# 文件操作

王肖燕

# 内容介绍

一．文件、文件对象和常用操作

二．文本文件操作

三．Excel文件操作

四．文件夹操作

五．举例

# 文件

- 是长久保存信息并允许重复使用和反复修改的重要方式，同时也是信息交换的重要途径。
- 数据库文件、图像文件、音频和视频文件、可执行文件、Office文档、动态链接库文件等，都以文件的形式存储在不同形式的存储设备（如磁盘、U盘、光盘、云盘等）
- 按文件中数据的组织形式可以把文件分为文本文件和二进制文件

# 文本文件

- 存储的是常规字符串，由若干文本行组成，通常每行以换行符'\n'结尾
- 常规字符串：指记事本之类的文本编辑器能正常显示和编辑，人类能够直接阅读和理解的字符串，如英文字母、汉字、数字字符串
- 在windows平台中，扩展名为txt, log, ini的文件都属于文本文件，可以使用字处理软件（如记事本、笔记本）进行编辑

# 二进制文件

- 常见的图形图像文件、音频和视频文件、可执行文件、资源文件、各种数据库文件、各类Office文档等都属于二进制文件
- 把信息以字节串（Bytes）进行存储，无法用记事本或其他普通字处理软件直接进行编辑，无法被人类直接阅读和理解，需要使用对应的软件进行解码后读取、显示、修改或执行

# 文件对象常用方法与属性

- 无论是文本文件还是二进制文件，其操作流程基本都是一致的

  ① 首先，打开文件并创建文件对象

  ② 其次，通过该文件对象对文件内容进行读取、写入、删除、修改等操作

  ③ 最后，关闭并保存文件内容

# 文件对象常用方法与属性

- Python内置了文件对象，通过open()函数即可以指定模式打开指定文件并创建文件对象，该函数用法为：

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

# Open函数的主要参数

- file: 指定要打开或创建的文件名称，如果该文件不在当前目录中，则需要指定完整路径，为了减少完整路径中“\”符号的输入，可以使用原始字符串
- mode: 指定打开文件后的处理方式，如“只读”、“只写”、“读写”、“追加”、“二进制只读”、“二进制读写”等、默认为“文本只读模式”。以不同方式打开文件时，文件指针的初始位置略有不同
  - 例如：以只读和只写模式打开文件时文件指针的初始位置是文件头，而以追加模式打开文件时则文件指针的初始位置为文件尾

# 文件打开模式

| 模式 | 说明 |
| --- | --- |
| r | 读模式（默认模式，可省略），如果文件不存在则抛异常 |
| w | 写模式，如果文件已存在，先清空原有内容 |
| x | 写模式，创建新文件 |
| a | 追加模式，不覆盖文件中原有内容 |
| b | 二进制模式（可与其他模式组合使用） |
| 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） |

# open函数

- 如果执行正常，open函数返回一个可迭代的文件对象，通过该文件对象可以对文件进行读写操作；如果指定文件不存在，访问权限不够、磁盘空间不够或其他原因导致创建文件对象失败，则抛异常
- 以读方式打开文件，并创建与之对应的文件对象
  - f1=open('file1.txt', 'r') 或 f1=open(r'c:\file1.txt', 'r')
- 以写方式打开文件，并创建文件对象
  - f2=open('file2.txt', 'w')
- 当对文件内容操作完后，要关闭文件对象，才能保证修改都确定保存到文件中
  - f1.close()

# 文件对象的常用属性

| 属性 | 说明 |
| --- | --- |
| closed | 判断文件是否关闭，若文件已关闭则返回True |
| mode | 返回文件的打开模式 |
| name | 返回文件的名称 |

# 文件对象的常用方法

| 方法 | 功能说明 |
| --- | --- |
| flush() | 把缓冲区的内容写入文件，但不关闭文件 |
| close() | 把缓冲区的内容写入文件，同时关闭文件，释放文件对象 |
| read([n]) | 从文件中读取n个字符的内容作为结果返回，没有n表示读取全部内容 |
| readline() | 从文本文件中读取一行内容作为结果返回 |
| readlines() | 把文本文件中的每行文本作为一个字符串存入列表，并返回该列表 |
| seek(offset[,n]) | 把文件指针移动到新的位置，offset表示相对于n的位置<br>n=0: 从文件头，n=1: 从当前位置，n=2: 从文件尾，默认为0 |
| tell() | 返回文件指针的当前位置，字节索引，不同于read中的n |
| truncate([size]) | 删除从当前指针位置到文件末尾的内容。如果指定了size，则不论指针在什么位置都只留下前size个字节，其余的删除 |
| write(s) | 把字符串s的内容写入文件 |
| writelines(s) | 把字符串列表写入文件，不添加换行符 |
| writable() | 测试当前文件是否可写 |
| readable() | 测试当前文件是否可读 |

