Python程序设计

陈远祥

chenyxmail@gmail.com

北京邮电大学 电子工程学院

上周主要内容

- 文件的读取

利用正则表达式从网页提取图片

图片.py - F:\北邮课程\Python程序设计\备课\第五周\抓取图片\图片.py (3.6.4)

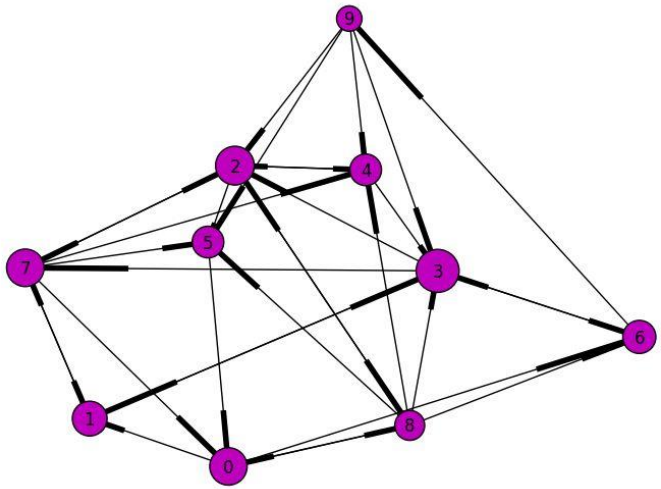
File Edit Format Run Options Window Help

```
#抓取图片
import re
import urllib #urllib库，操作URL的功能
#获取网页信息
url="https://bj.lianjia.com/zufang/"
page = urllib.request.urlopen(url)#抓取网页的源代码
html = page.read()#bytes类型
html=html.decode('utf-8')#str类型
#获取图片，正则表达式
reg=r'data-img="(.*?\.jpg)" alt'#源代码格式图片
imgre=re.compile(reg)
imglist = re.findall(imgre,html)
#抓取页面图片并保存到本地
x=0
for img in imglist:
    urllib.request.urlretrieve(img,'%s.jpg' % x)#直接将远程数据下载到本地
    x+=1
```

findall

- findall()
  - ✓ 如果调用在一个没有分组的正则表达式上，则返回一个匹配字符串的列表
  - ✓ 如果调用在一个有分组的正则表达式上，则返回一个字符串的元组的列表
  - ✓ 如果单个分组，只返回分组列表

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

"addressbook":[
{'姓名':'小明','性别':'男','地址':'西土城路1号’ },
{'姓名':'小黑','性别':'男','地址':'西土城路2号’ },
{'姓名':'小红','性别':'女','地址':'西土城路1号’ }
]

数据组织的维度

- 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：
  - ✓ 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

  - ✓ 用逗号分隔，例如：

中国,美国,日本,德国,法国,英国,意大利

  - ✓ 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

数据组织的维度

- 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

数据组织的维度

- 该格式的应用有一些基本规则，如下：
  - ✓ 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；
  - ✓ 以逗号分隔每列数据，列数据为空也要保留逗号；
  - ✓ 可以包含或不包含列名，包含时列名放置在文件第一行

数据组织的维度

- 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

数据组织的维度

- 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OfficeExcel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
  - ✓ CSV创建，打开

数据组织的维度

- 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'],
]
```

数据组织的维度

- 导入CSV格式数据到列表
  - ✓ #房价

房价.py

```
1 fo=open("房价.csv","r",encoding='UTF-8_sig')
2 #有些软件UTF-8编码，带bom，去掉bom。BOM——Byte Order Mark，就是字节序标记
3 ls=[]
4 for line in fo:
5     line=line.replace("\n","")
6 #换行符多余,去掉
7     ls.append(line.split(","))
8     print(ls)
9 fo.close()
```

数据组织的维度

- 逐行处理CSV格式数据
  - ✓ #房价1

```
1 fo=open("房价.csv","r",encoding='UTF-8_sig')
2 #有些软件UTF-8编码，带bom，去掉bom。BOM——Byte Order Mark，就是字节
3 ls=[]
4 for line in fo:
5     line=line.replace("\n","")
6 #换行符多余,去掉
7     ls.append(line.split(","))
8 lns=""
9 for s in ls:
10    lns+="{}\t".format(s)
11 print(lns)
12 fo.close()
13
```

数据组织的维度

- 一维数据写入CSV文件
  - ✓ #房价2

