









Python

数据分析及可视化-6

文件及文件夹操作



本章学习目标

-  熟练掌握内置函数open()的应用
-  理解字符串编码格式对文本文件操作的影响
-  熟练掌握上下文管理语句with的用法
-  了解标准库json对JSON文件的读写方法
-  了解扩展库python-docx、openpyxl、python-pptx对Office文档的操作
-  了解掌握os库、shutil库的使用





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。它有两种类型：**文本文件**和**二进制文件**。

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，但**打开后的操作**不同。





文件概述

理解文本文件和二进制文件的区别

```
>>> textf=open("text1.txt",'rt')
>>> textf.read()
'2020年春季合肥工业大学宣城校区 '
>>> textf.close()

>>> bitsf=open("text1.txt",'rb')
>>> bitsf.read()
b'2020\xc4\xea\xb4\xba\xbc\xbe\xba\xcf\xb7\xca\xb9\xa4\xd2\xb5\xb4\xf3\xd1\xa7
\xd0\xfb\xb3\xc7\xd0\xa3\xc7\xf8 '

>>> bitsf.seek(0)
0
>>> bitsf.read().decode("gbk")
'2020年春季合肥工业大学宣城校区 '
>>> bitsf.close()
```

#将文件位置指针移到头部

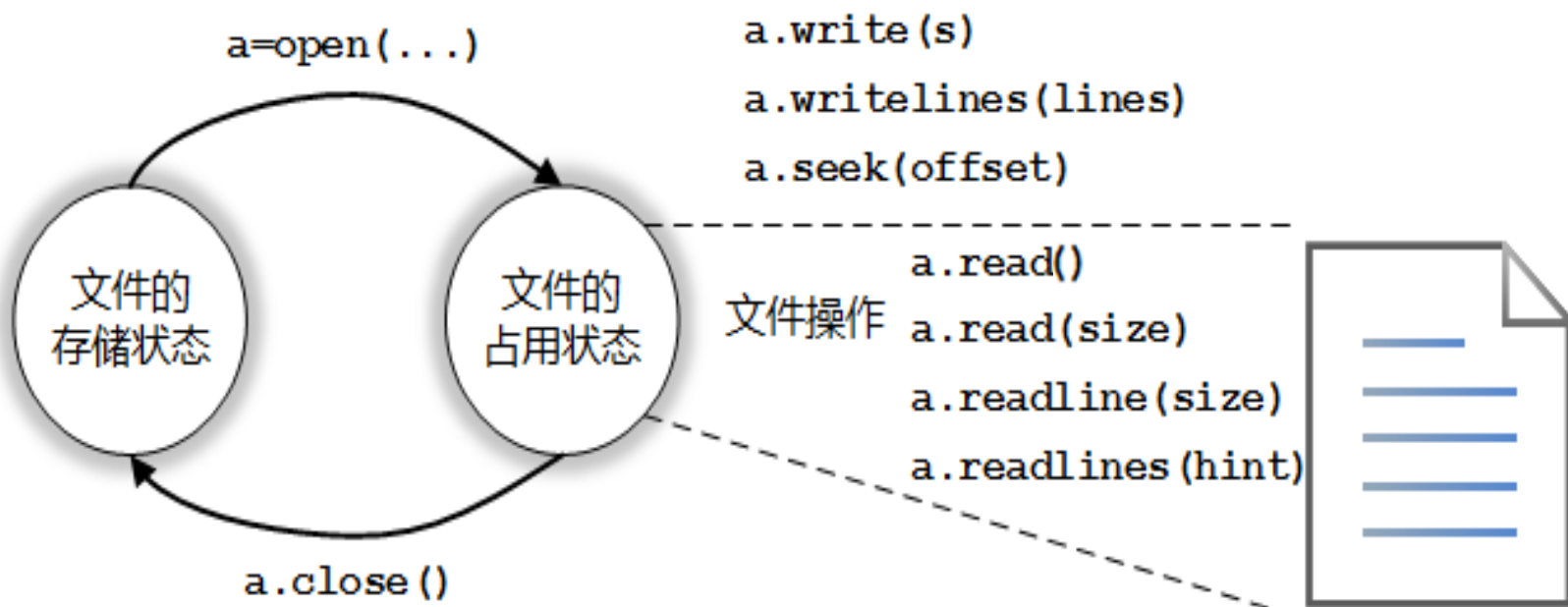
#使用gbk编码进行解码





文件的打开与关闭

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”





文件操作基础

open()

Python内置函数open()使用指定的模式打开指定文件并创建文件对象，该函数完整的用法如下：

```
open(file, mode='r', buffering=-1, encoding=None, errors=None,
      newline=None, closefd=True, opener=None)
```

模式	说明
r	读模式（默认模式，可省略），如果文件不存在，抛出异常
w	写模式，如果文件已存在，先清空原有内容；如果文件不存在，创建新文件
x	写模式，创建新文件，如果文件已存在则抛出异常FileExistsError
a	追加模式，不覆盖文件中原有内容
b	二进制模式（可与r、w、x或a模式组合使用）
t	文本模式（默认模式，可省略）
+	读、写模式（可与其他模式组合使用）





