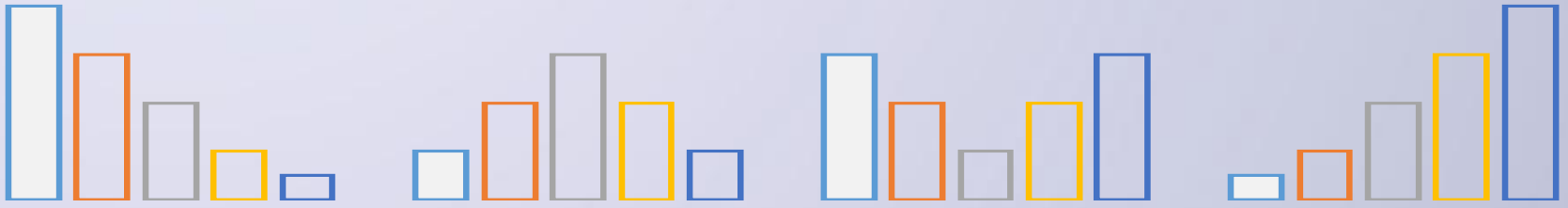
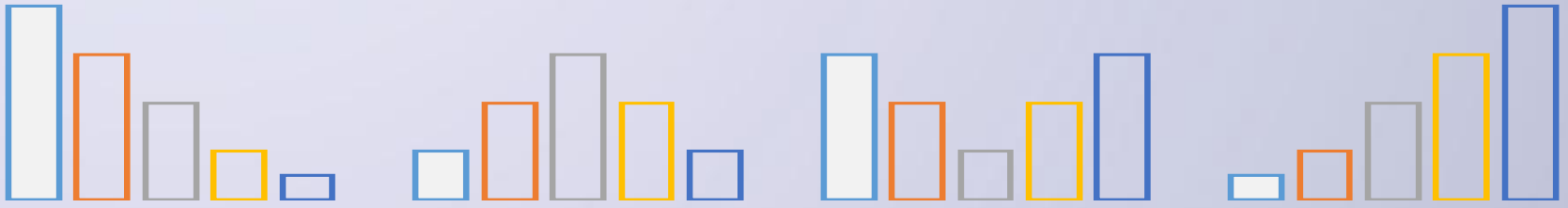
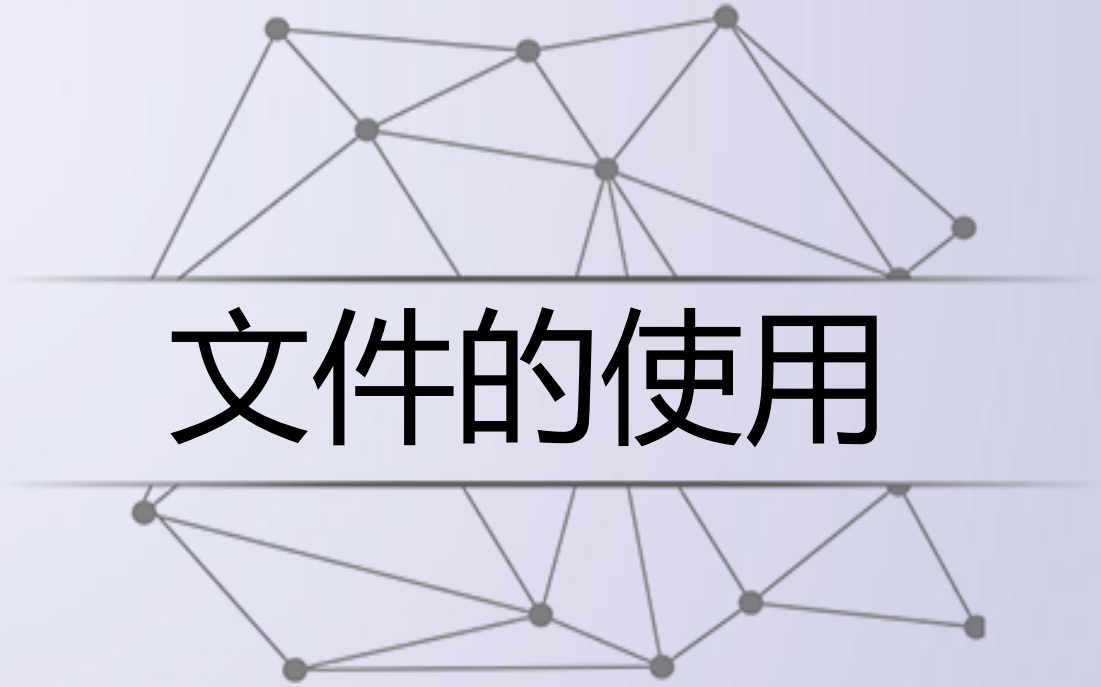
Python语言程序设计