# 文本文件操作

- 例1：向文本文件中写入内容

```
s='Hello world\n文本文件的读取方法\n文本文件的写入方法\n'
f=open('sample1.txt','a+')
f.write(s)
f.close()
```

- 文件操作一般遵循“打开”->”读写”->”关闭”的标准流程，但是如果文件读写操作代码引发了异常，很难保证文件能够被正确关闭
- 使用上下文管理关键字with可以避免此问题，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
- 常用于文件操作、数据库连接、网络通信连接等场合

```
s='Hello world\n文本文件的读取方法\n文本文件的写入方法\n'
with open('sample1.txt','a+') as f:
    f.write(s)
```

# 文本文件操作

- 例2：读取文本文件的内容

```
fp=open('sample1.txt')
print(fp.read(4))
print(fp.read(18))
print(fp.read())
fp.close()
```

# 文本文件操作

- 例3：读取并显示文本文件的所有行

```
with open('sample1.txt') as fp:
    while True:
        line = fp.readline()
        if not line:
            break
        print(line)
```

# 文本文件操作

- 例4：移动文件指针

```
>>> fp=open('sample1.txt','r+')
>>> fp.tell()
0
>>> fp.read(20)
'Hello world\n文本文件的读取方'
>>> fp.seek(13)
13
>>> fp.read(5)
'文本文件的'
>>> fp.seek(13)
13
>>> fp.write('test')
4
>>> fp.flush()
>>> fp.seek(0)
0
>>> fp.read()
'Hello world\ntest文件的读取方法\n文本文件的写入方法\n'
>>> fp.close()
```

# 文本文件操作

- 例5：假设文件data.txt中有若干整数，整数之间使用英文逗号分隔，编写程序读取所有的整数，将其按升序排序后再写入文件文件data_asc.txt中。

```
with open('data.txt','r') as fp:
    data = fp.readlines()
data=[line.strip() for line in data]
data = ','.join(data)
data = data.split(',')
data=[int(item) for item in data]
data.sort()
data = ','.join(map(str,data))
with open('data_asc.txt','w') as fp:
    fp.write(data)
```

# 文本文件操作

- 例6：计算文本文件中最长行的长度和该行的内容

```
with open('sample1.txt') as fp:
    result=[0,'']
    for line in fp:
        t=len(line)
        if t>result[0]:
            result=[t,line]
print(result)
```

# Excel文件操作

- openpyxl读取和修改Excel电子表格文件
- 基本概念
  - 工作簿
  - 工作表
  - 活动工作表
  - 单元格
- 导入模块
  - import openpyxl

# 读取Excel文档

- 打开文档
  - ```
    import openpyxl
    wb=openpyxl.load_workbook('1.xlsx')
    ```
- 取得工作表
  - ```
    print(wb.sheetnames) #工作簿中所有工作表
    st=wb['Sheet1'] #通过名字获得工作表对象
    print(st.title) #工作表的名字
    acs=wb.active    #获得活动的工作表对象
    ```
- 取单元格中的内容
  - ```
    acs['a1'].value #取得活动单元格中的值
    print(acs.cell(1, 2).value) #cell方法
    ```
- 行数、列数
  - ```
    print("活动表行数：",acs.max_row)
    print("活动表列数：",acs.max_column)
    ```

# 读取Excel文档

- 内容遍历
  - 矩形区域

```
for r in acs['a1':'c3']:
    for c in r:
        print(c.coordinate,c.value)
    print()
```

  - 某一列数据

```
for c in list(acs.columns)[1]:
    print(c.value)
```

  - 全部数据

```
mr=acs.max_row
mc=acs.max_column
for i in range(1,mr+1):
    for j in range(1,mc+1):
        print(acs.cell(i,j).value, end="\t")
    print()
```

# 写入Excel

- 创建并保存Excel文档
  - ```
    import openpyxl
    wb=openpyxl.Workbook()
    wsn=wb.active
    wsn.title="表1"
    wb.save("One.xlsx")
    ```
- 创建和删除表
  - ```
    wb.create_sheet(index=0,title="新表1")
    print(wb.sheetnames)
    wb.save("One.xlsx")
    ```

  ```
  wb.remove(wb[wb.sheetnames[0]])
  del wb[wb.sheetnames[0]]
  print(wb.sheetnames)
  wb.save("One.xlsx")
  ```

