山东科技大学——测绘与空间信息学院

Python程序设计

地理信息科学系　刘洪强

J6-557　电话：86081170

2021年3月10日星期三

课程安排：

36个学时，其中授课24个学时，实验12个学时

章节内容

第1章 认识Python

第2章 Python编程基础

第3章 函数、类、包和模块

第4章 文件操作

第5章 地图文档管理与数据访问

第6章 空间数据定位与查询

第7章 空间数据分析

第8章 地图制图与输出

第4章 文件操作

文件路径

文件基本操作

案例

os与os.path模块

目录操作

文件操作

- 为了长期保存数据以便重复使用、修改和共享，必须将数据以文件的形式存储到外部存储介质(如磁盘、U盘、光盘或云盘、网盘、快盘等)中。
- 文件操作在各类应用软件的开发中均占有重要的地位：
  - ✓ 管理信息系统是使用数据库来存储数据的，而数据库最终还是要以文件的形式存储到硬盘或其他存储介质上。
  - ✓ 应用程序的配置信息往往也是使用文件来存储的，图形、图像、音频、视频、可执行文件等等也都是以文件的形式存储在磁盘上的。

文件操作

- 按文件中数据的组织形式把文件分为文本文件和二进制文件两类。
  - ✓ **文本文件**：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。常规字符串是指记事本或其他文本编辑器能正常显示、编辑并且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。文本文件可以使用字处理软件如记事本进行编辑。
  - ✓ **二进制文件**：二进制文件把对象内容以字节串(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，通常也无法被人类直接阅读和理解，需要使用专门的软件进行解码后读取、显示、修改或执行。常见的如图形图像文件、音视频文件、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。

文件路径

- 指向一个本地存储的文件，是一个链接或者一个映射

```
path1 = 'C:/Users/Unique/Desktop/text.txt'  # 单个反斜杠：/
path2 = 'C:\\Users\\Unique\\Desktop\\text.txt'  # 两个斜杠：\\（第一个\是转义符）
path3 = r'C:\Users\Unique\Desktop\text.txt'  # r用于防止字符转义
# 路径书写格式
print(path1)
print(path2)
print(path3)
```

```
C:/Users/Unique/Desktop/text.txt
C:\Users\Unique\Desktop\text.txt
C:\Users\Unique\Desktop\text.txt
```

4.1 文件基本操作

- 文件内容操作三步走：打开、读写、关闭。

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

- ✓ file参数指定了被打开的文件名称。
- ✓ mode参数指定了打开文件后的处理方式。
- ✓ buffering参数指定了读写文件的缓存模式。0表示不缓存，1表示缓存，如大于1则表示缓冲区的大小。默认值-1表示由系统管理缓存。
- ✓ encoding参数指定对文本进行编码和解码的方式，只适用于文本模式，可以使用Python支持的任何格式，如GBK、utf8、CP936等等。

4.1 文件基本操作

- 如果执行正常，open()函数返回1个文件对象，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、磁盘空间不足或其他原因导致创建文件对象失败则抛出异常。

```
f1 = open('file1.txt', 'r')      # 以读模式打开文件
f2 = open('file2.txt', 'w')       # 以写模式打开文件
```

- 当对文件内容操作完以后，一定要关闭文件对象，这样才能保证所做的任何修改都确实被保存到文件中。

```
f1.close()
```

4.1 文件基本操作

```
# 读取文件：open语句
path2 = 'C:\\Users\\Unique\\Desktop\\text.txt'
f = open(path2)

print(type(f))
print(f)
print(f.read())
print('读取完毕')
# open('路径', '模式', enconding = '编码' )
# 模式：r：读取文件，默认；w：写入；rw：读取+写入；a：追加
# 简答的读取方法：.read() → 读取后，光标将会留在读取末尾

print(f.read()) # 运行第一次.read()之后，光标位于末尾，再次读取输出为空
print('读取为空')

f.seek(0) # 所以现在用 f.seek(0) 来移动光标
print(f.read())
print('第二次读取')

f.close() # 关闭文件链接  f.close()，养成一个好习惯
# print(f.read())    # 关闭后无法读取
```

```
<class '_io.TextIOWrapper'>
<_io.TextIOWrapper name='C:\\Users\\Unique\\Desktop\\text.txt' mode='r' encoding='cp936'>
package main;

import java.io.File;
import java.io.IOException;
import java.net.MalformedURLException;
读取完毕

读取为空
package main;

import java.io.File;
import java.io.IOException;
import java.net.MalformedURLException;
第二次读取
```