```
fo=open("2018房价.csv","w",encoding='UTF-8_sig')
ls=['北京','101.5','120.7','121.4']
fo.write(",".join(ls)+"\n")
fo.close()
```

数据组织的维度

- 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，代码样式如下：

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

数据组织的维度

- 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%

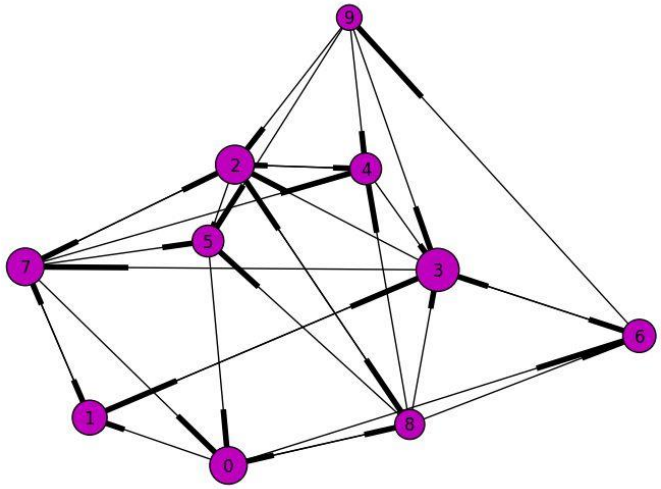
  深圳,1.0%,1.4%,1.5%

  沈阳,1.0%,1.0%,1.0%

  ✓ #房价3

```
fr= open("房价.csv", "r",encoding='UTF-8_sig')
fw= open("房价2017.csv", "w",encoding='UTF-8_sig')
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比
    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()
```

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

高维数据的格式化

- 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"

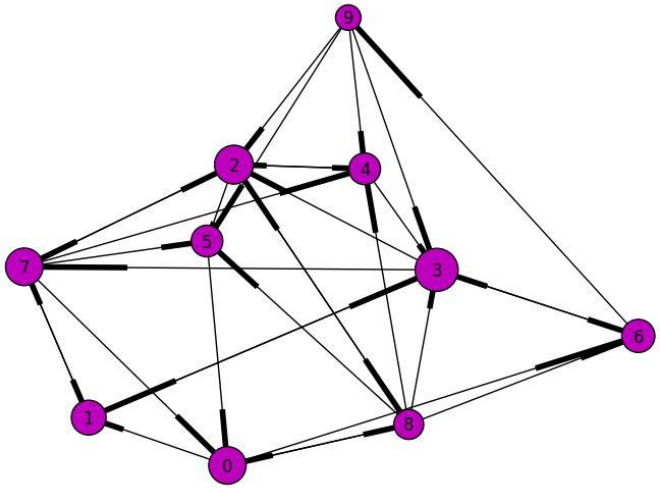
高维数据的格式化

- 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；
  - ✓ 花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

数据组织的维度

"addressbook":[
{‘姓名’:‘小明’,‘性别’:‘男’,‘地址’:‘西土城路1号’ },
{'姓名':'小黑','性别':'男','地址':'西土城路2号’ },
{'姓名':'小红','性别':'女','地址':'西土城路1号’ }
]

json库的使用



json库的使用

- 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。数据保存在键值对中

json库的使用

- 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

- ✓ #json1

```
import json
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=2)#help(json.dumps)
print(s1)
print(s2)
```

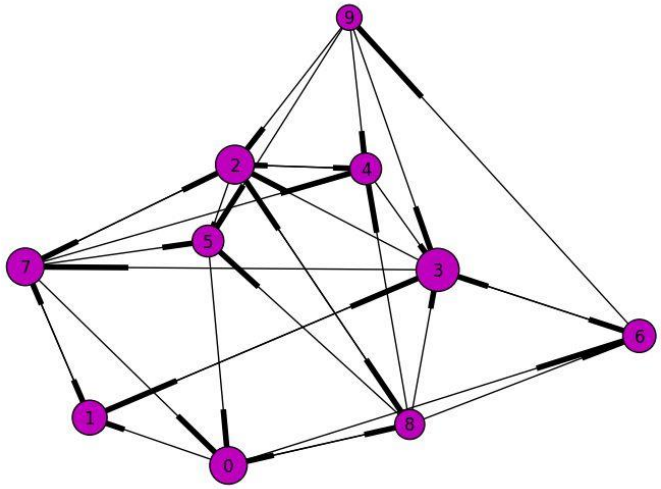
CSV和JSON格式相互转换

- 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
  - ✓ #转换1
  - ✓ #转换2

```
import json
fr= open("房价.csv", "r",encoding='UTF-8_sig')
ls = []
for line in fr:
    line = line.replace("\n","")
    ls.append(line.split(','))
fr.close()
fw= open("房价.json", "w",encoding='UTF-8_sig')
for i in range(1,len(ls)):
    ls[i] = dict(zip(ls[0], ls[i]))
    #把key和value的list组合在一起，再转成字典(dict)
json.dump(ls[1:],fw, sort_keys=True,ensure_ascii=False,indent=4)
#ensure_ascii,当它为True的时候，所有非ASCII码字符显示为\uXXXX序列，
fw.close()
```