- 把值写入单元格
  - ```
    acs=wb.active
    acs['a1']="Hello world!"
    print(acs['a1'].value)
    wb.save("One.xlsx")
    ```
    ```

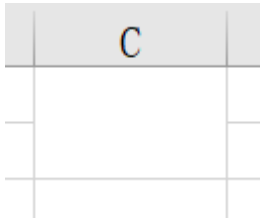
# 写入Excel

- 设置公式

```
import openpyxl as px
wb=px.Workbook()
acs=wb.active
for r in acs['a1':'a3']:
    for c in r:
        c.value=1
acs['a4']='=SUM(A1:A3)'
print(acs['a4'].value)
wb.save('test.xlsx')
```

- 合并和拆分单元格

```
acs=wb.active
acs.merge_cells("c1:c2")
wb.save('test.xlsx')
```




```
acs.unmerge_cells("c1:c2")
wb.save('test.xlsx')
```

# openpyxl的操作

```
import openpyxl
from openpyxl import Workbook

fn=r'D:\test.xlsx' #文件名
wb=Workbook()#创建工作簿
ws=wb.create_sheet(title='你好，世界')#创建工作表
ws['A1']='这是第一个单元格'#单元格赋值
ws['B1']=3.1415926
wb.save(fn)#保存Excel文件
wb=openpyxl.load_workbook(fn)#打开已有的Excel文件
ws=wb.worksheets[1]#打开指定索引的工作表
print(ws['A1'].value)#读取并输出指定单元格的值
ws.append([1,2,3,4,5])#添加一行数据
ws.merge_cells('F2:F3')#合并单元格
ws['F2']="=sum(A2:E2)"#写入公式
for r in range(10,15):
    for c in range(3,8):
        ws.cell(row=r,column=c,value=r*c)#写入单元格数据
wb.save(fn)
```

# 文件夹操作

- 文件夹也叫目录，用于分层保护文件
- os模块
  - Python内置的与操作系统功能和文件系统相关
  - 不同操作系统，结果可能不同
- os.path模块
  - 对目录和文件夹进行操作

# 路径

- 用于定位一个文件或者目录的字符串
- 相对路径
  - import os
  - print(os.getcwd())

```
with open(r"demo\1.txt") as ff:
    pass
```

- 绝对路径
  - 实际路径，不依赖于当前目录
  - Os.path.abspath(r”demo\1.txt”)
  -

```
os.path.abspath(r"demo\1.txt")
'C:\\Users\\Jessica Wang\\Desktop\\demo\\1.txt'
```

# 是否存在

- 判断给定的目录是否存在
  - os.path.exists(path)函数
  - path可以是绝对路径、相对路径、文件
  - 路径存在返回True，否则返回False

```
In [14]: os.path.exists("demo")
Out[14]: True

In [15]: os.path.exists(r"demo\1.txt")
Out[15]: True

In [16]: os.path.exists(r"C:\Users\Jessica Wang\Desktop\demo")
Out[16]: True
```

# 创建目录

- 创建一级目录
  - os.mkdir(path)
  - 已经存在抛出异常
- 创建多级目录
  - os.makedirs(path)

```
In [19]: os.mkdir(r"abc")

In [20]: os.mkdir(r"C:\Users\Jessica Wang\Desktop\abc\def")

In [21]: os.mkdir(r"C:\Users\Jessica Wang\Desktop\abc\def")
Traceback (most recent call last):

  File "<ipython-input-21-237f258681ff>", line 1, in <module>
    os.mkdir(r"C:\Users\Jessica Wang\Desktop\abc\def")

FileExistsError: [WinError 183] 当文件已存在时，无法创建该文件。
```

```
os.makedirs(r"abc\def\ghi\jkl")

os.makedirs(r"C:\Users\Jessica Wang\Desktop\abc\def\ghi\jkl\mno\pqr\stu")
```

# 删除目录

- Os.rmdir()要删除的目录为空时可删除

```
In [26]: os.rmdir(r"abc\def\ghi\jkl\mno\pqr\stu")

In [27]: os.rmdir(r"abc\def\ghi\jkl\mno")
Traceback (most recent call last):

