文件的打开和关闭

1.文件的打开：<变量名>=open(<文件名>,<打开模式>)

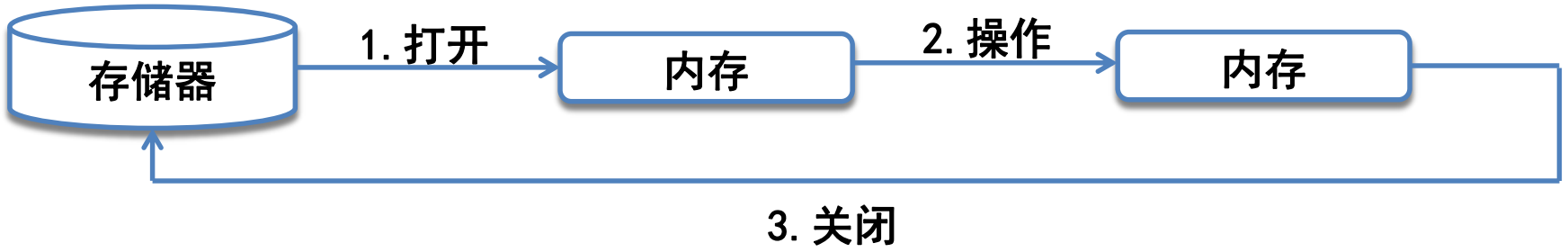
open(file, mode='r', buffering=-1, encoding=None, errors=None, newline=None, closefd=True, opener=None)：在使用该函数的时候，除了file参数必填外，其他参数可以选用，否则为默认值。

打开模式	含义
r	只读模式，如果文件不存在，返回异常FileNotFoundError，为默认值
w	覆盖写模式，文件不存在则创建，存在则覆盖
x	创建写，文件不存在则创建，存在则返回异常FileExistsError
a	追加写，文件不存在则创建，存在则在文件的最后追加内容
b	二进制文件模式
t	文本文件模式，为默认值
+	与r/w/x/a配合使用，在原有功能基础上增加同时读写功能

模式	描述
r	以只读方式打开文件。文件的指针将会放在文件的开头。这是默认模式。
rb	以二进制格式打开一个文件用于只读。文件指针将会放在文件的开头。这是默认模式。
r+	打开一个文件用于读写。文件指针将会放在文件的开头。
rb+	以二进制格式打开一个文件用于读写。文件指针将会放在文件的开头。
w	打开一个文件只用于写入。如果该文件已存在则打开文件，并从开头开始编辑，即原有内容会被删除。如果该文件不存在，创建新文件。
wb	以二进制格式打开一个文件只用于写入。如果该文件已存在则打开文件，并从开头开始编辑，即原有内容会被删除。如果该文件不存在，创建新文件。
w+	打开一个文件用于读写。如果该文件已存在则打开文件，并从开头开始编辑，即原有内容会被删除。如果该文件不存在，创建新文件。
wb+	以二进制格式打开一个文件用于读写。如果该文件已存在则打开文件，并从开头开始编辑，即原有内容会被删除。如果该文件不存在，创建新文件。
a	打开一个文件用于追加。如果该文件已存在，文件指针将会放在文件的结尾。也就是说，新的内容将会被写入到已有内容之后。如果该文件不存在，创建新文件进行写入。
ab	以二进制格式打开一个文件用于追加。如果该文件已存在，文件指针将会放在文件的结尾。也就是说，新的内容将会被写入到已有内容之后。如果该文件不存在，创建新文件进行写入。
a+	打开一个文件用于读写。如果该文件已存在，文件指针将会放在文件的结尾。文件打开时会是追加模式。如果该文件不存在，创建新文件用于读写。
ab+	以二进制格式打开一个文件用于追加。如果该文件已存在，文件指针将会放在文件的结尾。如果该文件不存在，创建新文件用于读写。

7.1 文件的使用

文件的打开和关闭

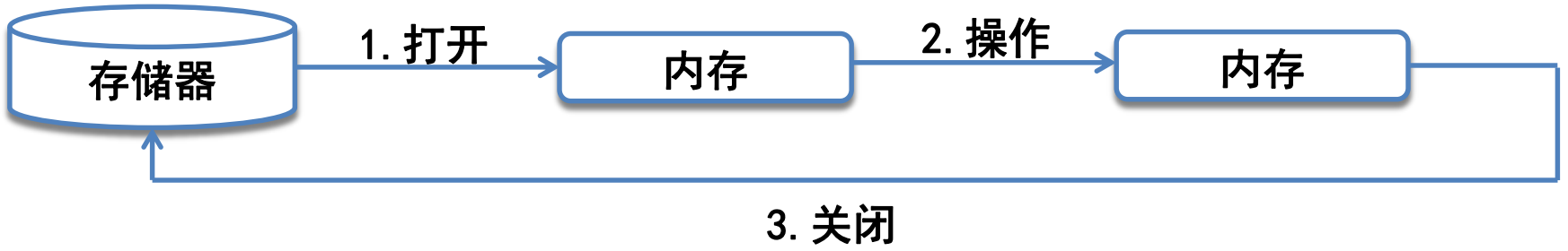


3.文件的关闭：<变量名>.close()