文件对象常用方法


方法	功能说明
<code>close()</code>	把缓冲区的内容写入文件，同时关闭文件，释放文件对象
<code>read([size])</code>	从文本文件中读取并返回size个字符，或从二进制文件中读取并返回size个字节，省略size参数表示读取文件中全部内容
<code>readline()</code>	从文本文件中读取并返回一行内容
<code>readlines()</code>	返回包含文本文件中每行内容的列表
<code>seek(cookie, whence=0, /)</code>	定位文件指针，把文件指针移动到相对于whence的偏移量为cookie的位置。其中whence为0表示文件头，1表示当前位置，2表示文件尾。对于文本文件，whence=2时cookie必须为0；对于二进制文件，whence=2时cookie可以为负数
<code>write(s)</code>	把s的内容写入文件，如果写入文本文件则s应该是字符串，如果写入二进制文件则s应该是字节串
<code>writelines(s)</code>	把列表s中的所有字符串写入文本文件，并不在s中每个字符串后面自动增加换行符。也就是说，如果确实想让s中的每个字符串写入文本文件之后各占一行，应由程序员保证每个字符串以换行符结束





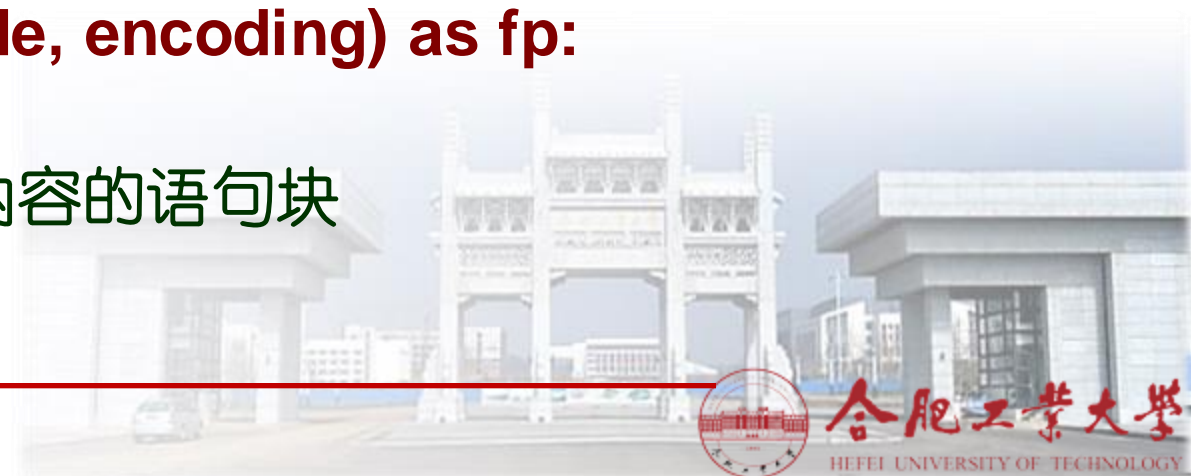
文件操作基础

文件对象常用方法

 在实际开发中，读写文件应优先考虑使用上下文管理语句**with**。关键字**with**可以自动管理资源，不论因为什么原因跳出**with**块，总能保证**文件被正确关闭**。除了用于文件操作，**with**关键字还可以用于数据库连接、网络连接或类似场合。用于文件内容读写时，**with**语句的语法形式如下：

with open(filename, mode, encoding) as fp:

这里写通过文件对象**fp**读写文件内容的语句块





文件对象常用方法

打开一个文本文件，再分别采用字符串和列表两种写入方式写入到两个文本文件中。

```
>>> with open("d:/p_train/tangsil.txt", 'r') as textf:  
    print(textf.read())
```

忆江南 • 江南好
[唐] 白居易
江南好，
风景旧曾谙。
日出江花红胜火，
春来江水绿如蓝，
能不忆江南？

```
>>> with open("tangsil.txt", "r") as oldf,  
    open("new1.txt", 'w') as nf1,  
    open("new2.txt", 'w') as nf2:  
    lines=oldf.readlines()  
    nf1.writelines(lines)  
    oldf.seek(0)  
    for line in oldf:  
        nf2.write(line)
```





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

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：2018年=100。





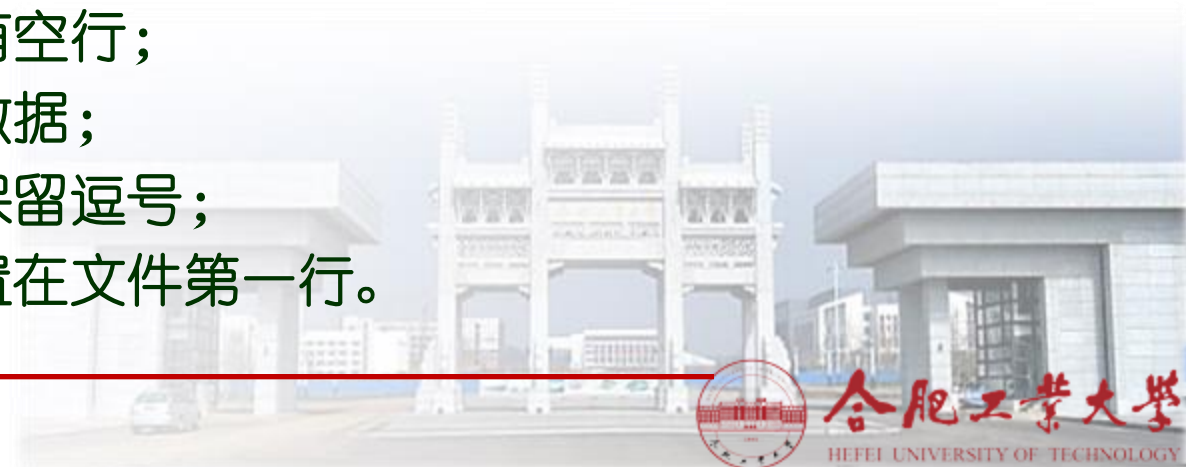
二维数据的存储格式

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。



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。





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
```

```
[  
    ['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'],  
]
```

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。





低维数据格式化和处理

二维数据的表示和读写

导入CSV格式数据到列表

