随笔 - 62  文章 - 16  评论 - 222

[**python学习笔记——文件和目录操作**](https://www.cnblogs.com/zhoujie/archive/2013/04/10/python7.html)

**目录和文件操作**

　　语言只有和外部连起来操作才会实现更强大的功能，比如操作文件、数据库等，这样数据可以有一块单独存储的地方，而不是存放在内存中。更强大的是网络编程，当然这些后续都会学习。接下来学习python对目录和文件的操作。前面的笔记都是基础理论知识，我觉得从这里开始几乎就可以干一些事了。

　　有关文件夹与文件的查找，删除等功能 在 **os** 模块中实现。使用时需先导入这个模块：import  os

**目录**

1、取得当前目录——os.getcwd()

>>> import os

>>> s=os.getcwd()#获得当前运行脚本所在目录

>>> s

'C:\\Python27'

比如运行test.py，那么输入该命令就会返回脚本所在的文件夹位置。

例如将test.py放入A文件夹。并且希望不管将A文件夹放在硬盘的哪个位置，都可以在A文件夹内生成一个新文件夹。且文件夹的名字根据时间自动生成。

>>> import os

>>> import time

>>> folder = time.strftime(r"%Y-%m-%d\_%H-%M-%S",time.localtime())

>>> **os.makedirs**(r'%s/%s'%(os.getcwd(),folder)) #创建以时间命名文件夹名

这是运行脚本的目录即'C:\\Python27'下多了一个以当前时间命名的文件夹

2、创建子目录——os.makedirs("path")，path是要创建的子目录创建多级目录

>>> os.makedirs("C:\\temp\\test") #这是C盘下就创建了temp目录，temp下嵌套的文件夹是test

(当然，也可能创建失败，比如path已存在，或者驱动器不在，或者无写权限等等)

3、更改当前目录——os.chdir()

相当于dos或Linux下的cd命令

>>> os.chdir('c:\\') #将当前目录改为C盘根目录下

4、将路径分解为目录名和文件名——os.path.split()

格式为：fpath , fname = os.path.split( "要分解的路径")

>>> a,b=os.path.split("c:\\dir1\\dir2\\file.txt")

>>> print a

c:\dir1\dir2

>>> print b

file.txt

5、分解文件名的扩展名——os.path.splitext()

格式为：fpath\_name , ftext = os.path.splitext( "要分解的路径")

>>> a,b=os.path.splitext("c:\\dir1\\dir2\\file.txt")

>>> print a

c:\dir1\dir2\file

>>> print b

.txt

6、判断一个路径（目录或文件）是否存在——os.path.exists()

格式为：os.path.exists(“要判断的路径或文件”)

[复制代码](javascript:void(0);)

>>> os.path .exists ("C:\\") #该路径存在

True

>>> os.path .exists ("C:\\123\\") #该路径不存在

False

>>> os.path .exists ("C:\\123.txt") #该文件不存在

False

>>> os.path .exists ("C:\\test.txt") #该文件存在

True

[复制代码](javascript:void(0);)

7、判断一个路径是否有需要的文件——os.path.isfile("文件")

>>> os.path .isfile("C:\\test.txt")

True

>>> os.path .isfile("C:\\123.txt")

False

8、判断一个路径是否存在——os.path.isdir("路径")

>>> os.path .isdir("C:\\")

True

>>> os.path .isdir("H:\\")

False

9、获取目录中的文件及子目录的列表——os.listdir("路径")

相当于Windows下powershell中获取Get-ChildItem命令和Linux中的ls命令。但是这个显示不是以常见的列表的形式：

>>> os.listdir("C:\\") #这里包括隐藏文件也显示出来了

['$Recycle.Bin', '360ld', '360rescue', '360SANDBOX', '360SysRt', 'Boot', 'bootmgr', 'BOOTSECT.BAK', 'CacheTemp', 'Documents and Settings', 'grldr', 'IFRToolLog.txt', 'inetpub', 'MSOCache', 'pagefile.sys', 'Program Files', 'Program Files (x86)', 'ProgramData', 'Python27', 'Recovery', 'RECYCLER', 'SBTDR', 'System Volume Information', 'test.txt', 'Users', 'Windows']