将操作后的文件从内存移动到存储器中。

7.1 文件的使用

文件的读写



2.文件读写：当文件以文本方式打开时，按照字符串方式读写，当以二进制文件方式打开时，按照字节流的方式读写。

文件内容的读取方法：

操作方法	含义
file.readall()	读入整个文件内容，返回一个字符串或字节流
file.read(size=-1)	从文件中读取整个文件的内容，如果给出参数，读入前size长度的字符或字节流
file.readline(size=-1)	从文件中读入一行内容，如果给出参数，读入该行前szie长度的字符或字节流
file.readlines(hint=-1)	从文件中读入所有行，以每行为元素形成一个列表，如果给出参数，读入hint行

7.1 文件的使用

文件的读写

微实例7.2 文本文件逐行打印

```
fname=input("请输入要打开的文件：")
fo=open(fname,"r")
for line in fo.readlines():
    print(line)
fo.close()
```

- fname存储从键盘输入的文件名，如果不包含路径，则为当前路径中的文件
- fo.open(fname,”r“)：以只读的方式打开对应文件，
- fo.readlines():读取fo所指文件的所有行，生成列表;
- for-in语句依次从列表中选择元素并输出。
- fo.close()关闭文件。

缺点：当文件很大时，内存占用比较大，影响程序的执行速度，

改进：由于fo本身作为一个行序列，可以改为如下程序

```
fname=input("请输入要打开的文件：")
fo=open(fname,"r")
for line in fo:
    print(line)
fo.close()
```

7.1 文件的使用

文件的读写

逐行写入方法：

操作方法	含义
file.write(s)	像文件写入一个字符或字节流
file.writelines(lines)	将一个元素全为字符串的列表写入文件
file.seek(offset)	定位文件当前位置，offset的取值：0-文件开头，1-当前位置，2-文件结尾

微实例7.3：

```
fname=input("请输入要写入的文件： ")
fo=open(fname,"w+")
ls=["飞雪连天射白路","笑书神侠倚碧鸳"]
fo.writelines(ls)
for line in fo:
    print(line)
fo.close()
```

程序运行结果：
虽然将内容写入到了文件中，但文件内容没有输出，原因在于当文件写入完成时，当前位置定位到文件的结尾，没有需要输出的内容。
可以在for-in语句前增加：
fo.seek(0)
可解决上述问题。

7.2 模块5：PIL库的使用

PIL库概述

PIL是python处理图像处理的的第三方库，需要通过pip等方法引入（对应的安装库的名字是pillow），PIL库主要实现图像的归档和图像处理两个功能：

- **图像归档：对图像进行批处理、生成图像预览、图像格式转换等；**
- **图像处理：图像的基本处理、像素处理、颜色处理等。**

PIL包含21个与图片有关的子类：

Image、ImageChops、ImageColor、ImageCrackCode、ImageDraw、ImageEnhance、ImageFile、ImageFileIO、ImageFilter、ImageFont、ImageGL、ImageGrab、Imagemath、ImageOps、ImagePalette、ImagePath、ImageQt、ImageSequence、ImageStat、ImageTk、ImageWin

7.2 模块5：PIL库的使用

PIL库Image类解析

Image类是PIL最重要的类，引入方法为：

Import PIL　　#引入PIL中的所有类

From PIL import Image　#单独引入Image类

任何一个图像文件都可以用Image对象表示，Image类的图像读取和创建方法如下：

方法	描述
Image.open(filename)	根据参数加载图像文件
Image.new(mode,size,color)	根据给定参数创建一个新图像
Image.open(StringIO.StringIO(buffer))	从字符串中获取图像
Image.frombytes(mode,size,data)	根据像素点data创建图像
Image.verify()	对图像文件完整性进行检查，返回异常