北京理工大学 嵩天



第7章 文件和数据格式化





文件的使用

文件概述

文件是一个存储在辅助存储器上的数据序列，可以包含任何数据内容。概念上，文件是数据的集合和抽象，类似地，函数是程序的集合和抽象。用文件形式组织和表达数据更有效也更为灵活。文件包括两种类型：文本文件和二进制文件。

文件概述

二进制文件直接由比特0和比特1组成，没有统一字符编码，文件内部数据的组织格式与文件用途有关。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，打开后的操作不同。

文件概述

微实例7.1：理解文本文件和二进制文件的区别。

微实例7.1　　m7.1DiffTextBin.py

```
1  textFile = open("7.1.txt","rt") #t表示文本文件方式
2  print(textFile.readline())
3  textFile.close()
4  binFile = open("7-1.txt","rb")    #r表示二进制文件方式
5  print(binFile.readline())
6  binFile.close()
```

文件概述

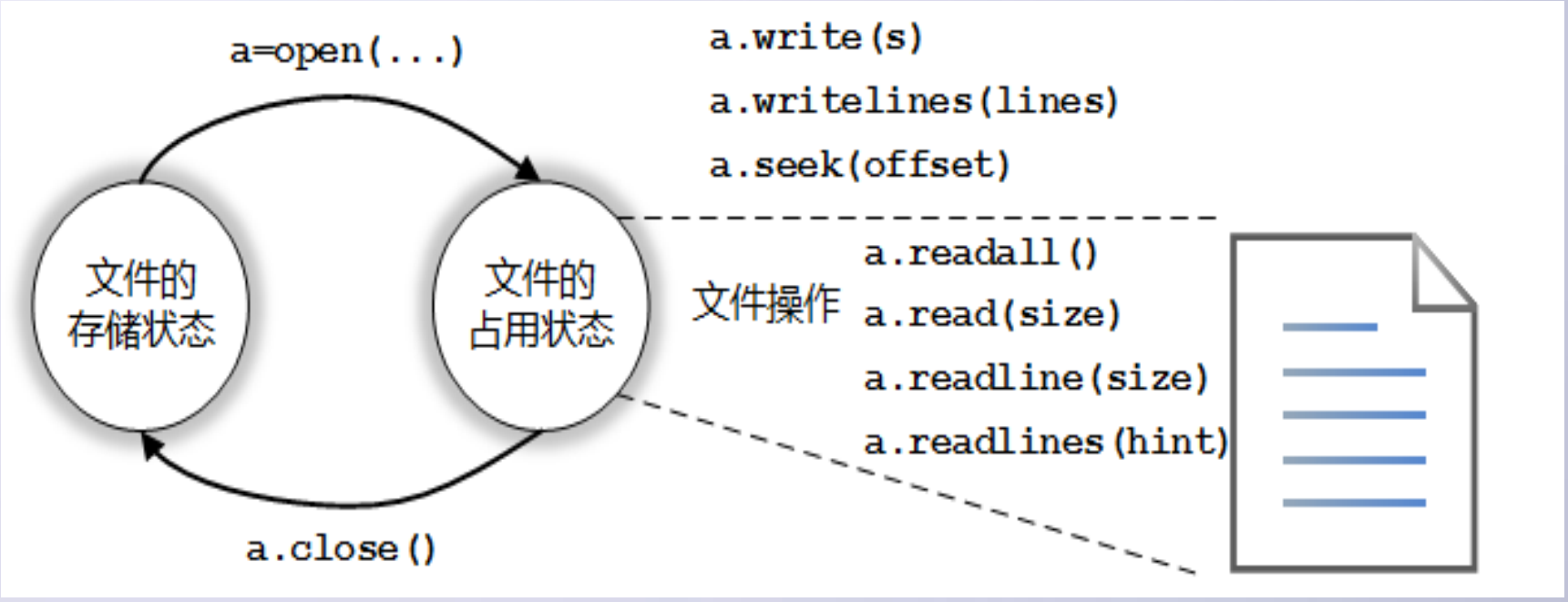
输出结果为：

```
>>>
中国是个伟大的国家！
b'\xd6\xd0\xb9\xfa\xca\xc7\xb8\xf6\xce\xb0\x
b4\xf3\xb5\xc4\xb9\xfa\xbc\xd2\xa3\xa1'
```

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（byte）流。由于存在编码，字符串中的一个字符由2个字节表示。

文件的打开关闭

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”