4.1 文件基本操作

- 但是，即使写了关闭文件的代码，也无法保证文件一定能够正常关闭。例如，如果在打开文件之后和关闭文件之前发生了错误导致程序崩溃，这时文件就无法正常关闭。在管理文件对象时推荐使用with关键字，可以有效地避免这个问题。

4.1 文件基本操作

- with语句的用法如下：

```
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
```

- 上下文管理语句with还支持下面的用法：

```
with open('test.txt', 'r') as src, open('test_new.txt', 'w') as dst:
    dst.write(src.read())
```

4.1 文件基本操作

- 文件打开方式

模式	说明
r	读模式（默认模式，可省略），如果文件不存在则抛出异常
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，如果文件已存在则抛出异常
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）

4.1 文件基本操作

- 文件对象常用属性

属性	说明
buffer	返回当前文件的缓冲区对象
closed	判断文件是否关闭，若文件已关闭则返回True
fileno	文件号，一般不需要太关心这个数字
mode	返回文件的打开模式
name	返回文件的名称

4.1 文件基本操作

- 文件对象常用方法

方法	功能说明
close()	把缓冲区的内容写入文件，同时关闭文件，并释放文件对象
read([size])	从文本文件中读取size个字符（Python 3.x）的内容作为结果返回，或从二进制文件中读取指定数量的字节并返回，如果省略size则表示读取所有内容
readable()	测试当前文件是否可读
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中，返回该列表，对于大文件会占用较多内存，不建议使用
seek(offset[, whence])	把文件指针移动到新的字节位置，offset表示相对于whence的位置。whence为0表示从文件头开始计算，1表示从当前位置开始计算，2表示从文件尾开始计算，默认为0
write(s)	把s的内容写入文件
writable()	测试当前文件是否可写
writelines(s)	把字符串列表写入文本文件，不添加换行符

4.2 案例

- **例1** 向文本文件中写入内容，然后再读出。

```
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('sample.txt', 'w') as fp:      #默认使用cp936编码
    fp.write(s)

with open('sample.txt') as fp:           #默认使用cp936编码
    print(fp.read())
```

```
Hello world
文本文件的读取方法
文本文件的写入方法
```

微软的CP936通常被视为等同GBK，连IANA 也以“CP936”为“GBK”之别名。不过实际上 GBK 定义的字符比 CP936 多出95个字，其中包括了15个非汉字及80个汉字。

4.2 案例

- **例2** 读取并显示文本文件的前5个字符。

```
# 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('sample.txt', 'r') as f:
    s = f.read(5)

print('s=',s)
print('字符串s的长度(字符个数)=', len(s))
```

```
s= Hello
字符串s的长度(字符个数)= 5
```

4.2 案例

- **例3** 读取并显示文本文件所有行。

```
# 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('sample.txt') as fp:          #假设文件采用CP936编码
    for line in fp:                     #文件对象可以直接迭代
        print(line)
        print('----------')
```

```
Hello world

----------
文本文件的读取方法

----------
文本文件的写入方法

----------
```

4.2 案例

- **例3** 读取并显示文本文件所有行。

```
# 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('sample.txt') as fp:          #假设文件采用CP936编码
    for line in fp:                     #文件对象可以直接迭代
        print(line)
        print('----------')
```

```
Hello world

----------
文本文件的读取方法

----------
文本文件的写入方法

----------
```

```
with open('sample.txt') as fp
    lines = fp.readlines()
    for line in lines:
        print(line)
        print('----------')
```

4.2 案例

- **例4** 移动文件指针，然后读取并显示文本文件中的内容。
  - ✓ seek()方法把文件指针定位到文件中指定字节的位置。读取时遇到无法解码的字符会抛出异常。