加载图像的方法：

```
from PIL import Image
im=Image.open("bird.jpg")
```

7.2 模块5：PIL库的使用

PIL库Image类解析

Image类的4个处理图片的常用属性：

属性	描述
Image.format	标识图像格式和来源，如果图像不是从文件读取，值为None
Image.mode	图像的色彩模式，”L”为灰度图像，“RGB”为真彩色图像，“CMYK”为出版图像
Image.size	图像的宽度和高度，单位为像素(px)，返回值为二元元组
Image.pallette	调色板属性，返回一个ImagePallete类型

Image还能读取序列图像，包括GIF、FLI、FLC、TIFF等格式，open()方法打开时自动加载序列中的第一帧，使用seek()和tell()方法可以在不同帧之间移动

Image.seek(frame):跳转并返回图像中的指定帧
Image.tell():返回当前帧的序号

7.2 模块5：PIL库的使用

微实例：GIF文件图像提取

```
from PIL import Image
im=Image.open('pybit.gif')
try:
    im.save('picframe{:02d}.png'.format(im.tell()))
    while True:
        im.seek(im.tell()+1)
        im.save('picframe{:02d}.png'.format(im.tell()))
except:
    print("处理结束")
```

```
from PIL import Image
im=Image.open("picframe22.png")
images=[]
for i in range(21):
    images.append(Image.open('picframe{:02d}.png'.format(21-i)))
im.save('all.gif', save_all=True,
append_images=images,loop=0,duration=0,comment=b"aaabb")
```

7.2 模块5：PIL库的使用

PIL库Image类解析

Image类的图像转换和保存方法：

方法	描述
Image.save(filename,format)	将图像保存为filename文件，format是图片格式
Image.convert(mode)	使用不同的参数，转换图像为新的模式
Image.thumbnail(size)	创建图像的缩略图，size是缩略图尺寸的二元元组

Image类的图像旋转和缩放：

方法	描述
Image.resize(size)	按size大小调整图像，生成副本
Image.rotate(angle)	按angle角度逆时针旋转图像，生成副本

```
Im=Image.open(“birdnest.jpg”)
Im.thumbnail((128,128))
Im.save(“birdnesttn”,”jpeg”)
```

7.2 模块5：PIL库的使用

Image类的图像像素和通道处理方法：

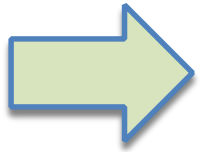
方法	描述
Image.point(func)	根据函数func的功能对每个元素进行运算，返回图像副本
Image.split()	提取RGB图像的每个颜色通道，返回图像副本
Image.merge(mode,bands)	合并通道，其中mode表示色彩，bands表示新的色彩通道
Image.blend(im1,im2,alpha)	将两幅图片im1和im2按照如下公式插值后生成新的图像：im1*(1.0-alpha)+im2*alpha

微实例7.5：图像的颜色交换

```
from PIL import Image
im=Image.open('bird.jpg')
r,g,b=im.split()
om=Image.merge("RGB",(b,g,r))
om.save("birdnestbgr.jpg")
```

7.2 模块5：PIL库的使用

```
from PIL import Image
im=Image.open('bird.jpg')                    #打开鸟巢文件
r,g,b=im.split()                             #获得RGB通道
newg=g.point(lambda i:i*0.9)                 #将G通道颜色变为0.9倍
newb=b.point(lambda i:i<100)                 #选择B通道低于100的像素点
om=Image.merge(im.mode,(r,newg,newb))        #将3个通道合成为新图像
om.save("birdmerge.jpg")                     #输出图像
```