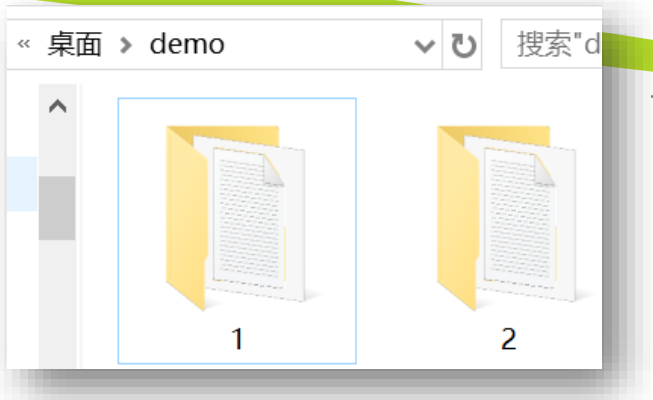
  File "<ipython-input-27-cd22c70516f6>", line 1, in <module>
    os.rmdir(r"abc\def\ghi\jkl\mno")

OSError: [WinError 145] 目录不是空的。: 'abc\\def\\ghi\\jkl\\mno'
```

  - 如果要删除非空目录，使用模块shutil.retree()

- 删除目录不在抛出异常

# 遍历目录




- os.walk(top)
  - top：用于指定要遍历内容的根目录

```
In [36]: import os

In [37]: t=os.walk(r"demo")

In [38]: t
Out[38]: <generator object walk at 0x0000024FC07DE0C8>

In [39]: for p in t:
    ...:     print(p)
    ...:
('demo', ['1', '2'], [])
('demo\\1', [], ['1.txt'])
('demo\\2', [], ['2.txt'])
```

# 文件操作

- 删除文件
  - os.remove(path)
  - 判断文件是否存再删除os.path.exists(path)
- 重命名
  - 文件和目录皆可
  - os.rename(src, dst)
    - src: 要重命名的目录或文件
    - dst: 重命名后的目录或文件

# Jieba库

- jieba是Python中一个重要的第三方中文分词函数库

```
>>>import jieba
>>>jieba.lcut("中国是一个伟大的国家")
['中国', '是', '一个', '伟大', '的', '国家']
```

- jieba库是第三方库，不是安装包自带，需要通过pip指令安装

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

- **原理**：利用一个中文词库，将待分词的内容与分词词库进行比较，通过图结构和动态规划找到最大概率的词组

# Jieba库的常用函数

| 函数 | 描述 |
| --- | --- |
| jieba.cut(s) | 精确模式，返回一个可迭代的数据类型 |
| jieba.cut(s, cut_all=True) | 全模式，输出文本s中所有可能单词 |
| jieba.cut_for_search(s) | 搜索引擎模式，适合搜索引擎建立索引的分词结果 |
| jieba.lcut(s) | 精确模式，返回一个列表类型，建议使用 |
| jieba.lcut(s, cut_all=True) | 全模式，返回一个列表类型，建议使用 |
| jieba.lcut_for_search(s) | 搜索引擎模式，返回一个列表类型，建议使用 |
| jieba.add_word(w) | 向分词词典中增加新词w |

推荐

```
>>>import jieba
>>>jieba.lcut("中华人民共和国是一个伟大的国家") #完整不多余地组成原始文本
['中华人民共和国', '是', '一个', '伟大', '的', '国家']
>>>jieba.lcut("中华人民共和国是一个伟大的国家", cut_all=True)
['中华', '中华人民', '中华人民共和国', '华人', '人民', '人民共和国', '共和', '共和国', '国是', '一个', '伟大', '的', '国家']
>>>jieba.lcut_for_search("中华人民共和国是一个伟大的国家")
['中华', '华人', '人民', '共和', '共和国', '中华人民共和国', '是', '一个', '伟大', '的', '国家']
```

#输出原始文本中能产生所有词，冗余性最大

#首先执行精确模式，然后再对其中的长词进一步切分获得结果

# 文本词频直接统计

```
import jieba
f=open("abc.txt","r")
txt = f.read()
f.close()
words=jieba.lcut(txt)
counts={}
for word in words:
    if len(word)==1:
        continue
    else:
        counts[word]=counts.get(word,0)+1

items=list(counts.items())
items.sort(key=lambda x:x[1],reverse=True)
for i in range(10):
    word,count=items[i]
    print(word+","+str(count))
```

# 文本词频统计——去除

```
import jieba
import os
excludes = {"我们","中国","中华民族","社会主义",
f = open("abc.txt","r")
txt = f.read()
f.close()
words  = jieba.lcut(txt)
counts = {}

for word in words:
    if len(word) == 1:
        continue
    else:
        counts[word] = counts.get(word,0) + 1

for word in excludes:
    del(counts[word])

items = list(counts.items())

items.sort(key=lambda x:x[1], reverse=True)
for i in range(15):
    word, count = items[i]
    print ("{0:<10}{1:>5}".format(word, count))
```

# Thank You!