```
>>> import csv
>>> f1=open("2018price.csv",'r')
>>> csv_r=csv.reader(f1)
>>> for cs in csv_r:
    print(cs)
```

格式化打印输出

```
>>> f1=open("2018price.csv",'r')
>>> csv_r=csv.reader(f1)
>>> for cs in csv_r:
    for ch in cs:
        print(' {} \t'.format(ch),end="")
    print("")
```

```
['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']
```

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





低维数据格式化和处理

二维数据的表示和读写

- 使用csv模块，可以通过csv的reader方法，返回一个迭代读对象，每次迭代可返回文件中一行数据的列表。如不使用csv模块，如何获取类似的结果？

```
>>> f1=open("2018price.csv",'r')
>>> for line in f1:
    line=line.replace('\n',"")
    print(line.split(','))
```

```
['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']
```

- 一维数据写入csv文件

```
>>> f0=open("2018p.csv",'w')
>>> csv_w=csv.writer(f0)
>>> ls=['北京', '101.5', '120.7', '121.4']
>>> csv_w.writerow(ls)
22
>>> f0.close()
```

#根据csv文本文件对象创建并返回写对象。

#写对象将一维列表数据写入目标文件。

#交互模式下会显示出成功写入文件的字符数量





二维数据的表示和读写

编写程序，模拟生成某商场2019年3月1日开始连续100天试营业期间的营业额数据并写入CSV文件。文件中共两列，第一列为日期，第二列为营业额，文件第一行为表头或字段名称。假设该商场第一天营业额基数为50000元，每天增加200元，除此之外每天还会随机增减-100到100元不等。

```
>>> import csv, random, datetime
>>> fw=open("2019test.csv", "w", newline="")
>>> csv_w=csv.writer(fw)
>>> dates=datetime.date(2019, 3, 1)
>>> sales=50000
>>> csv_w.writerow(['日期', '销售金额'])
>>> for i in range(100):
>>>     csv_w.writerow([dates, sales])
>>>     dates=dates+datetime.timedelta(days=1)
>>>     sales=sales+200+random.randint(-100, 100)
>>> fw.close()
```

#查看生成的csv文件

```
>>> fr=open("2019test.csv", "r")
>>> csv_r=csv.reader(fr)
>>> for cr in csv_r:
>>>     print(cr)
>>> fr.close()
```





json数据格式化和处理

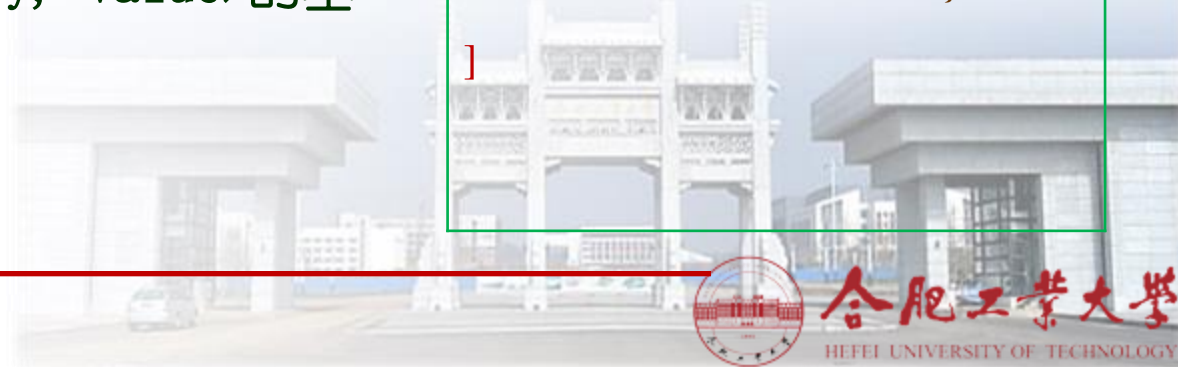
高维数据组织的维度

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。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，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"

```
"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数据格式化和处理

Json数据特点



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(字典)；
- 方括号用于保存键值对数据组成的数组(列表)。

```
"本书作者" : [  
    {"姓氏" : "嵩",  
      "名字" : "天",  
      "单位" : "北京理工大学",  
      "电话" : [{"单位电话" : "010-64895321"},  
                 {"手机" : "13901011432"}] },  
    {"姓氏" : "礼",  
      "名字" : "欣",  
      "单位" : "北京理工大学",  
      "电话" : [{"单位电话" : "010-64895321"},  
                 {"手机" : "13901015678"}] }  
]
```



json数据格式化和处理

Json数据解析



json库两类函数：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。

`dumps()` 和 `loads()` 分别对应编码和解码功能

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





json数据格式化和处理

Json数据解析

 Python格式与JSON格式的相互转换。

```
>>> import json
>>> dt = {'b':2,'c':4,'a':6}
>>> s1=json.dumps(dt)
>>> s1
'{"b": 2, "c": 4, "a": 6}'
>>> s2=json.dumps(dt, indent=4)
>>> s2
'{\n    "b": 2, \n    "c": 4, \n    "a": 6\n}'
>>> print(s2)
{
    "b": 2,
    "c": 4,
    "a": 6
}
```

```
>>> type(s2)
<class 'str'>

>>> d1=json.loads(s2)
>>> d1
{'b': 2, 'c': 4, 'a': 6}
>>> type(d1)
<class 'dict'>
```