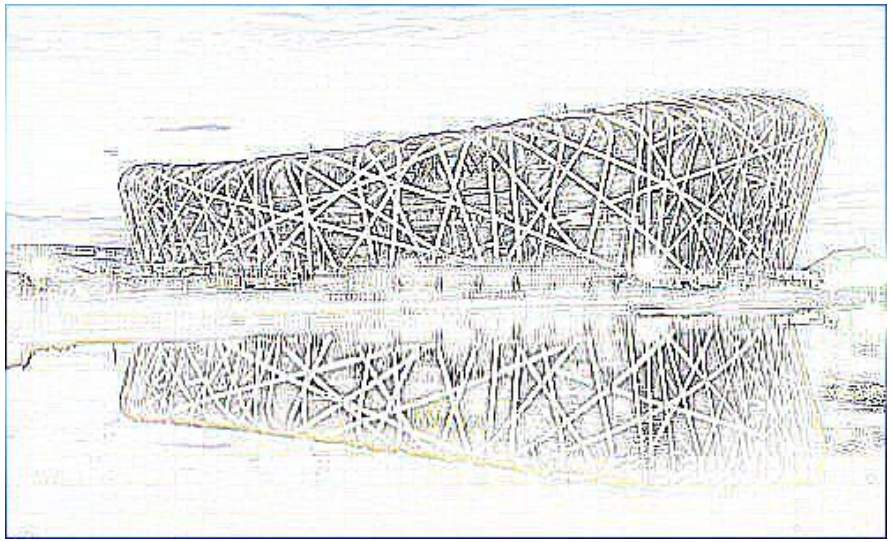
图像的过滤和增强

PIL的ImageFilter类提供了10种图像过滤方法。

方法	描述
ImageFilter.BLUR	图像的模糊效果
ImageFilter.CONTOUR	图像的轮廓效果
ImageFilter.DEFAULT	图像的细节效果
ImageFilter.EDGE_ENHANCE	图像的边界加强效果
ImageFilter.EDGE_ENHANCE_MORE	图像的阈值加强效果
ImageFilter.EMBOSS	图像的浮雕效果
ImageFilter.FIND_EDGES	图像的边界效果
ImageFilter.SMOOTH	图像的平滑效果
ImageFilter.SMOOTH_MORE	图像的阈值平滑效果
ImageFilter.SHARPEN	图像的锐化效果

利用Image类的filter()方法可以使用ImageFilter类，格式：Image.filter(ImageFilter.function)

```
from PIL import Image
from PIL import ImageFilter
im=Image.open('bird.jpg')
om=im.filter(ImageFilter.CONTOUR)
om.save("birdcontour.jpg")
```



7.2 模块5：PIL库的使用

图像的过滤和增强

PIL的ImageEnhance类提供了5种图像增强和滤镜方法。

方法	描述
ImageEnhance.enhance(factor)	对选择属性的数值增强factor倍
ImageEnhance.Color(im)	调整图像的颜色平衡
ImageEnhance.Contrast(im)	调整图像的对比度
ImageEnhance.Brightness(im)	调整图像的亮度
ImageEnhance.Sharpness(im)	调整图像的锐度

```
from PIL import Image
from PIL import ImageEnhance
im=Image.open('birdnest.jpg')
om=ImageEnhance.Contrast(im)
om.enhance(20).save("birdcontrast.jpg")
```

7.3 实例12：图像的字符画绘制

位图图片是有不同颜色像素点组成的规则分布，如果采用字符代替像素，图像就变成了字符画。

首先定义一个字符集，将字符集中的字符代替图像中的像素点，使得每个字符对应图像中的不同颜色。

```
ascii_char=list("$%_&WM#*oahkbdpqwmZOQLCJUYXzcvunxrjft/\|()1{}[]?-/+@<>!;:\^.')
```

将彩色图像转换为0到255的灰度图像，灰度值从小到大依次对应字符集中的符号，对于每个像素，其运算公式为：

$$gray=0.2126*r+0.7152*g+0.0722*b$$

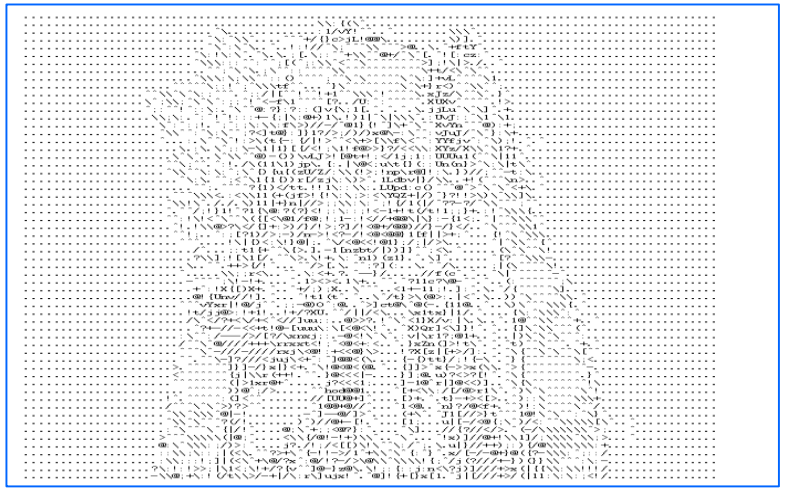
像素的RGB颜色与字符集的对应函数如下：

```
def get_char(r,b,g,alpha=256):
    if alpha==0:
        return ' '
    gray=int(0.2126*r+0.7152*g+0.0722*b)
    unit=256/len(ascii_char)
    return ascii_char[int(gray//unit)]
```