示例：获取指定目录下的所有子目录的列表

[+ View Code](https://www.cnblogs.com/zhoujie/archive/2013/04/10/python7.html)

　　获取指定目录下所有文件的列表

[+ View Code](https://www.cnblogs.com/zhoujie/archive/2013/04/10/python7.html)

10、删除子目录——os.rmdir("path"),只能删除空目录

>>> os.rmdir("C:\\temp\\test") #注意只删除了test目录

>>> os.rmdir("C:\\temp") #这里才删除了temp目录

**文件**

　　python中模块的引入使得对文件的操作变的很简单。最基本的文件操作就是在文件中进行读写数据，在操作文件之前要打开文件。

打开文件——open('file'[,'mode'])

>>>import os  
>>> os.getcwd()

'c:\\'

>>> file=open('test.txt') #默认的mode是'r'，即读模式

>>> file.read() #读取文件内容

'hello\nworld\nhello,python' # \n在文件中的形式是换行

mode的选项即含义如下：

|  |  |
| --- | --- |
| **模 式** | **描述** |
| r | 以读方式打开文件，可读取文件信息。 |
| w | 以写方式打开文件，可向文件写入信息。如文件存在，则清空该文件，再写入新内容；如果文件不存在则创建 |
| a | 以追加模式打开文件（即一打开文件，文件指针自动移到文件末尾），如果文件不存在则创建 |
| r+ | 以读写方式打开文件，可对文件进行读和写操作。 |
| w+ | 消除文件内容，然后以读写方式打开文件。 |
| a+ | 以读写方式打开文件，并把文件指针移到文件尾。 |
| b | 以二进制模式打开文件，而不是以文本模式。该模式只对Windows或Dos有效，类Unix的文件是用二进制模式进行操作的。 |

关于文件的其他操作我觉得没必要记录的很详细了，因为基本都很简单，下面列出的是文件的常用方法，并且在例子中有相关说明。另外有一点注意的是读取文件中经常存在的编码问题。不同的解释器默认的编码不同，具体解决方案会再介绍。

常见文件操作方法：

|  |  |
| --- | --- |
| **方法** | **描述** |
| f.close() | 关闭文件，记住用open()打开文件后一定要记得关闭它，否则会占用系统的可打开文件句柄数。 |
| f.name() | 获取文件名称 |
| f.next() | 返回下一行，并将文件操作标记位移到下一行。把一个file用于for … in file这样的语句时，就是调用next()函数来实现遍历的。 |
| f.fileno() | 获得文件描述符，是一个数字。返回一个长整型的”文件标签“ |
| f.flush() | 刷新输出缓存，把缓冲区的内容写入硬盘 |
| f.isatty() | 如果文件是一个终端设备文件（Linux系统中），则返回True，否则返回False。 |
| f.read([size]) | 读出文件，size为读取的长度，以byte为单位 |
| f.readline([size]) | 读出一行信息，若定义了size，则读出 一行的一部分 |
| f.readlines([size]) | 读出所有行，也就是读出整个文件的信息。(把文件每一行作为一个list的一个成员，并返回这个list。其实它的内部是通过循环调用readline()来实现的。如果提供size参数，size是表示读取内容的总长，也就是说可能只读到文件的一部分) |
| f.seek(offset[,where]) | 把文件指针移动到相对于where的offset位置。where为0表示文件开始处，这是默认值 ；1表示当前位置；2表示文件结尾。(注意：如果文件以a或a+的模式打开，每次进行写操作时，文件操作标记会自动返回到文件末尾) |
| f.tell() | 获得文件指针位置，标记当前位置，以文件开头为原点 |
| f.truncate([size]) | 把文件裁成规定的大小，默认的是裁到当前文件操作标记的位置。如果size比文件的大小还要大，依据系统的不同可能是不改变文件，也可能是用0把文件补到相应的大小，也可能是以一些随机的内容加上去。 |
| f.write(string) | 把string字符串写入文件，write()不会在str后加上一个换行符。 |
| f.writelines(list) | 把list中的字符串一行一行地写入文件，是连续写入文件，没有换行。 |