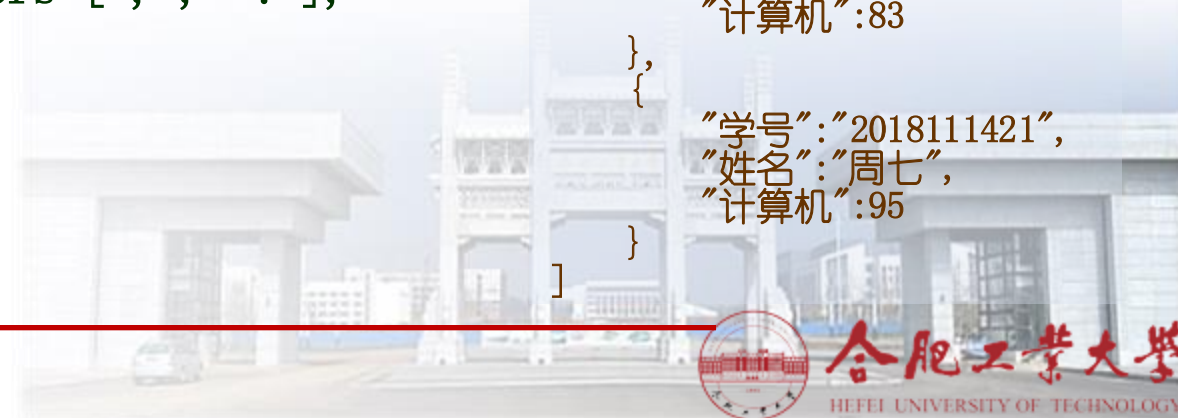
json数据格式化和处理

Json数据解析

■ 编写程序，把包含若干学生信息的列表写入JSON文件，然后再读取并输出这些信息。

```
>>> import json
>>> stinfo=[{"学号":"2018111234","姓名":"张三","计算机":91},
            {"学号":"2018112234","姓名":"王五","计算机":79},
            {"学号":"2018111435","姓名":"李四","计算机":89},
            {"学号":"2018114312","姓名":"赵六","计算机":83},
            {"学号":"2018111421","姓名":"周七","计算机":95} ]
>>> with open('stinfo.json', 'w') as fp:
    json.dump(stinfo, fp, indent=4, separators=(',', ':'),
              ensure_ascii=False)
>>> with open('stinfo.json') as fp:
    information = json.load(fp)
    for info in information:
        print(info)
```

```
[
  {
    "学号": "2018111234",
    "姓名": "张三",
    "计算机": 91
  },
  {
    "学号": "2018112234",
    "姓名": "王五",
    "计算机": 79
  },
  {
    "学号": "2018111435",
    "姓名": "李四",
    "计算机": 89
  },
  {
    "学号": "2018114312",
    "姓名": "赵六",
    "计算机": 83
  },
  {
    "学号": "2018111421",
    "姓名": "周七",
    "计算机": 95
  }
]
```





Office 文件操作实例

docx库

编写程序，读入一个Word文档(.docx)，显示输出每一段落的内容，并在结尾空一行后加上一段“此文档已阅！”信息后保存为一备份文档。

```
>>> import docx
>>> doc=docx.Document('word.docx')
>>> for p in doc.paragraphs:
    if p.text is not None:
        p.text
    else:
        pass
#从段落列表中遍历每一段落

>>> parap=doc.add_paragraph("")
>>> parap=doc.add_paragraph("此文档已阅读！")
>>> doc.save("wordbak.docx")
```

忆江南·江南好

[唐] 白居易

江南好，
风景旧曾谙。
日出江花红胜火，
春来江水绿如蓝，
能不忆江南？

这几句词是说，春日，朝阳照耀下的江畔花朵，红得胜似烈火；江水碧绿的颜色，只有蓝草可以相比。这一切，怎能不撩起我对江南的回忆？诗人用对比、夸张的手法，对江南水乡的诱人春色进行了形象的描绘，红绿相映，光彩夺目，印象强烈，引人入胜。

'忆江南·江南好'
'[唐] 白居易'

'江南好，'
'风景旧曾谙。'
'日出江花红胜火，'
'春来江水绿如蓝，'
'能不忆江南？'

'这几句词是说，春日，朝阳照耀下的江畔花朵，红得胜似烈火；江水碧绿的颜色，只有蓝草可以相比。这一切，怎能不撩起我对江南的回忆？诗人用对比、夸张的手法，对江南水乡的诱人春色进行了形象的描绘，红绿相映，光彩夺目，印象强烈，引人入胜。'





Office 文件操作实例

docx库

 编写程序，读入一个Word文档(.docx)，显示输出表格内容。

```
>>> import docx
>>> doc=docx.Document('table.docx')
>>> print(doc.paragraphs[0].text)
>>> for table in doc.tables:
    for row in table.rows:
        for cell in row.cells:
            print(cell.text,end=' \t')
        print("")
```

青年歌手大奖赛决赛选手基本情况表

编号	姓名	性别	年龄	唱法类型	选送单位
1	马丽	女	21	美声	市教育局
2	刘绪艳	女	22	流行	市教育学院
3	付建丽	女	19	流行	市地方税务局
4	魏翠香	女	23	美声	天地公司
5	赵晓英	女	19	美声	市电台
6	刘宝娜	女	21	流行	市有线电视台
7	郑会锋	男	20	美声	市财政局
8	申永琴	女	23	民族	市幼师学校
9	许宏伟	男	22	流行	市技术监督局
10	张琪	女	21	美声	中天公司
11	李彦宾	男	25	民族	市电视台
12	洪峰	男	24	美声	市有线电视台
13	卞永辉	男	23	流行	市卫生局
15	韩朝辉	男	22	民族	市行政学院