文件的打开关闭

Python通过解释器内置的open()函数打开一个文件，并实现该文件与一个程序变量的关联，open()函数格式如下：

<变量名> = open(<文件名>, <打开模式>)

open()函数有两个参数：文件名和打开模式。文件名可以是文件的实际名字，也可以是包含完整路径的名字

文件的打开关闭

open()函数提供7种基本的打开模式

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

文件的读写

根据打开方式不同可以对文件进行相应的读写操作，Python提供4个常用的文件内容读取方法

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)	从文件中读入整个文件内容，如果给出参数，读入前size长度的字符串或字节流
<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size长度的字符串或字节流
<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

文件的读写

微实例7.2：文本文件逐行打印

微实例7.2　　m7.2PrintFilebyLines.py

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

文件的读写

遍历文件的所有行可以直接这样完成

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

文件的读写

如果程序需要逐行处理文件内容，建议采用上述代码格式：

```
fo = open(fname, "r")
for line in fo:
    # 处理一行数据
fo.close()
```

文件的读写

Python提供3个与文件内容写入有关的方法，如表所示。

方法	含义
<file>.write(s)	向文件写入一个字符串或字节流
<file>.writelines(lines)	将一个元素为字符串的列表写入文件
<file>.seek(offset)	改变当前文件操作指针的位置，offset的值： 0：文件开头； 1: 当前位置； 2: 文件结尾

文件的读写

微实例7.3　　m7.3WriteListtoFile.py

```
1  fname = input("请输入要写入的文件： ")
2  fo = open(fname, "w+")
3  ls = ["唐诗", "宋词", "元曲"]
4  fo.writelines(ls)
5  for line in fo:
6  print(line)
7  fo.close()
```

程序执行结果如下：

```
>>>请输入要写入的文件： test.txt
>>>
```

PIL库的使用

PIL库概述

PIL（Python Image Library）库是Python语言的第三方库，需要通过pip工具安装。

```
:\>pip install pillow   # 或者 pip3 install pillow
```

PIL库支持图像存储、显示和处理，它能够处理几乎所有图片格式，可以完成对图像的缩放、剪裁、叠加以及向图像添加线条、图像和文字等操作。

PIL库概述

PIL库可以完成图像归档和图像处理两方面功能需求：

- 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：图像基本处理、像素处理、颜色处理等。

PIL库Image类解析

在PIL中，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。

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的图像
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

PIL库Image类解析

要加载一个图像文件，最简单的形式如下，之后所有操作对im起作用

```
>>>from PIL import Image
>>>im = Image.open("D:\\pycodes\\birdnest.jpg")
```

PIL库Image类解析

Image类有4个处理图片的常用属性

属性	描述
Image.format	标识图像格式或来源，如果图像不是从文件读取，值是None
Image.mode	图像的色彩模式，"L"灰度图像、"RGB"真彩色图像、"CMYK"出版图像
Image.size	图像宽度和高度，单位是像素（px），返回值是二元元组（tuple）
Image.palette	调色板属性，返回一个ImagePalette类型

PIL库Image类解析

微实例7.1：GIF文件图像提取。

对一个GIF格式动态文件，提取其中各帧图像，并保存为文件。

微实例7.1　　m7.1GifExtractor.py

```
from PIL import Image
im = Image.open('pybit.gif')          # 读入一个GIF文件
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")
```

PIL库Image类解析

Image类的图像转换和保存方法如表所示。

方法	描述
Image.save(filename, format)	将图像保存为filename文件名，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

PIL库Image类解析

生成"birdnest.jpg"图像的缩略图，其中（128，128）是缩略图的尺寸。

```
>>>im.thumbnail((128, 128))
>>>im.save("birdnestTN","JPEG")
```

北京鸟巢图片及其缩略图

PIL库Image类解析

Image类可以缩放和旋转图像，其中，rotate()方法以逆时针旋转的角度值作为参数来旋转图像。

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

PIL库Image类解析

Image类能够对每个像素点或者一幅RGB图像的每个通道单独进行操作,split()方法能够将RGB图像各颜色通道提取出来，merge()方法能够将各独立通道再合成一幅新的图像。

方法	描述
Image.point(func)	根据函数func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，采用mode色彩，bands是新色的色彩通道
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

PIL库Image类解析

微实例7.2：图像的颜色交换。

交换图像中的颜色。可以通过分离RGB图片的三个颜色通道实现颜色交换

微实例7.2　　m7.1ChangeRGB.py

```
1  from PIL import Image
2  im = Image.open('birdnest.jpg')
3  r, g, b = im.split()
4  om = Image.merge("RGB", (b, g, r))
5  om.save('birdnestBGR.jpg')
```

