



第7章 文件操作

刘 卉

huiliu@fudan.edu.cn

主要内容



文本文件

存储常规字符串

- 由若干文本行组成, 通常每行以换行符'\n'结束.
- 常规字符串: 记事本或其它文本编辑器能正常显示和编辑, 人类能够直接阅读和理解, 如: 英文字母、汉字、数字字符串,

文本文件的编辑

- 字处理软件: gedit, 记事本,

二进制文件

存储数据的二进制表示

- 数据库文件, 图像文件, 可执行文件, 音视频文件, Office文档,

二进制文件的编辑

- 不能使用记事本或其它文本编辑软件进行读写, 人类无法直接阅读和理解.
- 常用的二进制文件工具: ultraedit, ultrahex; notepad++,

7.1 文件对象

python的内置对象之一

- 是一种资源;
- 使用时, 打开文件对象;
- 使用完毕, 释放文件对象.

文件对象的操作遵循特定流程

- 1) 打开: 使用函数open()
- 2) 操作: 读/写/删除等操作
- 3) 关闭: 使用函数close()

文件打开

文件对象名 = open(file, mode='r', buffering=-1, encoding=None, ...)

e.g. f = open('c.dat', 'rb')

- file: 文件名

- 被打开的文件名称(包括路径)

- mode: 访问模式

- 打开文件后的处理方式(右表).
 - 红色标记的方式可与其它方式组合使用.
 - 默认值是'r'.

值	说明
r	读模式
w	写模式
a	追加模式
b	二进制模式
+	读写模式

文件对象名 = open(file, mode='r', buffering=-1, encoding=None, ...)

□ buffering: 缓冲区

- 指定读写文件的缓存模式.
- 0: 不缓存, 1: 缓存; 大于1: 指定缓冲区大小; 小于0: 使用默认缓冲区大小.
- 默认值: 缓存模式.

□ encoding: 文本文件的编码

```
>>>help(open)
```

- 仅用于文本文件模式, 默认编码方式取决于系统.
- 为了避免跨系统打开包含中文字符的文本文件时出现乱码, 需明确指定编码方式: `f = open(fname, 'r', encoding = 'UTF-8')`

文件对象名 = open(file, mode='r', buffering=-1, encoding=None, ...)

❑ 获取本机系统的默认编码

```
>>> locale.getpreferredencoding(False)
'cp936'
```

❑ Python源程序文件的编码方式

- 默认是ASCII编码
- 当py文件中出现中文字符时可能会出现问题.
- 解决方法: 在首行添加魔法注释/声明

```
# coding = utf-8      或
# coding: utf-8       或
# -*- coding: utf-8 -*-
```


文件对象名 = open(file, mode='r', buffering=-1, encoding=None, ...)

□ open()函数返回一个文件对象

- 利用该对象可进行各种文件操作.
- 文件对象的属性可访问.

文件打开的典型调用方法

□ 以读模式打开文件

- 要求文件已经存在, 否则产生异常.

1. 文件在当前目录下 `f = open('grades.txt')`

2. 用绝对路径方式

```
f = open(r' C:\Users\comet\Documents\Python\Examples\Chap7_File\grades.txt')
```

3. 用相对路径方式 `f = open('../grades.txt')`

../表示py程序所在目录的上一级目录

□ 以写模式打开文件

```
f = open('a.txt', 'w')
```

- 文件存在则清空文件内容, 不存在则创建文件.

□ 以读写模式打开文件

```
f = open('b.csv', 'a+')
```

- 文件不存在则创建文件, 否则置文件指针到文件尾.