现有一个test.txt文件，格式如下：

|  |
| --- |
| hello  world  hello  python |

下面是一些常见操作：

[复制代码](javascript:void(0);)

>>> file=open('test.txt')

>>> file.read (4) #读取前4个字节

'hell'

>>> file.read(6) #注意这里是在刚才读过的基础上再向后读的

'o\nworl'

>>> file.read () #不指定size，则读到文件结尾

'd\nhello\npython'

>>> file.read() #再读时已是文件结尾

''

>>> file.seek(0) #将文件位置定位到第一个字节

>>> file.readline () #一次读一行

'hello\n'

>>> file.readline ()

'world\n'

>>> file.seek(0) #将文件定位到开始

>>> file.readlines () #读取整个文件的内容

['hello\n', 'world\n', 'hello\n', 'python']  
>>> file .tell() #读完之后显示seek位置，即文件的最后了  
27L #以长整型表示  
>>> file.name #查看文件的名称  
'test.txt'  
>>> file.close() #关闭文件

[复制代码](javascript:void(0);)

　　刚开始测试read和readline等用法的时候，因为只要读取一次就在上次基础上往后读，当时我还以为这是一个出栈操作，显然，知道有seek这个方法后，我才知道它不是，只是每次读的时候seek的位置就往后移动一个，而每次读取是以seek所在的位置为起点的。所以如果需要从头开始读取文件内容时，将文件位置设为开始即可，即seek(0)。

>>> file=open('test.txt','w')

>>> file.write ('\nwelcome') #会将之前的内容覆盖

>>> file.writelines ('I love python')

>>> file.close () #关闭文件时才能看到文件内容的修改

>>> file=open('test.txt','a') #追加到文件尾，而不会覆盖

>>> file.writelines ('this is a test')

>>> file.close()

对文件的相关操作有时需要引入shutil模块：

[复制代码](javascript:void(0);)

>>> import shutil

>>> shutil.copyfile('test.txt','123.txt') #参数只能是文件

>>> shutil.copy("olddir","newfileordir") #olddir只能是文件夹，newfile可以是文件，也可以是目标目录

>>> shutil.copytree("olddir","newdir") #olddir和newdir都只能是目录，且newdir必须不存在

>>> shutil.move("oldpos","newpos") #移动文件或目录

>>> shutil.rmtree("dir") #空目录、有内容的目录都可以删

>>> import os  
>>> os.rmdir("dir") #只能删除空目录

>>> os.remove("file") #删除文件

>>> os.rename("oldname","newname") #文件或目录都是使用这条命令

[复制代码](javascript:void(0);)

关于文件和目录操作的总结，推荐博客：<http://www.cnblogs.com/rollenholt/archive/2012/04/23/2466179.html>

文件编码：

[复制代码](javascript:void(0);)

#获得当前环境默认编码

>>> import sys

>>> import locale

>>> sys.getdefaultencoding() # 返回当前系统所使用的默认字符编码

'ascii'

>>> sys.getfilesystemencoding () # 返回用于转换Unicode文件名至系统文件名所使用的编码

'mbcs'

>>> locale.getdefaultlocale() # 获取默认的区域设置并返回元组(语言, 编码)

('zh\_CN', 'cp936')

>>> locale .getpreferredencoding () # 返回用户设定的文本数据编码

'cp936'

11、glob

使用glob.glob获得文件路径

import glob

imgsPath='test'

images=glob.glob(imgsPath+'/\*') #获得imgsPath下所有的文件路径

print(images)