4.2 案例

```
>>> s = '中国山东烟台SDIBT'
>>> with open(r'D:\sample.txt', 'w') as fp:
        fp.write(s)

>>> fp = open(r'D:\sample.txt', 'r')
>>> print(fp.read(3))
中国山
>>> fp.seek(2)
2
>>> print(fp.read(1))
国
>>> fp.seek(13)
13
>>> print(fp.read(1))
D
>>> fp.seek(3)
3
>>> print(fp.read(1))
UnicodeDecodeError: 'gbk' codec can't decode byte 0xfa in position 0: illegal
multibyte sequence
```

4.2 案例

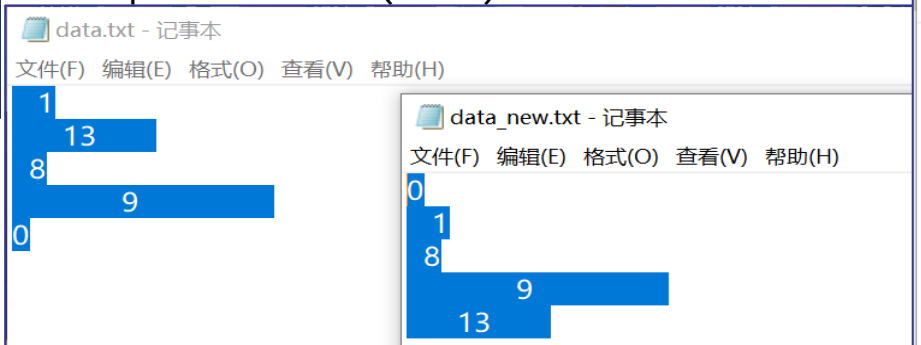
- **例5** 读取文本文件data.txt（文件中每行存放一个整数）中所有整数，按升序排序后再写入文本文件data_new.txt中。

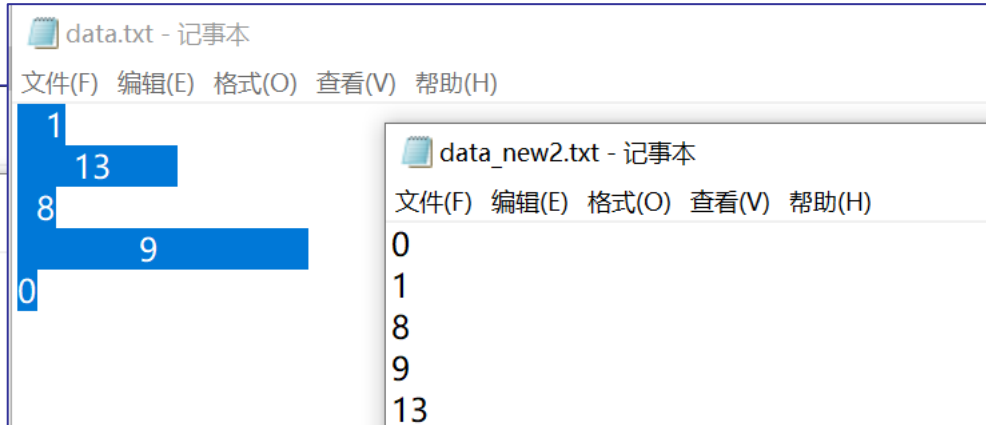
```
with open('data.txt') as fp:
    data = fp.readlines()

data.sort(key=int)

with open('data_new.txt', 'w') as fp:
    fp.writelines(data)
```

```
with open('data.txt') as fp:
    data = fp.readlines()
data = [int(line.strip()) for line in data]
data.sort()
data = [str(i)+'\n' for i in data]
with open('data_new2.txt', 'w') as fp:
    fp.writelines(data)
```