PIL库Image类解析

被改变颜色的北京鸟巢图片

PIL库Image类解析

操作图像的每个像素点需要通过函数实现，采用lambda函数和point()方法搭配使用，例子如下

```
>>>im = Image.open('D:\\pycodes\\birdnest.jpg') #打开鸟巢文件
>>>r, g, b = im.split() #获得RGB通道数据
>>>newg = g.point(lambda i: i * 0.9) # 将G通道颜色值变为原来的0.9倍
>>>newb = b.point(lambda i: i < 100) # 选择B通道值低于100的像素点
>>>om = Image.merge(im.mode, (r, newg, newb)) # 将3个通道合形成新图像
>>>om.save('D:\\pycodes\\birdnestMerge.jpg') #输出图片
```

PIL库Image类解析

去掉光线的北京鸟巢图片

图像的过滤和增强

PIL库的ImageFilter类和ImageEnhance类提供了过滤图像和增强图像的方法，共10种

方法表示	描述
ImageFilter.BLUR	图像的模糊效果
ImageFilter.CONTOUR	图像的轮廓效果
ImageFilter.DETAIL	图像的细节效果
ImageFilter.EDGE_ENHANCE	图像的边界加强效果
ImageFilter.EDGE_ENHANCE_MORE	图像的阈值边界加强效果
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

图像的过滤和增强

利用Image类的filter()方法可以使用ImageFilter类，如下：

Image.filter(ImageFilter.fuction)

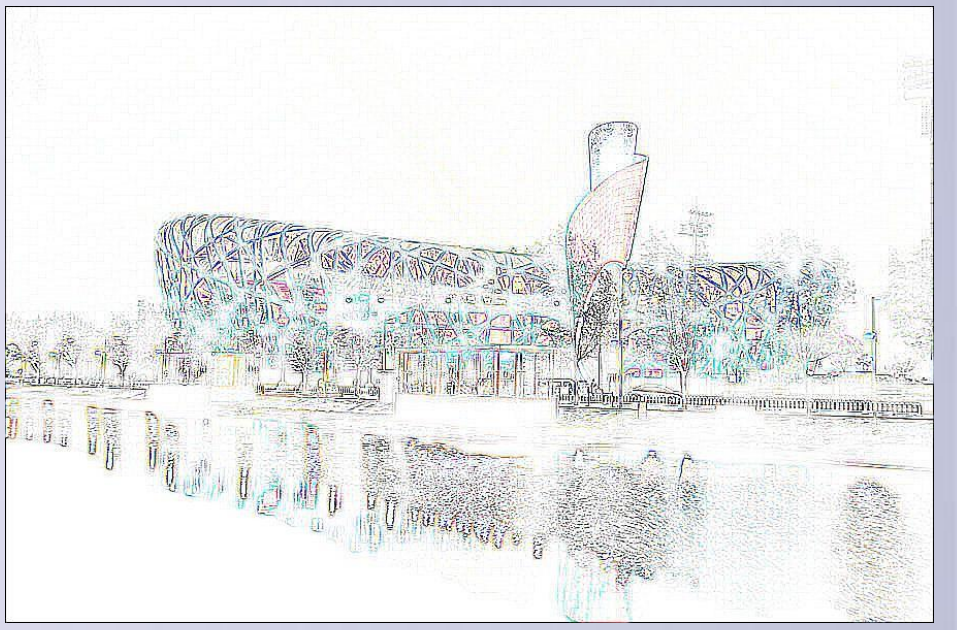
微实例7.3：图像的轮廓获取。

获取图像的轮廓，北京鸟巢变得更加抽象、更具想象空间！

图像的过滤和增强

微实例7.3　　m7.3GetImageContour.py

```
1  from PIL import Image
2  from PIL import ImageFilter
3  im = Image.open('birdnest.jpg')
4  om = im.filter(ImageFilter.CONTOUR)
5  om.save('birdnestContour.jpg')
```