```
import json
fr=open("房价.json","r",encoding='UTF-8_sig')
ls=json.load(fr)
data=[list(ls[0].keys())]
for item in ls:
    data.append(list(item.values()))
fr.close()
fw=open("房价from_json.csv","w")
for item in data:
    fw.write(",".join(item)+"\n")
fw.close()
```

处理Excel电子表格



Excel电子表格

- 一个Excel电子表格文档称为一个工作簿（workbook），一个工作簿保存在扩展名为.xlsx的文件中
- 每个工作簿可以包含多个表（也称为工作表workbook）
- 用户当前查看的表（或关闭Excel前最后查看的表），称为活动表
- 每个表都有一些列（地址是从A开始的字母）和一些行（地址是从1开始的数字）
- 在特定行和列的方格称为单元格（cell）。每个单元格都包含一个数字或文本值

Excel电子表格

- 安装openpyxl模块： import openpyxl
- 用openpyxl模块打开Excel文档

  wb=openpyxl.load_workbook()
- 从工作簿中取得工作表

  get_sheet_names()
  - ✓ #工作表1

```
import openpyxl
wb = openpyxl.load_workbook('example.xlsx')
sheet = wb.get_sheet_by_name('Sheet1')
sheet['A1']  #按名字访问cell对象
sheet['A1'].value#访问对象值
sheet.cell(row=1, column=2)
#可以传入整数作为row 和column 关键字
sheet.cell(row=1, column=2).value
```

Excel电子表格

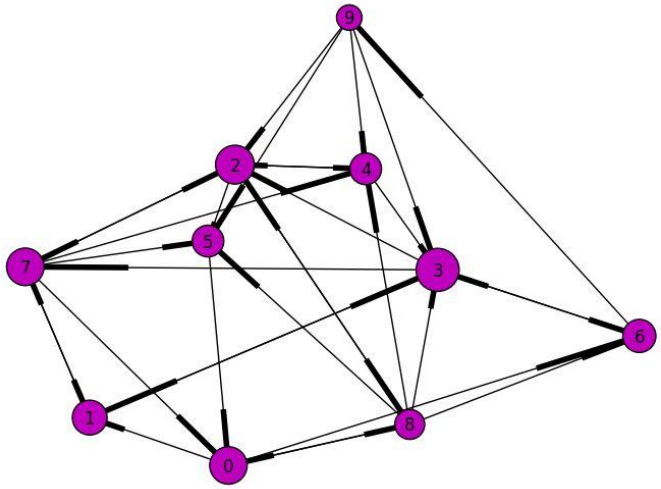
- 从表中取得行和列
  - ✓ 可以将Worksheet对象切片，取得电子表格中一行、一列或一个矩形区域中的所有Cell对象，然后可以循环遍历这个切片中的所有单元格
  - ✓ #工作表2

```
import openpyxl
wb = openpyxl.load_workbook('example.xlsx')
sheet = wb.get_sheet_by_name('Sheet1')
tuple(sheet['A1':'C3'])
#可以使用它的tuple()方法，在一个元组中列出它的Cell 对象。
for rowOfCellObjects in sheet['A1':'C3']:
    for cellObj in rowOfCellObjects:
        print(cellObj.coordinate, cellObj.value)
    print('--- END OF ROW ---')
```

Excel电子表格

- 人口普查
  - ✓ #readCensusExcel

处理PDF和Word文档



处理PDF和Word文档

- PDF和Word文档是二进制文件，所以它们比纯文本文件要复杂得多。除了文本之外，它们还保存了许多字体、颜色和布局信息
- PDF表示Portable Document Format，使用.pdf文件扩展名
- 用于处理PDF的模块是PyPDF2：import PyPDF2
- PyPDF2没有办法从PDF文档中提取图像、图表或其他媒体，但它可以提取文本，并将文本返回为Python字符串

处理PDF和Word文档

- 从PDF提取文本
  - ✓ #pdf1

```
import PyPDF2
pdfFileObj = open('meetingminutes.pdf', 'rb')
pdfReader = PyPDF2.PdfFileReader(pdfFileObj)
pdfReader.numPages
pageObj = pdfReader.getPage(0)#页码
pageObj.extractText()
```

处理PDF和Word文档

- 解密PDF
  - ✓ 某些PDF文档有加密功能，以防止别人阅读，只有在打开文档时提供口令才能阅读
  - ✓ #pdf2