7.3 实例12：图像的字符画绘制

```
from PIL import Image
ascii_char=list("$%_&WM#*oahkbdpqwmZOQLCJUYXzcvunxrjft/\|()1{}[]?-/+@<>!;:\^.")
def get_char(r,b,g,alpha=256):
    if alpha==0:
        return ' '
    gray=int(0.2126*r+0.7152*g+0.0722*b)
    unit=256/len(ascii_char)
    return ascii_char[int(gray//unit)]
```

```
def main():
    im=Image.open('astro.jpg')
    WIDTH,HEIGHT=100,60
    im=im.resize((WIDTH,HEIGHT))
    txt=""
    for i in range(HEIGHT):
        for j in range(WIDTH):
            txt+=get_char(*im.getpixel((j,i)))
        txt+='\n'
    fo=open('pic_char.txt',"w")
    fo.write(txt)
    fo.close()
main()
```



实验7.1 将实例10的统计的《三国演义》的人物的结果写入文本文件result.txt中

实验7.2 将ujn1.jpg灰度化为ujn_gray.jpg

2、将图像ujn1.jpg缩小为原来的0.5倍

3、将图像依次旋转10度、20度、30度、...350度、360度，然后将36幅图像生成一个rota.gif；

4、将ujn1.jpg嵌入ujn2.jpg上（比例为8:2）：

5、将图像Monalisa.jpg 转换为字符画，存入Monalisa.txt,各灰度与字符的对应如下表所示：

0	1	2	3	4	5	6	7	8	9	10	11	12
a	b	c	d	e	f	g	h	i	j	k	l	m
13	14	15	16	17	18	19	20	21	22	23	24	25
n	o	p	q	r	s	t	u	v	w	x	y	z

26	27	28	29	…	…	…	…	…	…	253	254	255
a	b	c	d	…	…	…	…	…	…	v	w	x

7.4 一二维数据的格式化和处理

数据组织的维度

一维数据由对等关系的有序或无序数据组成，采用线性方式组织，对应于数字中的数组和集合等概念，

二维数据，也称为表格数据，由关联关系数据构成，采用表格方式组织，对应数学中的矩阵。

一维数据：

北京大学	清华大学	武汉大学	复旦大学	浙江大学

二维数据：

序号	*学校*	*地区*	*类型*	*评分*
1	北京大学	北京	综合	100
2	清华大学	北京	理工	96.91
3	武汉大学	湖北	综合	82.57
4	复旦大学	上海	综合	82.47
5	浙江大学	浙江	综合	82.31

7.4 一二维数据的格式化和处理

数据组织的维度

高维数据由键值对数据组成，采用对象方式组织，数据整合度更好的数据组织方法。如HTML、XML、JSON等都是高维数据组织的语法结构。

```
“本书作者”：
[
        {“姓氏”：“嵩”，
         “名字”：“天”，
         “单位”：“北京理工大学”}，
        {“姓氏”：“礼”，
         “名字”：“欣”，
         “单位”：“北京理工大学”}，
        {“姓氏”：“黄”，
         “名字”：“天宇”，
         “单位”：“北京理工大学”}，
]
```

7.4 一二维数据的格式化和处理

一二维数据的存储格式

一维数据有多种存储格式，常用特殊符号分割，如：

- **用一个或多个空格分隔：**

中国 美国 日本 德国 法国 英国 意大利

- **用逗号分隔:**

中国,美国,日本,德国,法国,英国,意大利

- **用其它特殊符号或符号组合分割：**

中国；美国；日本；德国；法国；英国；意大利

7.4 一二维数据的格式化和处理

一二维数据的存储格式

二维数由多条一维数据组成，如CSV格式，如：

逗号分隔数据的存储格式叫做CSV（Comma-Separated Values）格式，是一种通用的、相对简单的文件格式，经常用于在程序之间转义表格数据，有如下规则：

- **纯文本格式，通过单一编码表示字符；**
- **以行为单位，开头不留空行，行之间没有空行；**
- **每行表示一个一维数据，多行表示二维数据；**
- **以逗号（英文，半角）分割每列数据，列数据为空也要保留逗号；**
- **对于表格数据，可以包含或不包含列名，包含时列名放置在文件第一行；**

序号，学校，地区，类型，评分
1，北京大学，北京，综合，100
2，清华大学，北京，理工，96.91
3，武汉大学，湖北，综合，82.57
4，复旦大学，上海，综合，82.47
5，浙江大学，浙江，综合，82.31

7.4 一二维数据的格式化和处理

一二维数据的表示和读写

CSV文件每行是一维数据，在Python中使用列表表示，如：

```
[
  [“序号”，” 学校”，” 地区”，” 类型”，” 评分\n”],
  [“1”，“北京大学”，“北京”，“综合”，“100\n”],
  [“2”，“清华大学”，“北京”，“理工”，“96.91\n”],
  [“3”，“武汉大学”，“湖北”，” 综合”，” 82.57\n”],
  [“4”，” 复旦大学”，“上海”，“综合”，“82.47\n”],
  [“5”，“浙江大学”，“浙江”，“综合”，“82.31\n”]
]
```

