#-\*-\*coding:utf-8-\*-

*"""*

*python中对文件、文件夹（文件操作函数）的操作需要涉及到os模块和shutil模块。*

*"""*

import os

os.getcwd() #得到当前工作目录，即当前Python脚本工作的目录路径

os.listdir(*r'C:\E\workspace'*) #返回指定目录下的所有文件和目录名

*os.remove(r'C:\E\test\test.py') #函数用来删除一个文件,不能用来删除目录*

os.removedirs(r'C:\E\test1') #删除多个目录,只能删除目录，如果目录下包含文件删除时会抛出错误

print os.path.isfile(*r'C:\E\test\string.py'*)#检验给出的路径是否是一个文件

print os.path.isdir(*r'C:\E\test'*)#检验给出的路径是否是一个目录

print os.path.isabs(*r'C:\E\zzb'*) #判断是否是绝对路径,只根据名称来判定，不会检测这个路径是否存在

print os.path.exists(*r'C:\E'*)#检测路径是否真实存在，不包括文件

print os.path.split(*r'C:\E\test\string.py'*)#返回一个路径的目录名和文件名

print os.path.splitext(*r'C:\E\test\string.py'*)#('C:\\E\\test\\string', '.py')分离扩展名

print os.path.dirname(*r'C:\E\test\string.py'*)#获取路径名

print os.path.basename(*r'C:\E\test\string.py'*)#获取文件名

#os.system('cmd')#运行shell命令

#os.getenv() #读取环境变量

#os.putenv() #设置环境变量

os.environ # 获取环境变量

print os.linesep #给出当前平台使用终止符,Windows使用'\r\n'，Linux使用'\n'而Mac使用'\r'

print os.name #指示你正使用平台， 对于Windows，它是'nt'，而对于Linux/Unix用户，它是'posix'

#os.rename(r'C:\E\test\stringnew.py', r'C:\E\test\string.py')#重命名

#os.makedirs(r'C:\E\PYTHON') #创建多级目录，*makedirs(path [, mode=0777])*

#os.mkdir('test')#创建单个目录

print os.stat(*r'C:\E\test\string.py'*) #获取文件属性

#os.chmod(path, mode) #修改文件权限和时间戳

#os.\_exit(0) #一般来说os.\_exit()用于在线程中退出,直接退 python程序，其后的代码也不会继续执行。sys.exit() 用于在主线程中退出。

print os.path.getsize(*r'C:\E\test\string.py'*) #获取文件大小

os.path.abspath(path) #返回绝对路径

os.path.basename(path) #返回文件名

os.path.commonprefix(list) #返回list(多个路径)中，所有path共有的最长的路径。

os.path.dirname(path) #返回文件路径

os.path.exists(path)  #路径存在则返回True,路径损坏返回False

os.path.lexists  #路径存在则返回True,路径损坏也返回True

os.path.expanduser(path)  #把path中包含的"~"和"~user"转换成用户目录

os.path.expandvars(path)  #根据环境变量的值替换path中包含的”$name”和”${name}”