北京鸟巢图片的轮廓效果

图像的过滤和增强

ImageEnhance类提供了更高级的图像增强需求，它提供调整色彩度、亮度、对比度、锐化等功能。

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

图像的过滤和增强

微实例7.4：图像的对比度增强。

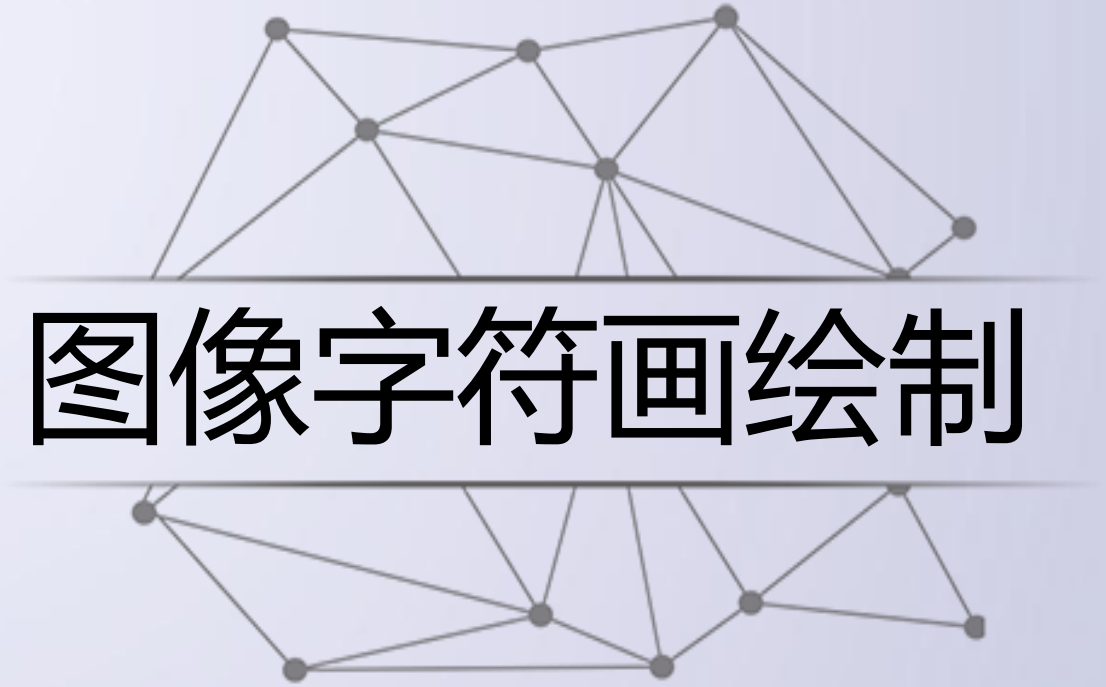
增强图像的对比度为初始的20倍。

微实例7.4　m7.4EnImageContrast.py

```
1  from PIL import Image
2  from PIL import ImageEnhance
3  im = Image.open('birdnest.jpg')
4  om = ImageEnhance.Contrast(im)
5  om.enhance(20).save('birdnestEnContrast.jpg')
```

图像的过滤和增强

北京鸟巢图片的20倍对比度增强效果



图像字符画绘制

图像字符画绘制

位图图片是由不同颜色像素点组成的规则分布，如果采用字符串代替像素，图像就成为了字符画。

定义一个字符集，将这个字符集替代图像中的像素点。

```
ascii_char =list("$@B%8&WM#*oahkbdpqwmZO0QLCJUYXzcvunxrjf\
                 1234568795t/\|()1{}[]?-_+~<>i!;:,\"^`'.")
```

图像字符画绘制

定义彩色向灰度的转换公式如下，其中R、G、B分别是像素点的RGB颜色值：

Gray = R * 0.2126 + G * 0.7152 + B * 0.0722

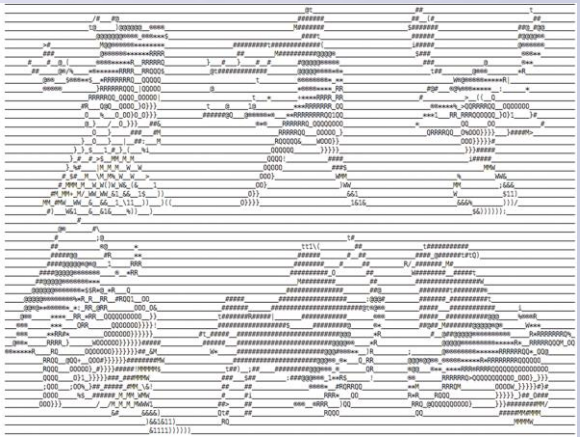
因此，像素的RGB颜色值与字符集的对应函数如下：

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

实例代码12.1 e12.DrawCharImage.py

```
#e12.1DrawCharImage.py.py
from PIL import Image
ascii_char  = list('"$%_&WM#*oahkbdpqwmZO0QLCJUYXzcvunxr\
jft/\|()1{}[]?-/+@<>i!;:,\^`.')
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('pic.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()
```

图像字符画绘制



一二维数据格式化和处理

数据组织的维度

一维数据由对等关系的有序或无序数据构成，采用线性方式组织，对应于数学中的数组和集合等概念。

中国、美国、日本、德国、法国、英国、意大利、加拿大、俄罗斯、欧盟、澳大利亚、南非、阿根廷、巴西、印度、印度尼西亚、墨西哥、沙特阿拉伯、土耳其、韩国

数据组织的维度

二维数据，也称表格数据，由关联关系数据构成，采用表格方式组织，对应于数学中的矩阵，常见的表格都属于二维数据。

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	120.0
深圳	102.0	140.9	145.5
沈阳	100.1	101.4	101.6