7.4 一二维数据的格式化和处理

微实例7.8 导入CSV格式数据到列表

```
fo=open(“price2016.csv”,”r”)
ls=[]
for line in fo:
        line=line.repalce(“\n”,“”)
        ls.append(line.split(“,”))
print(ls)
fo.close()
```

将每一行末尾的换行符‘\n’替换为空字符串。

将每行逗号分隔的文本分割后加入ls列表中。

```
[['城市', '环比', '同比', '定基'], ['北京', '101.5', '120.7',
'121.4'], ['上海', '101.2', '127.3', '127.8'], ['广州', '101.3',
'119.4', '120'], ['深圳', '102', '140.9', '145.5'], ['沈阳',
'100.1', '101.4', '101.6']]
```

7.4 一二维数据的格式化和处理

逐行处理（读取）CSV格式数据：

```
fo=open("price2016.csv","r"
ls=[]
for line in fo:
    line=line.replace("\n","")
    ls=line.split(",")
    lns=""
    for s in ls:
        lns+="{}\t".format(s)
    print(lns)
fo.close()
```

运行结果如下：

城市	环比	同比	定基
北京	101.5	120.7	121.4
上海	101.2	127.3	127.8
广州	101.3	119.4	120
深圳	102	140.9	145.5
沈阳	100.1	101.4	101.6

7.4 一二维数据的格式化和处理

一维数据写入CSV文件

将line行数据末尾的“\n”用“”替换后重新存入line

将line中以“,”分隔的数据存入ls列表中

“,”.join(row)+”\n”表示将列表元素row的元素以“,”分隔

```
fr=open("price2016.csv","r")
fw=open("price2016out.csv","w")
ls=[]
for line in fr:
    line=line.replace("\n","")
    ls.append(line.split(","))
for i in range(len(ls)):
    for j in range(len(ls[i])):
        if ls[i][j].replace(".","").isnumeric():
            ls[i][j]="{:.2}%".format(float(ls[i][j])/100)
for row in ls:
    print(row)
    fw.write(",".join(row)+"\n")
fr.close()
fw.close()
```

如果元素中将“.”替换为“”后为整数，则将该数据表示成百分数

运行结果如下：

```
城市,环比,同比,定基
北京,1.0%,1.2%,1.2%
上海,1.0%,1.3%,1.3%
广州,1.0%,1.2%,1.2%
深圳,1.0%,1.4%,1.5%
沈阳,1.0%,1.0%,1.0%
```

7.6 用excel生成如表1所示的文件，保存为example.csv,然后读取该文件，生成一个example_2.csv,如表2所示。

姓名	学号	数学	语文	英语
张三锋	170101	76	90	88
李四娘	170102	83	86	87
王五指	170103	92	79	86
赵六子	170104	65	61	85
钱多多	170105	59	93	84

表1 example.csv

姓名	学号	数学	语文	英语	总分
张三锋	170101	76	90	88	254
李四娘	170102	83	86	87	256
王五指	170103	92	79	86	257
赵六子	170104	65	61	85	211
钱多多	170105	59	93	84	236

表2 example_2.csv

7.5 实例13：CSV格式的HTML展示

将CSV格式表示的二维表格数据展示为HTML：

1、读取CSV文件，获得文件的数据；

2、对数据进行处理和转换；

3、输出与上述HTML标记语言相同的HTML文件。

2016年7月部分大中城市新建住宅价格指数

城市	环比	同比	定基
北京	101.5	120.7	121.4
上海	101.2	127.3	127.8
广州	101.3	119.4	120
深圳	102	140.9	145.5
沈阳	100.1	101.4	101.6

price2016.html - 记事本

文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)

```
<!DOCTYPE HTML>
<html>
<body>
<meta charset=gb2312>
<h2 align=center>2016年7月部分大中城市新建住宅价格指数</h2>
<table border='1' align="center" width=70%>
<tr bgcolor='orange'>
<th width="25%">城市</th>
<th width="25%">环比</th>
<th width="25%">同比</th>
<th width="25%">定基</th>
</tr>
<tr><td align="center">北京</td><td align="center">101.5</td><td align="center">120.7</td><td
align="center">121.4</td></tr>
<tr><td align="center">上海</td><td align="center">101.2</td><td align="center">127.3</td><td
align="center">127.8</td></tr>
<tr><td align="center">广州</td><td align="center">101.3</td><td align="center">119.4</td><td
align="center">120</td></tr>
<tr><td align="center">深圳</td><td align="center">102</td><td align="center">140.9</td><td
align="center">145.5</td></tr>
<tr><td align="center">沈阳</td><td align="center">100.1</td><td align="center">101.4</td><td
align="center">101.6</td></tr>
</table>
</body>
</html>
```