Office 文件操作实例

Openpyxl库

编写程序，读入一个Excel文档(.xlsx)，输出其单元格内容。

```
>>> import openpyxl
>>> xls=openpyxl.load_workbook('test.xlsx')
>>> s1=xls.worksheets[0]
>>> for row in s1.rows:
    for cell in row:
        print(cell.value,end="\t")
    print("")
```

```
>>> s1['B17']="此表格已打印!"
>>> xls.save('excelbak.xlsx')
```

	A	B	C	D	E	F
1	青年歌手大奖赛决赛选手基本情况表					
2	编号	姓名	性别	年龄	唱法类型	选送单位
3	1	马丽	女	21	美声	市教育局
4	2	刘绪艳	女	22	流行	市教育学院
5	3	付建丽	女	19	流行	市地方税务局
6	4	魏翠香	女	23	美声	天地公司
7	5	赵晓英	女	19	美声	市电台
8	6	刘宝娜	女	21	流行	市有线电视台
9	7	郑会锋	男	20	美声	市财政局
10	8	申永琴	女	23	民族	市幼师学校
11	9	许宏伟	男	22	流行	市技术监督局
12	10	张琪	女	21	美声	中天公司
13	11	李彦宾	男	25	民族	市电视台
14	12	洪峰	男	24	美声	市有线电视台
15	13	卞永辉	男	23	流行	市卫生局
16	15	韩朝辉	男	22	民族	市行政学院
17						





Office 文件操作实例

pptx库

编写程序，读入一个ppt文档(.pptx)，输出其中一幻灯片中的信息内容。

```
>>> import pptx
>>> ppt=pptx.Presentation('point.pptx')
>>> len(ppt.slides)
7
>>> slid=ppt.slides[0]
>>> len(slid.shapes)
3
>>> s1=slid.shapes[0]
>>> s1.shape_type
14
>>> for s in slid.shapes:
    if s.text is None:
        pass
    else:
        s.text
```



```
'Python\数据处理及可视化'
'合肥工业大学 冷金麟\nlengjl@hfut.edu.cn\n2020.3'
'课程引导'
```





Office文件操作实例

pptx库

编写程序，读入一个ppt文档(.pptx)，输出其中一幻灯片中的信息内容。

```
>>> slid=ppt.slides[5]
>>> len(slid.shapes)
4
>>> s1=slid.shapes[2]
>>> s1.shape_type
19
>>> len(s1.table.rows)
6
>>> len(s1.table.rows[0].cells)
5
>>> s1.table.rows[0].cells[0].text
'编程语言'
```

为什么要学习Python?



不同编程语言的初心和适用对象

编程语言	教学内容	计算思维	解决问题	适用对象
C	指针、内存、数据类型	抽象计算机系统结构	性能	计算机类专业
Java	对象、跨平台、运行时	抽象主客体关系	跨平台	软件类专业
C++	对象、多态、继承	抽象主客体关系	大规模程序	计算机类专业
VB	对象、按钮、文本框	抽象交互逻辑	桌面应用	不确定
Python	编程逻辑、第三方库	抽象问题求解和方法	各类问题	所有专业

各编程语言所处历史时期和使命不同，Python是**计算时代演进**的选择！

```
>>> for row in s1.table.rows:
      for cell in row.cells:
          print(cell.text,end=' \t')
      print("")
```





OS模块

方法	功能说明
<code>chdir(path)</code>	把path设为当前工作目录
<code>curdir</code>	当前文件夹
<code>environ</code>	包含系统环境变量和值的字典
<code>extsep</code>	当前操作系统所使用的文件扩展名分隔符
<code>get_exec_path()</code>	返回可执行文件的搜索路径
<code>getcwd()</code>	返回当前工作目录
<code>listdir(path)</code>	返回path目录下的文件和目录列表





OS 模块

方法	功能说明
<code>remove(path)</code>	删除指定的文件，要求用户拥有删除文件的权限，并且文件没有只读或其他特殊属性
<code>rename(src, dst)</code>	重命名文件或目录，可以实现文件的移动，若目标文件已存在则抛出异常，不能跨越磁盘或分区
<code>replace(old, new)</code>	重命名文件或目录，若目标文件已存在则直接覆盖，不能跨越磁盘或分区
<code>scandir(path='.')</code>	返回包含指定文件夹中所有DirEntry对象的迭代对象，遍历文件夹时比listdir()更加高效
<code>sep</code>	当前操作系统所使用的路径分隔符
<code>startfile(filepath [, operation])</code>	使用关联的应用程序打开指定文件或启动指定应用程序
<code>system()</code>	启动外部程序





OS 模块

```
>>> import os
>>> import os.path
>>> #rename()可以实现文件的改名和移动
>>> os.rename('C:\\dfg.txt', 'D:\\test2.txt')
>>> [fname for fname in os.listdir('.')
      if fname.endswith(('.pyc', '.py', '.pyw'))] #结果略
>>> #返回当前工作目录
>>> os.getcwd()
'C:\\Python35'
>>> os.mkdir(os.getcwd()+ '\\temp')
>>> os.chdir(os.getcwd()+ '\\temp')
>>> os.getcwd()
'C:\\Python35\\temp'
```