文件对象的属性

属性	说明
f.closed	若文件关闭则为真，否则为假
f.encoding	文件所使用的编码
f.mode	文件的访问模式
f.name	文件的名称

```
# -*- coding: utf-8 -*-  
# 输出文本文件的属性  
# filename property.py
```

```
def printfp(filename):  
    try:  
        fh = open(filename)  
        print('file name : %s' % fh.name)  
        print('access mode: %s' % fh.mode)  
        print('encoding   : %s' % fh.encoding)  
        print('closed     : %s' % fh.closed)  
    finally:  
        fh.close()  
# end printfp  
  
if __name__ == '__main__':  
    printfp('sample.txt')  
# end if
```

```
file name : sample.txt  
access mode: r  
encoding   : cp936  
closed     : False
```



property.py

文件对象的常用操作

分组	方法	说明
读	<code>f.read([size])</code>	从文件中读取 <code>size</code> 个字节，默认读取所有内容, 返回字符串
	<code>f.readline()</code>	从文本文件中读取一行内容, 返回字符串
	<code>f.readlines()</code>	把文本文件中的每行作为字符串插入列表中返回该列表
写	<code>f.write(s)</code>	把字符串 <code>s</code> 的内容写入文件
	<code>f.writeline (s)</code>	把字符串列表 <code>s</code> 写入文本文件，不添加换行符
其他操作	<code>f.flush()</code>	把缓冲区的内容写入文件，不关闭文件
	<code>f.close()</code>	把缓冲区的内容写入文件，关闭文件，释放文件对象
	<code>f.tell()</code>	返回当前文件指针的位置
	<code>f.truncate([size])</code>	删除从当前指针位置到文件末尾的内容。如果指定了 <code>size</code> ，则不论指针在什么位置都只留下前 <code>size</code> 个字节，其余的删除
	<code>f.seek(offset[,whence])</code>	把文件指针移动到新的位置。 <code>offset</code> 表示相对于 <code>whence</code> 的位置； <code>whence</code> 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



文件操作模式

□ 模式一

```
handler = open(filename)
lines = handler.readlines()
handler.close()
print(lines)
```

□ 优缺点

- 常规操作方式: 三步;
- 如有例外产生, 程序运行不正常; 同时资源不能释放.

□ 模式二

```
with open(filename) as handler:
    lines = handler.readlines()
print(lines)
```

□ 优缺点

- 比较安全的操作方式;
- 自动进行资源管理.

□ 模式三

try:

```
handler = open(filename)
lines = handler.readlines()
```

finally:

```
handler.close()
```

print(lines)

□ 优缺点

- 较好的操作方式;
- 保证资源的释放.

7.2 文本文件操作案例精选

[例] 向文本文件写入内容.

- 仿照模式一和模式二写代码.
- 注意: s1和s2都有行结束标记.



writeFile.py

```
f = open('sample.txt', 'w')
s1 = '1\tfudan\t复旦大学\t中国上海\t200433\n'
s2 = '2\ttsjtu\t交通大学\t中国上海\t200240\n'
f.write(s1)
f.write(s2)
f.close()
```

```
s1 = '1\tfudan\t复旦大学\t中国上海\t200433\n'
s2 = '2\ttsjtu\t交通大学\t中国上海\t200240\n'
with open('sample.txt', 'w') as f:
    f.write(s1)
    f.write(s2)
```

sample.txt - 记事本				
文件(E) 编辑(E) 格式(O) 查看(V) 帮助(H)				
1	fudan	复旦大学	中国上海	200433
2	tsjtu	交通大学	中国上海	200240

[例] 读取并显示文本文件所有行.

```
def fa(fname):  
    f = open(fname)  
    while True:  
        line = f.readline()  
        if line == '':  
            break  
        print(line.strip())  
    f.close()  
fa('sample.txt')
```

```
def fb(fname):  
    with open(fname) as f:  
        lines = f.readlines()  
        for line in lines:  
            print(line, end='')  
fb('sample.txt')
```

```
def fc(fname):  
    with open(fname) as f:  
        print(f.read())
```

```
if __name__ == '__main__':  
    fc('sample.txt')
```

⚠ 读出的每一行(line)中包含一个换行符; 不做处理的话, 将输出一个空行.