price2016.html - 记事本

文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)

```
<!DOCTYPE HTML>
<html>
<body>
<meta charset=gb2312>
<h2 align=center>2016年7月部分大中城市新建住宅价格指数</h2>
<table border='1' align="center" width=70%>
<tr bgcolor='orange'>
<th width="25%">城市</th>
<th width="25%">环比</th>
<th width="25%">同比</th>
<th width="25%">定基</th>
</tr>
<tr><td align="center">北京</td><td align="center">101.5</td><td align="center">120.7</td><td
align="center">121.4</td></tr>
<tr><td align="center">上海</td><td align="center">101.2</td><td align="center">127.3</td><td
align="center">127.8</td></tr>
<tr><td align="center">广州</td><td align="center">101.3</td><td align="center">119.4</td><td
align="center">120</td></tr>
<tr><td align="center">深圳</td><td align="center">102</td><td align="center">140.9</td><td
align="center">145.5</td></tr>
<tr><td align="center">沈阳</td><td align="center">100.1</td><td align="center">101.4</td><td
align="center">101.6</td></tr>
</table>
</body>
</html>
```

①固定内容：用seg1表示

②从文件读取的表头：

③固定内容：用seg2表示

④从文件读取的详细数据：用fill_data函数实现

⑤固定内容：用seg3表示

```
seg1=‘’ ‘
<!DOCTYPE HTML>\n<html>\n<body>\n<meta charset=gb2312>
<h2 align=center>2016年7月部分大中城市新建住宅价格指数</h2>
<table border=’1‘ align=“center” width=70%>
<tr bgcolor=’orange‘>\n’ ‘’
seg2=“</tr>\n”
seg3=“</table>\n</body>\n</html>”
def fill_data(locls):
    seg=‘<tr><td align=“center”>{}</td><td align=“center”>{}</td><td align=“center”>{}</td><td
        align="center">{}</td></tr>\n'.format(*locls)
    return seg
fr=open("price2016.csv","r")
ls=[]
for line in fr:
    line=line.replace("\n","")
    ls.append(line.split(","))
fr.close()
fw=open("price2016.html","w")
fw.write(seg1)
fw.write('<th width="25%">{}</th>\n<th width="25%">{}</th>\n<th width="25%">{}</th>\n<th
        width="25%">{}</th>\n'.format(*ls[0]))
fw.write(seg2)
for i in range(len(ls)-1):
    fw.write(fill_data(ls[i+1]))
fw.write(seg3)
fw.close()
```

①固定内容：用seg1表示

②从文件读取的表头：

③固定内容：用seg2表示

④从文件读取的详细数据：用fill_data函数实现

⑤固定内容：用seg3表示

7.6 高维数据的格式化

高维数据为了保证其灵活性，不采用任何结构形式表示，采用基本的二元关系键值对表示。

XML和JSON是表示键值最常用的形式：
XML：eXtensible Markup Language
JSON:Javascript Object Notation
Python以JSON为主进行高维数据的表示。
有如下约定：

◆数据保存在键值对中；
◆键值对之间用逗号隔开；
◆大括号用于保存键值对数据组成的对象；
◆方括号用于保存键值对数据组成的数组。

```
"本书作者":[ {"姓氏":"嵩"
              "名字":"天"
              "单位":"北京理工大学"}
            {"姓氏":"礼"
              "名字":"欣"
              "单位":"北京理工大学"}
            {"姓氏":"黄"
              "名字":"天羽"
              "单位":"北京理工大学"}
           ]
```

7.7 json库的使用

json是python的标准库，直接用import json即可引入

操作类函数：外部JSON格式和程序内部数据类型之间的转换功能；

函数	描述
json.dumps(obj,sort_keys=False,indent=None)	将Python的数据类型转换为JSON格式，编码过程
json.loads(string)	将json格式字符串转换为Python的数据类型，解码过程
json.dump(obj,fp,sort_keys=False,indent=None)	与dumps功能一致，输出到文件fp
json.load(fp)	与loads功能一致，从文件fp读入

obj：python的列表或者字典类型

sort_keys:可以对字典元素按key进行排序，控制输出结果

indent：用于增加数据缩进，使得生成的json格式字符串更具有可读性

7.8 实例14：CSV和JSON格式相互转换

CSV：用于一二维数据的表示和存储，是一种纯文本形式存储表格数据的表示方式

“同比”：“120.7”
“城市”：“北京”
“定基”：“121.4”
“环比”：“101.5”