#创建目录

#改变当前工作目录





OS 模块

```
>>> os.mkdir(os.getcwd()+'\\test')
```

```
>>> os.listdir('.')
```

```
['test']
```

```
>>> os.rmdir('test')
```

```
>>> os.listdir('.')
```

```
[]
```

```
>>> os.environ.get('path')
```

```
>>> import time
```

```
>>> time.strftime('%Y-%m-%d %H:%M:%S',
```

```
time.localtime(os.stat('yilaizhuru2.py').st_ctime))
```

```
'2016-10-18 15:58:57'
```

```
>>> os.startfile('notepad.exe')
```

#删除目录

#获取系统变量path的值

#查看文件创建时间

#启动记事本程序





【案例】 使用递归法遍历指定目录下所有子目录和文件。

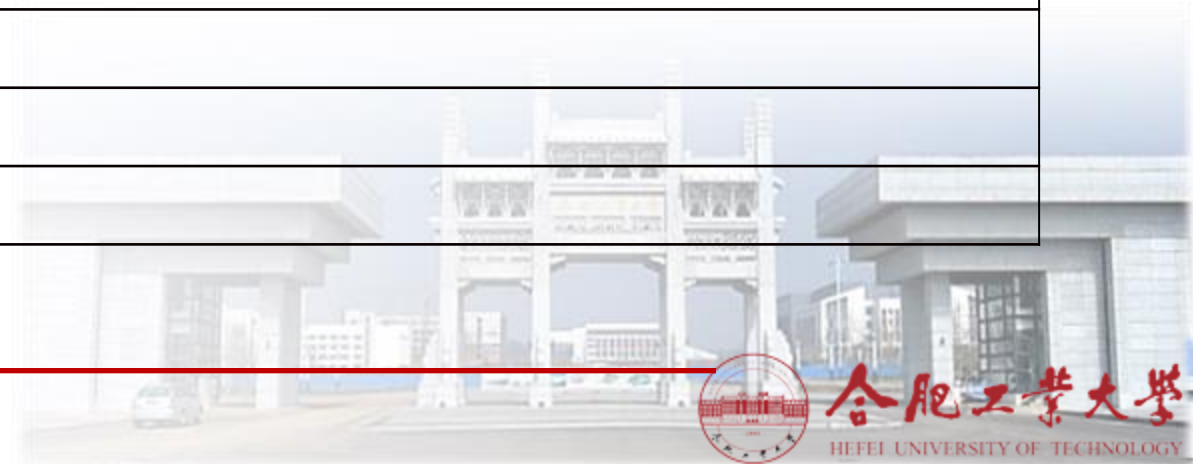
```
from os import listdir
from os.path import join, isfile, isdir
def listDirDepthFirst(directory):
    '''深度优先遍历文件夹'''
    #遍历文件夹，如果是文件就直接输出
    #如果是文件夹，就输出显示，然后递归遍历该文件夹
    for subPath in listdir(directory):
        path = join(directory, subPath)
        if isfile(path):
            print(path)
        elif isdir(path):
            print(path)
            listDirDepthFirst(path)
```





os.path 模块

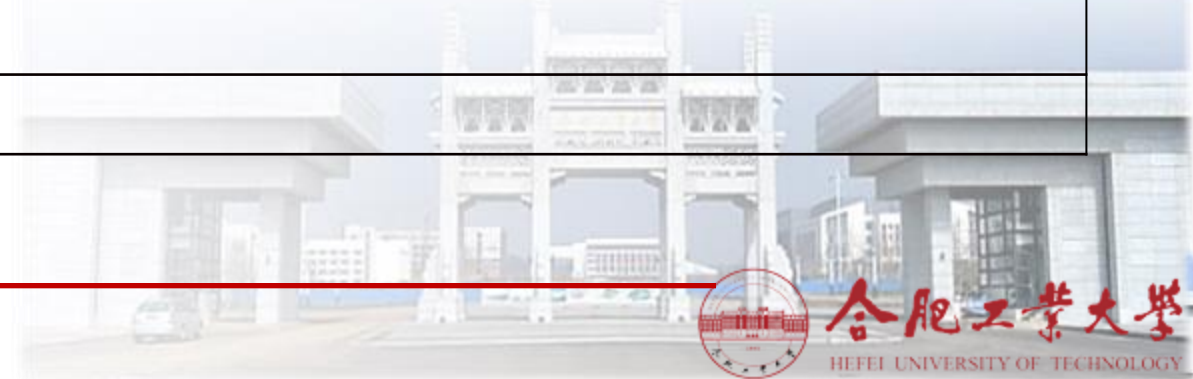
方法	功能说明
<code>abspath(path)</code>	返回给定路径的绝对路径
<code>basename(path)</code>	返回指定路径的最后一个组成部分
<code>commonpath(paths)</code>	返回给定的多个路径的最长公共路径
<code>commonprefix(paths)</code>	返回给定的多个路径的最长公共前缀
<code>dirname(p)</code>	返回给定路径的文件夹部分
<code>exists(path)</code>	判断文件是否存在
<code>getatime(filename)</code>	返回文件的最后访问时间
<code>getctime(filename)</code>	返回文件的创建时间
<code>getmtime(filename)</code>	返回文件的最后修改时间
<code>getsize(filename)</code>	返回文件的大小