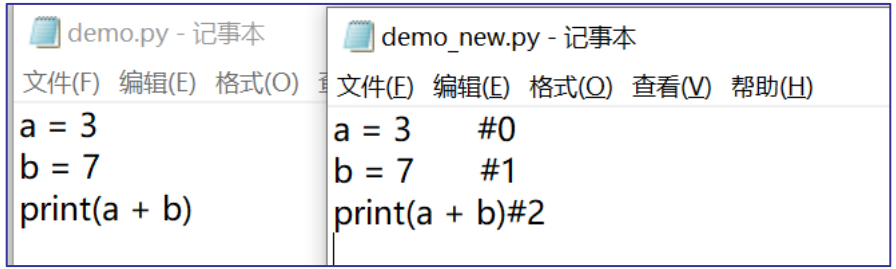


4.2 案例

- **例6** 编写程序，保存为demo.py，运行后生成文件demo_new.py，其中的内容与demo.py一致，但是在每行的行尾加上了行号。

```
filename = 'demo.py'
with open(filename, 'r') as fp:
    lines = fp.readlines()
maxLength = len(max(lines, key=len))
lines = [line.rstrip().ljust(maxLength)+'#'+str(index)+'\n'
         for index, line in enumerate(lines)]
with open(filename[:-3]+'_new.py', 'w') as fp:
    fp.writelines(lines)
```

enumerate() 函数用于将一个可遍历的数据对象(如列表、元组或字符串)组合为一个索引序列，同时列出数据下标和数据，一般用在 for 循环当中。



4.3 os与os.path模块

- os模块常用的文件操作函数

方法	功能说明
access(path, mode)	测试是否可以按照mode指定的权限访问文件
chdir(path)	把path设为当前工作目录
chmod(path, mode, *, dir_fd=None, follow_symlinks=True)	改变文件的访问权限
curdir	当前文件夹
environ	包含系统环境变量和值的字典
extsep	当前操作系统所使用的文件扩展名分隔符
get_exec_path()	返回可执行文件的搜索路径
getcwd()	返回当前工作目录
listdir(path)	返回path目录下的文件和目录列表
open(path, flags, mode=0o777, *, dir_fd=None)	按照mode指定的权限打开文件，默认权限为可读、可写、可执行
popen(cmd, mode='r', buffering=-1)	创建进程，启动外部程序

4.3 os与os.path模块

方法	功能说明
remove(path)	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
rename(src, dst)	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
replace(old, new)	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
scandir(path='.')	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
sep	当前操作系统所使用的路径分隔符
startfile(filepath [, operation])	使用关联的应用程序打开指定文件或启动指定应用程序
stat(path)	返回文件的所有属性
system()	启动外部程序
truncate(path, length)	将文件截断，只保留指定长度的内容
write(fd, data)	将bytes对象data写入文件fd

4.3 os与os.path模块

- os.path常用的文件操作函数

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

4.3 os与os.path模块

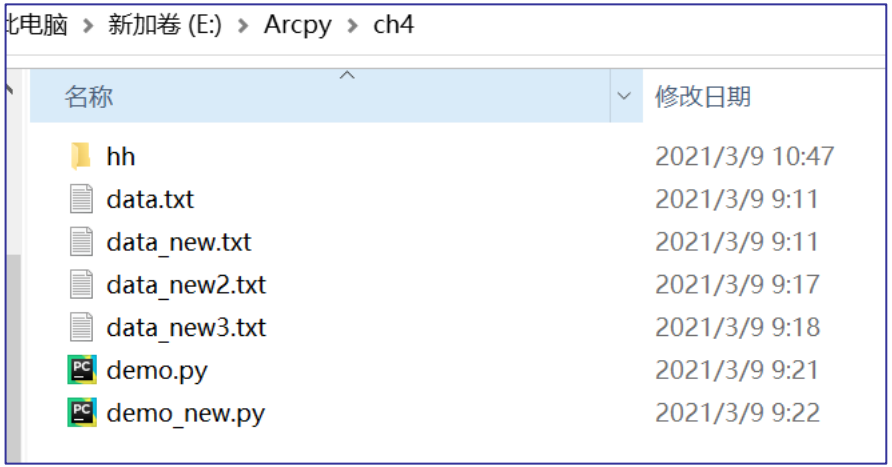
方法	功能说明
isabs(path)	判断path是否为绝对路径
isdir(path)	判断path是否为文件夹
isfile(path)	判断path是否为文件
join(path, *paths)	连接两个或多个path
realpath(path)	返回给定路径的绝对路径
relpath(path)	返回给定路径的相对路径，不能跨越磁盘驱动器或分区
samefile(f1, f2)	测试f1和f2这两个路径是否引用的同一个文件
split(path)	以路径中的最后一个斜线为分隔符把路径分隔成两部分，以元组形式返回
splitext(path)	从路径中分隔文件的扩展名
splitdrive(path)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