环比：上月=100；同比：上年同月=100；定基：2015年=100。

数据组织的维度

高维数据由键值对类型的数据构成，采用对象方式组织，属于整合度更好的数据组织方式。高维数据在网络系统中十分常用，HTML、XML、JSON等都是高维数据组织的语法结构。

数据组织的维度

```
"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"   }
]
```

一二维数据的存储格式

一维数据是最简单的数据组织类型，有多种存储格式，常用特殊字符分隔：

（1）用一个或多个空格分隔，例如：

中国 美国 日本 德国 法国 英国 意大利

（2）用逗号分隔，例如：

中国,美国,日本,德国,法国,英国,意大利

（3）用其他符号或符号组合分隔，建议采用不出现在数据中的特殊符号

中国; 美国; 日本; 德国; 法国; 英国; 意大利

一二维数据的存储格式

逗号分割数值的存储格式叫做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，尤其应用在程序之间转移表格数据。

一二维数据的存储格式

该格式的应用有一些基本规则，如下：

（1）纯文本格式，通过单一编码表示字符；

（2）以行为单位，开头不留空行，行之间没有空行；

（3）每行表示一个一维数据，多行表示二维数据；

（4）以逗号分隔每列数据，列数据为空也要保留逗号；

（5）可以包含或不包含列名，包含时列名放置在文件第一行。

一二维数据的存储格式

二维数据采用CSV存储后的内容如下：

```
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
```

CSV格式存储的文件一般采用.csv为扩展名，可以通过Windows平台上的记事本或微软Office Excel工具打开，也可以在其他操作系统平台上用文本编辑工具打开。

一二维数据的表示和读写

CSV文件的每一行是一维数据，可以使用Python中的列表类型表示，整个CSV文件是一个二维数据，由表示每一行的列表类型作为元素，组成一个二维列表。

```
[
['城市', '环比', '同比', '定基\n'],
['北京', '101.5', '120.7', '121.4\n'],
['上海', '101.2', '127.3', '127.8\n'],
['广州', '101.3', '119.4', '120.0\n'],
['深圳', '102.0', '140.9', '145.5\n'],
['沈阳', '100.1', '101.4', '101.6\n'],
]
```

一二维数据的表示和读写

微实例7.5：导入CSV格式数据到列表

微实例7.5　　m7.5GetCSV2List.py

```
1 fo = open("price2016.csv", "r")
2 ls = []
3 for line in fo:
4 line = line.replace("\n","")
5 ls.append(line.split(","))
6 print(ls)
7 fo.close()
```

一二维数据的表示和读写

需要注意，以split(",")方法从CSV文件中获得内容时，每行最后一个元素后面包含了一个换行符（"\n"）。对于数据的表达和使用来说，这个换行符是多余的，可以通过使用字符串的replace()方法将其去掉，如第4行。

一二维数据的表示和读写

微实例7.6：逐行处理CSV格式数据。

微实例7.6　　m7.6GetCSVbyLine.py

```
1  fo = open("price2016.csv", "r")
2  ls = []
3  for line in fo:
4  line = line.replace("\n","")
5  ls = line.line.split(",")
6  lns = ""
7  for s in ls:
8            lns += "{}\t".format(s)
9  print(lns)
10 fo.close()
```

一二维数据的表示和读写

运行后的输出结果如下：

```
>>>
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     120.0
深圳      102.0     140.9     145.5
沈阳      100.1     101.4     101.6
```

一二维数据的表示和读写

微实例7.7：一维数据写入CSV文件。

微实例7.7　　m7.7WriteD1toCSV.py

```
1  fo = open("price2016bj.csv", "w")
2  ls  =  [' 北 京 ',  '101.5',  '120.7',
   '121.4']
3  fo.write(",".join(ls)+ "\n")
4  fo.close()
```

一二维数据的表示和读写

对于列表中存储的二维数据，可以通过循环写入一维数据的方式写入CSV文件，参考代码样式如下：

```
for row in ls:
    <输出文件>.write(",".join(row)+"\n")
```