JSON：也可以表示一二维数据

```
{
        “同比”：“120.7”
        “城市”：“北京”
        “定基”：“121.4”
        “环比”：“101.5”
}
```

二者有时需要相互转换

7.8 实例14：CSV和JSON格式相互转换

```
import json
fr=open("price2016.csv","r")
ls=[]
for line in fr:
    line=line.replace("\n","")
    ls.append(line.split(','))
fr.close()
fw=open("price2016.json","w")
for i in range(1,len(ls)):
    ls[i]=dict(zip(ls[0],ls[i]))
json.dump(ls[1:],fw,sort_keys=True
fw.close()
```

将每行的换行符用“”替换

将每行字符按“，”分隔后追加到ls中；

```
Ls：[['城市', '环比', '同比', '定基'],
     ['北京', '101.5', '120.7', '121.4'],
     ['上海', '101.2', '127.3', '127.8'],
     ['广州', '101.3', '119.4', '120'],
     ['深圳', '102', '140.9', '145.5'],
     ['沈阳', '100.1', '101.4', '101.6']
    ]
```

7.8 实例14：CSV和JSON格式相互转换

```
import json
fr=open("price2016.csv","r")
ls=[]
for line in fr:
    line=line.replace("\n","")
    ls.append(line.split(','))
fr.close()
fw=open("price2016.json","w")
for i in range(1,len(ls)):
    ls[i]=dict(zip(ls[0],ls[i]))
json.dump(ls[1:],fw,sort_keys=True,indent=4,ensure_ascii=False)
fw.close()
```

zip函数是将两个长度相同的列表组合成一个关系对，一般用于键值对的生成，如：
ls[0]= ['城市', '环比', '同比', '定基']
ls[1]= ['北京', '101.5', '120.7', '121.4']
zip(ls[0],ls[1])
[('城市','北京'),('环比','101.5'),('同比','120.7'),('定基','121.4')]

第一行数据['城市', '环比', '同比', '定基']保留，从第二行开始，用zip函数生成键值对，对每个键值对利用dict转换为字典

json默认采用unicode编码处理非中文字符，为避免编码格式不同带来的问题，可设置ensure_ascii=False，使json库输出中文字符。

7.8 实例14：CSV和JSON格式相互转换

```
import json
fr=open("price2016.csv","r")
ls=[]
for line in fr:
    line=line.replace("\n","")
    ls.append(line.split(','))
fr.close()
fw=open("price2016.json","w")
for i in range(1,len(ls)):
    ls[i]=dict(zip(ls[0],ls[i]))
json.dump(ls[1:],fw,sort_keys=True,indent=4,ensure_ascii=False)
fw.close()
```

将ls[1]开始的数据写到fw指定的文件中，sort_keys=True表示按键排序，indent=4表示缩4个空格，

7.8 实例14：CS

```
import json
fr=open("price2016.csv","r")
ls=[]
for line in fr:
    line=line.replace("\n","")
    ls.append(line.split(','))
fr.close()
fw=open("price2016.json","w")
for i in range(1,len(ls)):
    ls[i]=dict(zip(ls[0],ls[i]))
json.dump(ls[1:],fw,sort_keys=True,indent=4,ensure_ascii=False)
fw.close()
```

```
Ls：[['城市', '环比', '同比', '定基'],
     ['北京', '101.5', '120.7', '121.4'],
     ['上海', '101.2', '127.3', '127.8'],
     ['广州', '101.3', '119.4', '120'],
     ['深圳', '102', '140.9', '145.5'],
     ['沈阳', '100.1', '101.4', '101.6']
    ]
```

zip函数是将两个长度相同的列表组合成一个关系对，一般用于键值对的生成，如：

x=[1,2,3]

y=[“a”,”b”,”c”]

list(zip(x,y)}

[(1,”a”),(2,”b”),(3,”c”)]

Json默认采用unicode编码处理非中文字符，为避免编码格式不同带来的问题，可设置ensure_ascii=False，使json库输出中文字符。

7.7 将表2的example_2.csv文件转换为example_3.json文件

姓名	学号	数学	语文	英语	总分
张三锋	170101	76	90	88	254
李四娘	170102	83	86	87	256
王五指	170103	92	79	86	257
赵六子	170104	65	61	85	211
钱多多	170105	59	93	84	236

表2 example_2.csv