4.3 os与os.path模块

- 列出当前目录下所有扩展名为py的文件：

```
>>> import os
>>> [fname for fname in os.listdir(os.getcwd()) if
     os.path.isfile(fname) and fname.endswith('.py')]
```

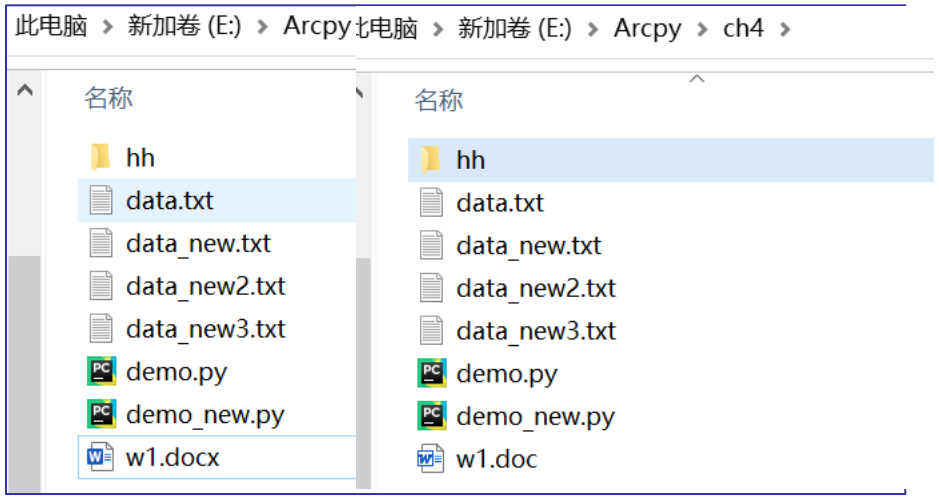
['demo.py', 'demo_new.py']



4.3 os与os.path模块

❖ 将当前目录中所有扩展名为docx的文件重命名为doc的文件：

```
import os
file_list = [filename for filename in os.listdir(".")\
             if filename.endswith(‘.docx')]
for filename in file_list:
    newname = filename[:-4]+’doc'
    os.rename(filename, newname)
    print(filename+"更名为："+newname)
```



4.4 目录操作

- os模块常用的目录操作函数

函数名称	使用说明
mkdir(path[, mode=0o777])	创建目录，要求上级目录必须存在
makedirs(path1/path2…, mode=511)	创建多级目录，会根据需要自动创建中间缺失的目录
rmdir(path)	删除目录，要求该文件夹中不能有文件或子文件夹
removedirs(path1/path2…)	删除多级目录
listdir(path)	返回指定目录下所有文件信息
getcwd()	返回当前工作目录
chdir(path)	把path设为当前工作目录
walk(top, topdown=True, onerror=None)	遍历目录树，该方法返回一个元组，包括3个元素：所有路径名、所有目录列表与文件列表

4.4 目录操作

```
>>> import os
>>> os.getcwd()                              #返回当前工作目录
'C:\\Python37'
>>> os.mkdir(os.getcwd()+'\\temp')           #创建目录
>>> os.chdir(os.getcwd()+'\\temp')           #改变当前工作目录
>>> os.getcwd()
'C:\\Python37\\temp'
>>> os.mkdir(os.getcwd()+'\\test')
>>> os.listdir('.')
['test']
>>> os.rmdir('test')                         #删除目录
>>> os.listdir('.')
[]
```

The End