os.path 模块

方法	功能说明
<code>isabs(path)</code>	判断path是否为绝对路径
<code>isdir(path)</code>	判断path是否为文件夹
<code>isfile(path)</code>	判断path是否为文件
<code>join(path, *paths)</code>	连接两个或多个path
<code>realpath(path)</code>	返回给定路径的绝对路径
<code>relpath(path)</code>	返回给定路径的相对路径，不能跨越磁盘驱动器或分区
<code>samefile(f1, f2)</code>	测试f1和f2这两个路径是否引用的同一个文件
<code>split(path)</code>	以路径中的最后一个斜线为分隔符把路径分隔成两部分，以元组形式返回
<code>splitext(path)</code>	从路径中分隔文件的扩展名
<code>splitdrive(path)</code>	从路径中分隔驱动器的名称





os.path 模块

```
>>> path='D:\\mypython_exp\\new_test.txt'
>>> os.path.dirname(path)
'D:\\mypython_exp'
>>> os.path.basename(path)
'new_test.txt'
>>> os.path.split(path)
('D:\\mypython_exp', 'new_test.txt')
>>> os.path.split('')
('', '')
>>> os.path.split('C:\\\\windows')
('C:\\\\', 'windows')
>>> os.path.split('C:\\\\windows\\')
('C:\\\\windows', '')
>>> os.path.splitdrive(path)
('D:', '\\mypython_exp\\new_test.txt')
>>> os.path.splitext(path)
('D:\\mypython_exp\\new_test', '.txt')
```

#返回路径的文件夹名

#返回路径的最后一个组成部分

#切分文件路径和文件名

#切分结果为空字符串

#以最后一个斜线为分隔符

#切分驱动器符号

#切分文件扩展名





10.3 shutil 模块

方法	功能说明
<code>copy(src, dst)</code>	复制文件，新文件具有同样的文件属性，如果目标文件已存在则抛出异常
<code>copy2(src, dst)</code>	复制文件，新文件具有原文件完全一样的属性，包括创建时间、修改时间和最后访问时间等等，如果目标文件已存在则抛出异常
<code>copyfile(src, dst)</code>	复制文件，不复制文件属性，如果目标文件已存在则直接覆盖
<code>copyfileobj(fsrc, fdst)</code>	在两个文件对象之间复制数据，例如 <code>copyfileobj(open('123.txt'), open('456.txt', 'a'))</code>
<code>copymode(src, dst)</code>	把src的模式位（mode bit）复制到dst上，之后二者具有相同的模式
<code>copystat(src, dst)</code>	把src的模式位、访问时间等所有状态都复制到dst上
<code>copytree(src, dst)</code>	递归复制文件夹
<code>disk_usage(path)</code>	查看磁盘使用情况
<code>move(src, dst)</code>	移动文件或递归移动文件夹，也可以给文件和文件夹重命名
<code>rmtree(path)</code>	递归删除文件夹
<code>make_archive(base_name, format, root_dir=None, base_dir=None)</code>	创建tar或zip格式的压缩文件
<code>unpack_archive(filename, extract_dir=None, format=None)</code>	解压缩压缩文件





shutil 模块

- 下面的代码演示了如何使用标准库shutil的copyfile()方法复制文件。

```
>>> import shutil #导入shutil模块
>>> shutil.copyfile('C:\\dir.txt', 'C:\\dir1.txt') #复制文件
```

- 下面的代码将C:\Python35\Dlls文件夹以及该文件夹中所有文件压缩至D:\a.zip文件：

```
>>> shutil.make_archive('D:\\a', 'zip', 'C:\\Python35', 'Dlls')
'D:\\a.zip'
```

- 下面的代码将刚压缩得到的文件D:\a.zip解压缩至D:\a_unpack文件夹：

```
>>> shutil.unpack_archive('D:\\a.zip', 'D:\\a_unpack')
```

- 下面的代码使用shutil模块的方法删除刚刚解压缩得到的文件夹：

```
>>> shutil.rmtree('D:\\a_unpack')
```





shutil 模块

- 下面的代码使用shutil的copytree()函数递归复制文件夹，并忽略扩展名为pyc的文件和以“新”字开头的文件和子文件夹：

```
>>> from shutil import copytree, ignore_patterns  
>>> copytree('C:\\python35\\test', 'D:\\des_test',  
            ignore=ignore_patterns('*.pyc', '新*'))
```





综合案例解析

【案例】 把指定文件夹中的所有文件名批量随机化，保持文件类型不变。

```
from string import ascii_letters
from os import listdir, rename
from os.path import splitext, join
from random import choice, randint
```

```
def randomFilename(directory):
    for fn in listdir(directory):
        #切分，得到文件名和扩展名
        name, ext = splitext(fn)
        n = randint(5, 20)
        #生成随机字符串作为新文件名
        newName = ''.join((choice(ascii_letters) for i in range(n)))
        #修改文件名
        rename(join(directory, fn), join(directory, newName+ext))
```

```
randomFilename('C:\\test')
```





综合案例解析

【案例】 编写程序，统计指定文件夹大小以及文件和子文件夹数量。

```
import os
totalSize = 0
fileNum = 0
dirNum = 0

def visitDir(path):
    global totalSize
    global fileNum
    global dirNum
    for lists in os.listdir(path):
        sub_path = os.path.join(path, lists)
        if os.path.isfile(sub_path):
            fileNum = fileNum+1
            totalSize = totalSize+os.path.getsize(sub_path)
        elif os.path.isdir(sub_path):
            dirNum = dirNum+1
            visitDir(sub_path)
```