```
import PyPDF2
pdfReader = PyPDF2.PdfFileReader(open('encrypted.pdf', 'rb'))
pdfReader.isEncrypted
#pdfReader.getPage(0)
pdfReader.decrypt('rosebud')
pageObj = pdfReader.getPage(0)
pageObj.extractText()
```

处理PDF和Word文档

- 加密PDF
  - ✓ PdfFileWriter对象也可以为PDF 文档进行加密
  - ✓ #pdf3

```
import PyPDF2
pdfFile = open('meetingminutes.pdf', 'rb')
pdfReader = PyPDF2.PdfFileReader(pdfFile)
pdfWriter = PyPDF2.PdfFileWriter()
#创建一个新的PDF 文件
for pageNum in range(pdfReader.numPages):
    pdfWriter.addPage(pdfReader.getPage(pageNum))#调用getPage(), 取得Page对象
pdfWriter.encrypt('AFORESAID')
resultPdf = open('encryptedfile.pdf', 'wb')
pdfWriter.write(resultPdf)
resultPdf.close()
pdfFile.close()
```

处理PDF和Word文档

- 从多个PDF中合并选择的页面
  - ✓ 将几十个PDF文件合并成一个PDF文件。每一个文件都有一个封面作为第一页，但你不希望合并后的文件中重复出现这些封面
  - ✓ #combinePdfs

处理PDF和Word文档

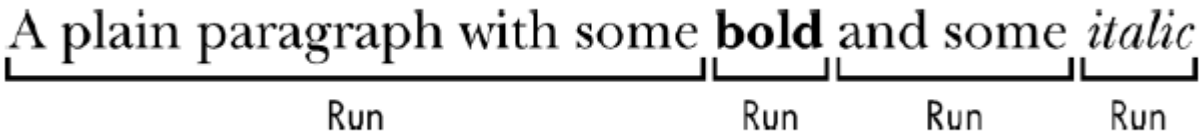
- 暴力PDF口令破解
  - ✓ #暴力破解

处理PDF和Word文档

- 利用python-docx模块，Python可以创建和修改Word文档，它带有.docx文件扩展名
- 运行pip install python-docx，可以安装该模块

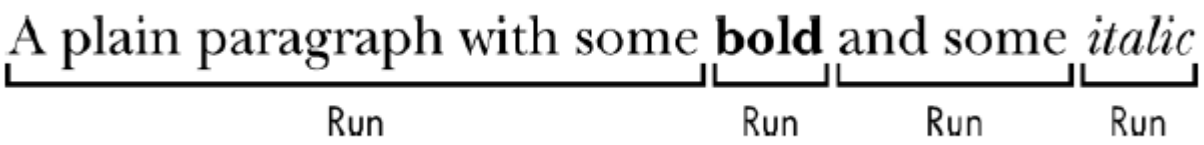
处理PDF和Word文档

- 和纯文本相比，.docx文件有很多结构。这些结构在python-docx中用3种不同的类型来表示
- 在最高一层，Document对象表示整个文档。Document对象包含一个Paragraph 对象的列表，表示文档中的段落
- 每个Paragraph 对象都包含一个Run对象的列表



处理PDF和Word文档

- 每个Paragraph对象都包含一个Run对象的列表



- Word 文档中的文本不仅仅是字符串。它包含与之相关的字体、大小、颜色和其他样式信息。在Word中，样式是这些属性的集合。一个Run对象是相同样式文本的延续。当文本样式发生改变时，就需要一个新的Run对象

处理PDF和Word文档

- 读取Word文档
  - ✓ #world1

处理PDF和Word文档

- 从.docx文件中取得完整的文本读取Word文档，
- 利用getText()函数。它接受一个.docx文件名，返回其中文本的字符串
  - ✓ #world2

处理PDF和Word文档

- 要创建自己的.docx文件，就调用docx.Document()，返回一个新的、空白的Word Document对象，Documen对象的add_paragraph()方法将一段新文本添加到文档中，并返回添加的Paragraph对象的引用。在添加完文本之后，向Document对象的save()方法传入一个文件名字符串，将Document 对象保存到文件
  - ✓ #world3

处理PDF和Word文档

- 可以用新的段落文本，再次调用add_paragraph()方法，添加段落
- 或者，要在已有段落的末尾添加文本，可以调用Paragraph 对象的add_run()方法，向它传入一个字符串
  - ✓ #world3

处理PDF和Word文档

- 文本信息不仅仅是纯文本文件，实际上，很有可能更经常遇到的是PDF和Word文档
- 可以利用PyPDF2模块来读写PDF文档。遗憾的是，从PDF文档读取文本并非总是能得到完美转换的字符串，因为PDF 文档的格式很复杂，某些PDF可能根本读不出
- Word文档更可靠，可以用python-docx模块来读取。可以通过Paragraph 和Run对象来操作Word文档中的文本

谢谢