readFile1-3.py

□ 列表推导式支持文件对象迭代

```
def fc(fname):  
    with open(fname) as f:  
        print([line for line in f])    # print(f.readlines())  
  
if __name__ == '__main__':  
    fc('sample.txt')
```

```
def fb(fname):  
    with open(fname) as f:  
        lines = f.readlines()  
        for line in f:  
            print(line, end='')  
fb('sample.txt')
```



[例] 对文件grades.txt中每行整数分别按升序排列, 并写入grades_n.txt.

```
with open('grades.txt', 'r') as f:
    data = f.readlines()

result = []
for line in data:
    nums = line.strip().split()
    nums = sorted(map(int, nums))
    nums = ' '.join(map(str, nums)) + '\n'
    result.append(nums)

with open('grades_n.txt', 'w') as f:
    f.writelines(result)
```



sort_write.py

[例] 把property.py中的代码, 在行首加上行号, 保存为文件

property_n.py

```
def comment(fname):  
    with open(fname, 'r', encoding='UTF-8') as f:  
        lines = [str(i)+") "+line for i, line in enumerate(f, 1)]  
  
    with open(fname[: -3]+'_n.py', 'w', encoding='UTF-8') as f:  
        f.writelines(lines)  
  
if __name__ == '__main__':  
    comment('property.py')
```

从1开始编号,
默认从0开始。

- 思考: 若要求文件中的注释行不算在内, 程序该如何编写?



7.3 二进制文件操作案例精选

Python中常用的序列化模块

- 包括struct、pickle、json、marshal和shelve, 其中pickle有C语言实现的cPickle, 速度约提高1000倍, 应优先考虑使用.



二进制文件: 使用pickle写入数据

[例] 使用pickle模块写入二进制数据.

```
import pickle
def write_to_pickle():
    f = open('T68.dat', 'wb')
    n = 7      #记录写入文件的数据项数
    i = 13000000
    a = 99.056
    s = '中国上海 200433'
    lst = [[1, 2, 3], [4, 5, 6], [7, 8, 9]]
    tu = (-5, 10, 8)
    coll = {4, 5, 6}
    dic = {'a': 'apple', 'b': 'banana', 'g':
'grape', 'o': 'orange'}

    try:
        pickle.dump(n, f)
        pickle.dump(i, f)
        pickle.dump(a, f)
        pickle.dump(s, f)
        pickle.dump(lst, f)
        pickle.dump(tu, f)
        pickle.dump(coll, f)
        pickle.dump(dic, f)
    except:
        print('写入文件异常')
    finally:
        f.close()

if __name__ == '__main__':
    write_to_pickle()
```



Chap6_8.py

Copy&Run

二进制文件: 使用pickle读取数据

[例] 读取二进制文件中的数据.

```
import pickle
def read_from_pickle():
    f = open('T68.dat', 'rb')
    # 读出写入文件的数据项数
    n = pickle.load(f)
    i = 0
    while i < n:
        x = pickle.load(f)
        print(x)
        i = i + 1
    f.close()

if __name__ == '__main__':
    read_from_pickle()
```

文件T68.dat 内容

Address	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f	Dump
00000000	80	03	4b	07	2e	80	03	4a	40	5d	c6	00	2e	80	03	47	€.K..€.J@]?.€.G□
00000010	40	58	c3	95	81	06	24	dd	2e	80	03	58	12	00	00	00	@X腰..\$?€.X....□□
00000020	e4	b8	ad	e5	9b	bd	e4	ba	ba	e6	b0	91	20	31	32	33	涓 球浜烘奸 123□
00000030	34	35	71	00	2e	80	03	5d	71	00	28	5d	71	01	28	4b	45q..€.]q.(]q.(K
00000040	01	4b	02	4b	03	65	5d	71	02	28	4b	04	4b	05	4b	06	.K.K.e]q.(K.K.K.
00000050	65	5d	71	03	28	4b	07	4b	08	4b	09	65	65	2e	80	03	e]q.(K.K.K.ee.€.
00000060	4a	fb	ff	ff	ff	4b	0a	4b	08	87	71	00	2e	80	03	63	J? K.K.喻..€.c
00000070	62	75	69	6c	74	69	6e	73	0a	73	65	74	0a	71	00	5d	builtins.set.q.]
00000080	71	01	28	4b	04	4b	05	4b	06	65	85	71	02	52	71	03	q.(K.K.K.e豺.Rq.[
00000090	2e	80	03	7d	71	00	28	58	01	00	00	00	67	71	01	58	.€.}q.(X....gq.X
000000a0	05	00	00	00	67	72	61	70	65	71	02	58	01	00	00	00grapeq.X....
000000b0	61	71	03	58	05	00	00	00	61	70	70	6c	65	71	04	58	aq.X....appleq.X
000000c0	01	00	00	00	6f	71	05	58	06	00	00	00	6f	72	61	6eoq.X....oran
000000d0	67	65	71	06	58	01	00	00	00	62	71	07	58	06	00	00	geq.X....bq.X...
000000e0	00	62	61	6e	61	6e	61	71	08	75	2e						.bananaq.u.