12、os.path.isdir()和os.path.isfile()

os.path.isdir()用于判断某一对象是否为目录(需提供绝对路径)

os.path.isfile()用于判断某一对象是否为文件(需提供绝对路径)

## 13. 判断给出的路径是否为绝对路径

import os

os.path.isabs(r'd:\home')

## 14. 将路径分离为文件名和目录名

import os

path, file = os.path.split(r'd:\home\a.py')

## 15. 获取目录（文件）属性以及文件大小

import os

os.stat(path) # 获取文件属性

os.path.getsize(path) # 获取文件大小

## 16与系统有关

import os

os.name # 返回系统名称，对于Windows，它是'nt'，而对于Linux/Unix用户，它是'posix'

os.linesep # 给出当前平台使用的行终止符

os.path.sep # 路径分隔符，可以跨平台使用

## 17. 文件的创建

import os

os.mknod("a.txt") # 创建空文件,文件名称为a.txt

## 18. 文件的读取

（1）file=open("c:\\a.txt")

print(file.read())

file.close() #用这种方式打开文件，必须要手动关闭

（2）import os

with open(file\_name, 'rb') as file: #用with + as 的模式避免未关闭文件而造成的内存浪费以及用简短的名称来代替打开的文件,'b'为只读二进制形式打开

a = file.read(20) # 打开的文件作为数据流，有很多种操作方法。#用这种方式打开文件不用手动关闭

文件的打开模式

模式名称 模式详解

w 以写方式打开

a 以追加模式打开 (从 EOF 开始, 必要时创建新文件)

r+ 以读写模式打开

w+ 以读写模式打开 (参见 w )

a+ 以读写模式打开 (参见 a )

rb 以二进制读模式打开

wb 以二进制写模式打开 (参见 w )

ab 以二进制追加模式打开 (参见 a )

rb+ 以二进制读写模式打开 (参见 r+ )

wb+ 以二进制读写模式打开 (参见 w+ )

ab+ 以二进制读写模式打开 (参见 a+ )

文件操作方法名称 方法详解

file.read([size]) size为读取的长度，以byte为单位

file.readline([size]) 读一行，如果定义了size，有可能返回的只是一行的一部分

file.readlines([size]) 把文件每一行作为一个list的一个成员，并返回这个list。其实它的内部是通过循环调用readline()来实现的。如果提供size参数，size是表示读取内容的总长，也就是说可能只读到文件的一部分。

file.write(str) 把str写到文件中，write()并不会在str后加上一个换行符

file.writelines(seq) 把seq的内容全部写到文件中(多行一次性写入)。这个函数也只是忠实地写入，不会在每行后面加上任何东西。

file.close() 关闭文件。python会在一个文件不用后自动关闭文件，不过这一功能没有保证，最好还是养成自己关闭的习惯。 如果一个文件在关闭后还对其进行操作会产生ValueError

file.flush() 把缓冲区的内容写入硬盘

file.fileno() 返回一个长整型的”文件标签“

file.isatty() 文件是否是一个终端设备文件（unix系统中的）

file.tell() 返回文件操作标记的当前位置，以文件的开头为原点

file.next() 返回下一行，并将文件操作标记位移到下一行。把一个file用于for … in file这样的语句时，就是调用next()函数来实现遍历的。

file.seek(offset[,whence]) 将文件打操作标记移到offset的位置。这个offset一般是相对于文件的开头来计算的，一般为正数。但如果提供了whence参数就不一定了，whence可以为0表示从头开始计算，1表示以当前位置为原点计算。2表示以文件末尾为原点进行计算。需要注意，如果文件以a或a+的模式打开，每次进行写操作时，文件操作标记会自动返回到文件末尾。

file.truncate([size]) 把文件裁成规定的大小，默认的是裁到当前文件操作标记的位置。如果size比文件的大小还要大，依据系统的不同可能是不改变文件，也可能是用0把文件补到相应的大小，也可能是以一些随机的内容加上去。