一二维数据的表示和读写

微实例7.8：二维数据写入CSV文件。

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%

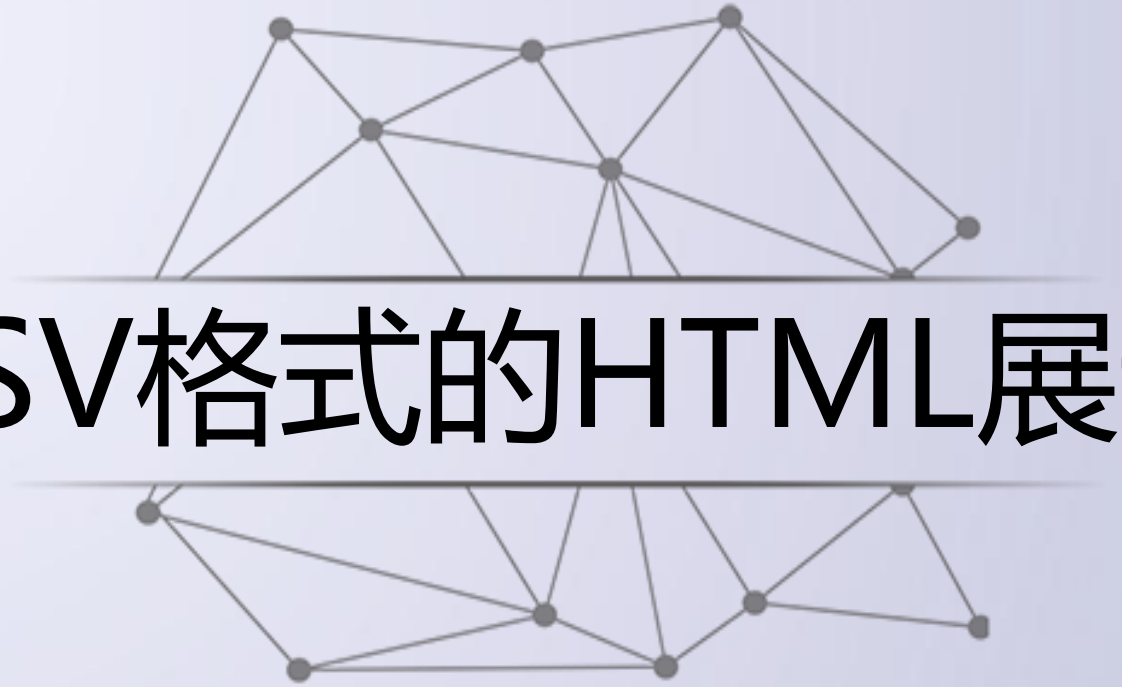
深圳,1.0%,1.4%,1.5%

沈阳,1.0%,1.0%,1.0%

一二维数据的表示和读写

微实例7.8　　m7.8WriteD2toCSV.py

```
fr = open("price2016.csv", "r")
fw = open("price2016out.csv", "w")
ls = []
for line in fr:      #将CSV文件中的二维数据读入到列表变量
    line = line.replace("\n","")
    ls.append(line.split(","))
for i in range(len(ls)):    #遍历列表变量计算百分数
    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:     #将列表变量中的二位数据输出到CSV文件
   print(row)
    fw.write(",".join(row)+"\n")
fr.close()
fw.close()
```



CSV格式的HTML展示

CSV格式的HTML展示

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	120.0
深圳	102.0	140.9	145.5
沈阳	100.1	101.4	101.6

文件名：CSV2HTML.html

```
<!DOCTYPE HTML>
<html>
<body>
<meta charset=utf-8>
<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>北京</td><td>101.5</td><td>120.7</td><td>121.4</td></tr>
<tr><td>上海</td><td>101.2</td><td>127.3</td><td>127.8</td></tr>
<tr><td>广州</td><td>101.3</td><td>119.4</td><td>120.0</td></tr>
<tr><td>深圳</td><td>102.0</td><td>140.9</td><td>145.5</td></tr>
<tr><td>沈阳</td><td>100.1</td><td>101.4</td><td>101.6</td></tr>
</table>
</body>
</html>
```

CSV格式的HTML展示

实例代码13.1　　e13.1csv2html.py

```
#e13.1csv2html.py
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><tdalign="center">\
{}</td></tr>\n'.format(*locls)
    return seg
fr = open("price2016.csv", "r")
ls = []
```

CSV格式的HTML展示

实例代码13.1　　e13.1csv2html.py

```
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()
```

高维数据的格式化

高维数据的格式化

与一维二维数据不同，高维数据能展示数据间更为复杂的组织关系。为了保持灵活性，表示高维数据不采用任何结构形式，仅采用最基本的二元关系，即键值对。万维网是高维数据最成功的典型应用。

JSON格式可以对高维数据进行表达和存储。JSON（JavaScript Object Notation）是一种轻量级的数据交换格式，易于阅读和理解。JSON格式表达键值对<key, value>的基本格式如下，键值对都保存在双引号中：

"key" : "value"