7.4 文件级操作

读写文件内容: 文件对象

处理文件路径: `os.path`

命令行读取文件内容: `fileinput`

创建临时文件和文件夹: `tempfile`

表示和处理文件系统路径: `pathlib`(Python 3.4以上)



os模块: 常用方法

方法	功能说明
<code>access(path, mode)</code>	按照 mode 指定的权限访问文件
<code>open(path, flags, mode=0o777, *, dir_fd=None)</code>	按照 mode 指定的权限打开文件, 默认权限为可读、可写、可执行
<code>chmod(path, mode, *, dir_fd=None, follow_symlinks=True)</code>	改变文件的访问权限
<code>remove(path)</code>	删除指定的文件
<code>rename(src, dst)</code>	重命名文件或目录
<code>stat(path)</code>	返回文件的所有属性
<code>fstat(path)</code>	返回打开的文件的所有属性
<code>listdir(path)</code>	返回 path 目录下的文件和目录列表
<code>startfile(filepath [, operation])</code>	使用关联的应用程序打开指定文



os.path模块: 常用方法

方法	使用说明
<code>abspath(path)</code>	返回绝对路径
<code>dirname(p)</code>	返回目录的路径
<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
<code>isabs(path)</code> 、 <code>isdir(path)</code> 、 <code>isfile(path)</code>	判断 <code>path</code> 是否为绝对路径、目录、文件
<code>split(path)</code>	对路径进行分割，以列表形式返回
<code>splittext(path)</code>	从路径中分割文件的扩展名
<code>splitdrive(path)</code>	从路径中分割驱动器的名称
<code>walk(top,func,arg)</code>	遍历目录

示例1

- 列出当前目录下, 所有扩展名为.py的文件.

```
import os
def list_file_ext(ext):
    file_list = [fn for fn in os.listdir(os.getcwd()) \
                  if os.path.isfile(fn) and fn.endswith(ext)]
    print(file_list)

if __name__ == '__main__':
    list_file_ext('.py')
```



listPyFile.py

示例2

□ 将当前目录下所有扩展名为.txt的文件更改为.tx

```
import os
def rename_extention(old, new):
    file_list = os.listdir()
    for filename in file_list:
        pos = filename.rindex('.')
        if filename[pos+1:] == old :
            newname=filename[: pos+1]+new
            os.rename(filename, newname)
            print(filename+"更改为: "+newname)

if __name__ == '__main__':
    rename_extention('txt', 'tx')
```



rename_extention.py

```
import os
def rename_extention2(old, new):
    file_list = [filename for filename in os.listdir() if filename.endswith(old)]
    for filename in file_list:
        newname = filename[: -len(old)]+new
        os.rename(filename, newname)
        print(filename+"更改为: "+newname)

if __name__ == '__main__':
    rename_extention2('tx', 'txt')
```

shutil模块

提供(多个)文件的高级(High Level)操作, 包括:

⑩ (多个)文件操作:

- copyfile 文件拷贝
- copystat 文件属性拷贝
- copytree , rmtree, ... 目录拷贝/删除

⑩ 压缩/解压缩操作

- make_archive(base_name, format, root_dir)
- Unpack_archive(filename, extract_dir, format)
- 支持的格式包括: zip, tar, gztar等

⑩ 可通过dir命令和help命令查看

```
>>> import shutil
>>> dir(shutil)
['Error', 'ExecError', 'ReadError', 'RegistryError', 'SameFileError', 'SpecialFileError', '_ARCHIVE_FORMATS', '_BZ2_SUPPORTED', '_LZMA_SUPPORTED', '_UNPACK_FORMATS', '__all__', '__builtins__', '__cached__', '__doc__', '__file__', '__loader__', '__name__', '__package__', '__spec__', '_basename', '_check_unpack_options', '_copyxattr', '_destinsrc', '_ensure_directory', '_find_unpack_format', '_get_gid', '_get_uid', '_make_tarball', '_make_zipfile', '_ntuple_diskusage', '_rmtree_safe_fd', '_rmtree_unsafe', '_samefile', '_unpack_tarfile', '_unpack_zipfile', '_use_fd_functions', 'chown', 'collections', 'copy', 'copy2', 'copyfile', 'copyfileobj', 'copymode', 'copystat', 'copytree', 'disk_usage', 'errno', 'fnmatch', 'get_archive_formats', 'get_terminal_size', 'get_unpack_formats', 'getgrnam', 'getpwnam', 'ignore_patterns', 'make_archive', 'move', 'nt', 'os', 'register_archive_format', 'register_unpack_format', 'rmtree', 'stat', 'sys', 'tarfile', 'unpack_archive', 'unregister_archive_format', 'unregister_unpack_format', 'which']
>>> help(shutil.rmtree)
```

Help on function rmtree in module shutil:

```
rmtree(path, ignore_errors=False, onerror=None)
    Recursively delete a directory tree.
```

.... #详细解释



shutil模块: 示例

- 压缩：将D:\Python\Examples\Chap6文件夹及该文件夹中所有文件压缩至D:\Python\Examples\Chap6.zip文件

```
>>> shutil.make_archive(r'D:\Python\Examples\Chap6', 'zip', r'D:\Python\Examples', 'Chap6')
'D:\Python\Examples\Chap6.zip'
```

- 解压缩：将刚压缩得到的文件Chap6.zip解压缩至D:\Chap6_unpack文件夹

```
>>> shutil.unpack_archive('D:\Python\Examples\Chap6.zip', 'D:\Chap6_unpack')
```

- 批量删除:

```
>>> shutil.rmtree('D:\Chap6_unpack')
```


7.5 目录操作

- `os`和`os.path`模块中包含目录操作方法
 - 可通过`dir(os)`和`dir(os.path)`查看
- 常用方法见下表

方法	使用说明
<code>chdir(path)</code>	把 <code>path</code> 设为当前工作目录
<code>makedirs(path[,mode=0777])</code>	创建目录
<code>makedirs(path1/path2...,mode=511)</code>	创建多级目录
<code>rmdir(path)</code>	删除目录
<code>removedirs(path1/path2...)</code>	删除多级目录
<code>listdir(path)</code>	返回指定目录下所有文件信息
<code>getcwd()</code>	返回当前工作目录
<code>walk(top,topdown=True,onerror=None)</code>	遍历目录树
<code>sep</code>	当前操作系统使用的路径分割符

□ 遍历目录: 使用os.walk方法

```
def visitDir(path):  
    if not os.path.isdir(path):  
        print('Error: "', path, '" is not a directory or does not exist.')  
        return  
    list_dirs = os.walk(path)  
    #os.walk返回元组(所有路径名,所有目录列表,所有文件列表)  
    for root, dirs, files in list_dirs:  
        for d in dirs: #遍历该元组的目录  
            print(os.path.join(root, d)) #获取完整路径  
        for f in files: #遍历该元组的文件  
            print(os.path.join(root, f)) #获取文件绝对路径  
  
if __name__ == '__main__':  
    visitDir2('c:\\')
```