#统计文件数量
#统计文件总大小
#统计文件夹数量
#递归遍历子文件夹



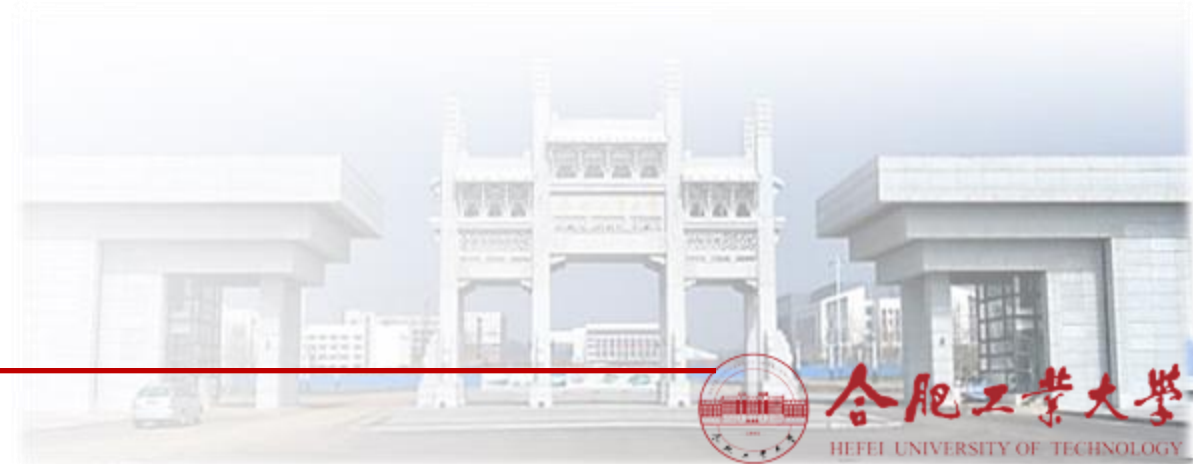


综合案例解析

```
def main(path):  
    if not os.path.isdir(path):  
        print('Error:', path, ' is not a directory or does not exist.')  
        return  
    visitDir(path)
```

```
def sizeConvert(size):  
    K, M, G = 1024, 1024**2, 1024**3  
    if size >= G:  
        return str(size/G)+'G Bytes'  
    elif size >= M:  
        return str(size/M)+'M Bytes'  
    elif size >= K:  
        return str(size/K)+'K Bytes'  
    else:  
        return str(size)+'Bytes'
```

#单位换算

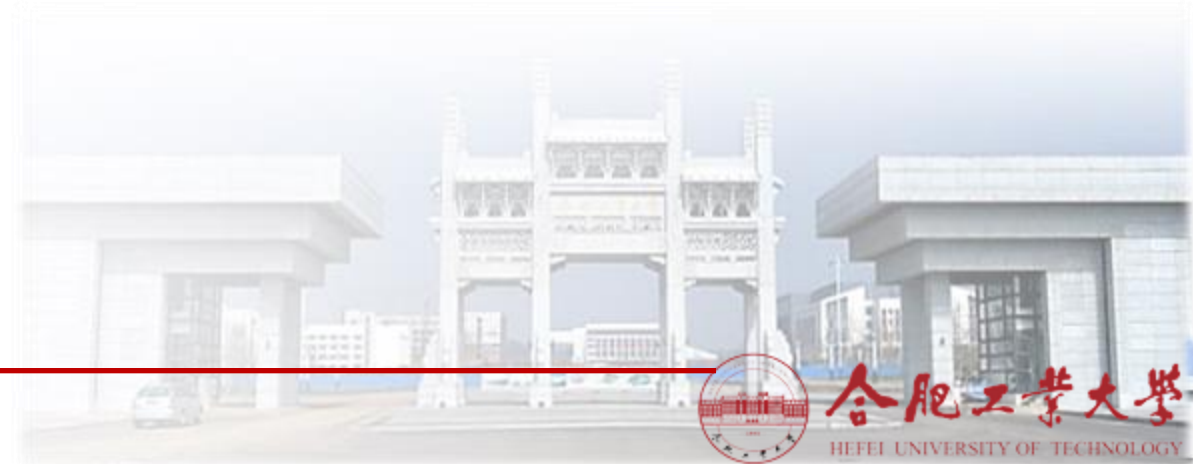


合肥工业大学
HEFEI UNIVERSITY OF TECHNOLOGY



10.4 综合案例解析

```
def output(path):  
    print('The total size of '+path+' is:'+sizeConvert(totalSize)  
          +'('+str(totalSize)+' Bytes)')  
    print('The total number of files in '+path+' is:',fileNum)  
    print('The total number of directories in '+path+' is:',dirNum)  
  
if __name__=='__main__':  
    path = r'd:\idapro6.5plus'  
    main(path)  
    output(path)
```





综合案例解析

【案例】 编写程序，递归删除指定文件夹中指定类型的文件和大小为0的文件。

```
from os.path import isdir, join, splitext
from os import remove, listdir, chmod, stat
filetypes = ('.tmp', '.log', '.obj', '.txt')      #指定要删除的文件类型
def delCertainFiles(directory):
    if not isdir(directory):
        return
    for filename in listdir(directory):
        temp = join(directory, filename)
        if isdir(temp):
            delCertainFiles(temp)                #递归调用
        elif splitext(temp)[1] in filetypes or stat(temp).st_size==0:
            chmod(temp, 0o777)                   #修改文件属性，获取删除权限
            remove(temp)                          #删除文件
            print(temp, ' deleted....')
delCertainFiles(r'C:\test')
```





祝学习进步!