高维数据的格式化

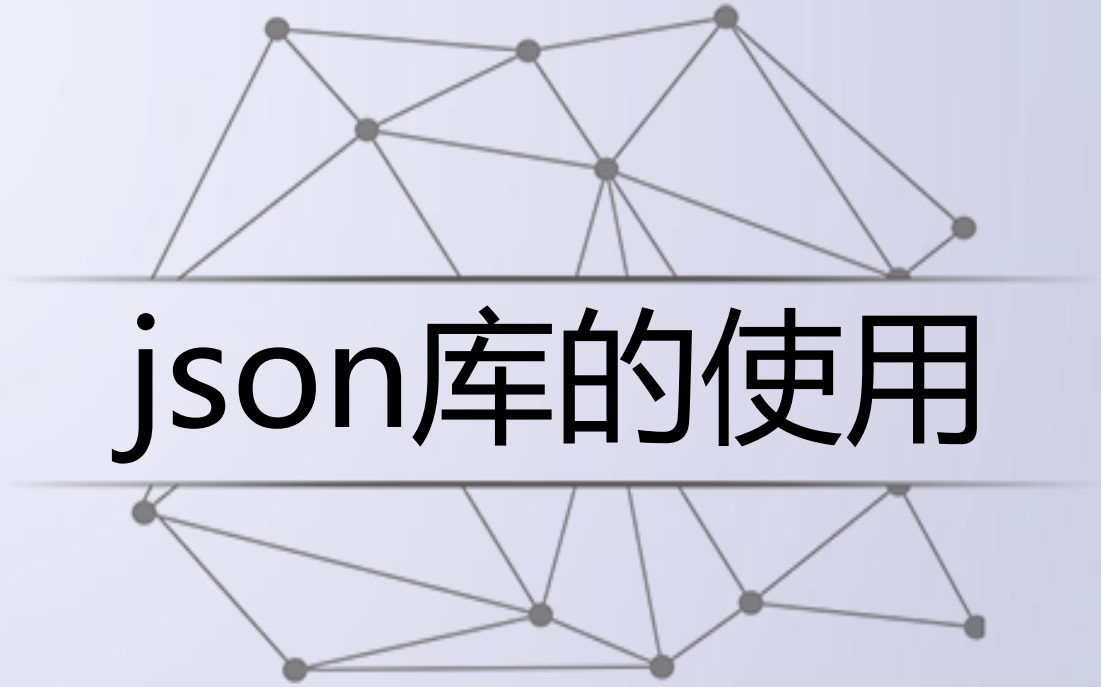
当多个键值对放在一起时，JSON有如下一些约定：

- 数据保存在键值对中；
- 键值对之间由逗号分隔；
- 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。

以“本书作者”JSON数据为例。

高维数据的格式化

```
"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"  }
]
```



json库的使用

Json库的概述

- json库主要包括两类函数：操作类函数和解析类函数
  - 操作类函数主要完成外部JSON格式和程序内部数据类型之间的转换功能
  - 解析类函数主要用于解析键值对内容。

Json库的解析

dumps()和loads()分别对应编码和解码功能。

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

Json库的解析

```
>>>dt = {'b':2,'c':4,'a':6}
>>>s1 = json.dumps(dt)  #dumps返回JSON格式的字符串类型
>>>s2 = json.dumps(dt,sort_keys=True,indent=4)
>>>print(s1)
{"c": 4, "a": 6, "b": 2}
>>>print(s2)
{
    "a": 6,
    "b": 2,
    "c": 4
}
>>>print(s1==s2)
False
>>>dt2 = json.loads(s2)
>>>print(dt2, type(dt2))
{'c': 4, 'a': 6, 'b': 2} <class 'dict'>
```

CSV和JSON格式相互转换

CSV和JSON格式相互转换

CSV格式常用于一二维数据表示和存储，JSON也可以表示一二维数据。在网络信息传输中，可能需要统一表示方式，因此，需要在CSV和JSON格式间进行相互转换。

"同比": "120.7",
"城市": "北京",
"定基": "121.4",
"环比": "101.5"

CSV和JSON格式相互转换

将CSV转换成JSON格式的代码如下

实例代码14.1　　e14.1csv2json.py

```
1  #e14.1csv2json.py
2  import json
3  fr = open("price2016.csv", "r")
4  ls = []
5  for line in fr:
6      line = line.replace("\n","")
7      ls.append(line.split(','))
8  fr.close()
9  fw = open("price2016.json", "w")
10 for i in range(1,len(ls)):
11     ls[i] = dict(zip(ls[0], ls[i]))
12 json.dump(ls[1:],fw, sort_keys=True, indent=4)
13 fw.close()
```

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

将二维JSON格式数据转换成CSV格式的代码如下，供参考。

实例代码14.2　　e14.2json2csv.py

```
1   #14.2json2csv.py
2   import json
3   fr = open("price2016.json", "r")
4   ls = json.load(fr)
5   data = [ list(ls[0].keys()) ]
6   for item in ls:
7       data.append(list(item.values()))
8   fr.close()
9   fw = open("price2016_from_json.csv", "w")
10  for item in data:
11      fw.write(",".join(item) + "\n")
12  fw.close()
```