os.path.getatime(path)  #返回最后一次进入此path的时间。

os.path.getmtime(path)  #返回在此path下最后一次修改的时间。

os.path.getctime(path)  #返回path的大小

os.path.getsize(path)  #返回文件大小，如果文件不存在就返回错误

os.path.isabs(path)  #判断是否为绝对路径

os.path.isfile(path)  #判断路径是否为文件

os.path.isdir(path)  #判断路径是否为目录

os.path.islink(path)  #判断路径是否为链接

os.path.ismount(path)  #判断路径是否为挂载点（）

os.path.join(path1[, path2[, ...]])  #把目录和文件名合成一个路径

os.path.normcase(path)  #转换path的大小写和斜杠

os.path.normpath(path)  #规范path字符串形式

os.path.realpath(path)  #返回path的真实路径

os.path.relpath(path[, start])  #从start开始计算相对路径

os.path.samefile(path1, path2)  #判断目录或文件是否相同

os.path.sameopenfile(fp1, fp2)  #判断fp1和fp2是否指向同一文件

os.path.samestat(stat1, stat2)  #判断stat tuple stat1和stat2是否指向同一个文件

os.path.split(path)  #把路径分割成dirname和basename，返回一个元组

os.path.splitdrive(path)   #一般用在windows下，返回驱动器名和路径组成的元组

os.path.splitext(path)  #分割路径，返回路径名和文件扩展名的元组

os.path.splitunc(path)  #把路径分割为加载点与文件

os.path.walk(path, visit, arg)  #遍历path，进入每个目录都调用visit函数，visit函数必须有

3个参数(arg, dirname, names)，dirname表示当前目录的目录名，names代表当前目录下的所有

文件名，args则为walk的第三个参数

os.path.supports\_unicode\_filenames  #设置是否支持unicode路径名

os.walk(top, topdown=True, onerror=None, followlinks=False)

可以得到一个三元tupple(dirpath, dirnames, filenames),

第一个为起始路径，第二个为起始路径下的文件夹，第三个是起始路径下的文件。

dirpath 是一个string，代表目录的路径，

dirnames 是一个list，包含了dirpath下所有子目录的名字。

filenames 是一个list，包含了非目录文件的名字。

这些名字不包含路径信息，如果需要得到全路径，需要使用os.path.join(dirpath, name).

**文件内建函数**

open(name[, mode[, buffering]]) -> file object

name是要打开的文件名字的字符串，可以是相对路径或绝对路径。mode代表文件打开的模式。buffering哟用于指示访问文件所采用的缓冲方式。其中0表示不缓冲，1表示只缓冲一行数据，任何其他大于1的值代表使用给定值作为缓冲区大小。

函数file()可与open（）互换使用，file可用来判断实例类型isinstance(open(‘1.txt’), file)

文件操作：

#-\*-coding:utf-8-\*-

import os

fp = open(*r'C:\E\test\text.txt'*, *'w+'*) #直接打开一个文件，如果文件不存在则创建文件,路径不存在则抛出异常

str = *"""*

*r 以读方式打开，文件必须存在，否则抛出IOError。*

*w 以写方式打开，文件不存在则创建文件，文件存在则原有内容被清除。*

*a 以追加模式打开 (从 EOF 开始, 必要时创建新文件)，文件不存在则创建新文件，不能读。*

*r+ 以读写模式打开（存疑？实际执行中发现不能写），文件必须存在，否则抛出IOError。*

*w+ 以读写模式打开 (参见 w )，文件不存在则创建文件。*

*a+ 以读写模式打开 ，追加读写(实际执行中发现一次操作读写不能同时执行)*

*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+ )*

*"""*

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

fp.writelines([*'1'*, *'2'*,*'3'*\*12]) # 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。

print fp.read(100) #fp.read([size]),size为读取的长度，以byte为单位

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

print fp.readlines() #把文件每一行作为一个list的一个成员，并返回这个list。

fp.flush() #把缓冲区的内容写入硬盘

print fp.fileno() #返回一个长整型的”文件标签“

print fp.isatty() #文件是否是一个终端设备文件（unix系统中的）

print fp.tell() #返回文件操作标记的当前位置，以文件的开头为原点

fp.seek(0) #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。

print fp.next() #fp.next() 返回下一行，并将文件操作标记位移到下一行。

fp.truncate(100) #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。

fp.close()

**目录操作：**  
**os.mkdir("file")**              创建目录  
复制文件：  
**shutil.copyfile("oldfile","newfile")**      oldfile和newfile都只能是文件  
**shutil.copy("oldfile","newfile")**         oldfile只能是文件夹，newfile可以是文件，也可以是目标目录  
复制文件夹：  
**shutil.copytree("olddir","newdir")**       olddir和newdir都只能是目录，且newdir必须不存在  
重命名文件（目录）  
**os.rename("oldname","newname")**      文件或目录都是使用这条命令  
移动文件（目录）  
**shutil.move("oldpos","newpos")**  
删除文件  
**os.remove("file")**  
删除目录  
**os.rmdir("dir")**只能删除空目录  
**shutil.rmtree("dir")**  空目录、有内容的目录都可以删  
转换目录  
**os.chdir("path")**换路径