**Day08笔记**

day08(上午)：

**1.os模块&os.path模块**：

os模块的作用：管理和维护目录以及文件

os.path模块的作用：管理路径的(物理地址)

os模块中的属性和函数：

**属性：  
name：返回当前正在使用的操作系统的内核版本；windows环境 --> nt  
environ：返回当前操作系统中所有的环境变量以及其内容，以键值对的形式进行存储管理(存入到environ中)  
environ.get(环境变量名)：获取某一个环境变量的内容，以字符串的形式返回**print(os.name)  
print(os.environ)  
print(os.environ.get(**'path'**))

**函数：**

**之后演示的函数如果需要传入path的，则既可以是绝对路径也可以是相对路径  
路径问题：  
绝对路径：称为完成路径；一串物理地址(字符串类型)，特点：带盘符  
相对路径：相对的概念，拿某一个位置作为参照物，得到相关的路径，理解为相对路径；区分：不带盘符  
getcwd()：返回当前正在被执行的文件的绝对路径(描述一串字符串格式的物理地址)  
listdir(path)：将path路径下的子目录名和文件名以字符串类型保存到列表中返回 【注意】不包含间接子内容**print(os.getcwd())  
print(os.listdir(**r'D:\python1809\day06'**))  
 **操作：创建目录  
mkdir(path)：创建单层目录，不能级联创建目录  
makedirs(path)：创建多级目录，也可以创建单层目录***# 如果路径下已经存在相同名字的目录，继续点击创建，会报错：FileExistsError*os.mkdir(**r'D:\python1809\day08\hello1'**)  
os.mkdir(**r'aa'**)  
 **对于mkdir()而言：只能一次创建一层目录，不能批量(级联)创建目录；会报错：FileNotFoundError**os.mkdir(**r'bb\cc\dd'**) ×  
os.makedirs(**r'bb\cc\dd'**)

**思考：makedirs()函数能不能创建单层目录？  
可以的；**os.makedirs(**r'bb\cc\dd\ee'**)

**删除目录&删除文件：  
【注意】：删除操作有风险，使用需谨慎！！！因为不走回收站...  
删除目录：  
rmdir(path)：删除单层目录，不能级联删除目录  
removedirs(path)：删除多层目录，也可以删除单层  
删除文件：  
remove(path)：删除文件**os.rmdir(**r'D:\python1809\day08\hello1'**) *#删除hello1目录*os.rmdir(**r'aa'**) *#删除aa目录*os.rmdir(**r'bb\cc\dd\ee'**) *#删除ee目录*

os.removedirs(**r'bb\cc\dd'**) *#级联删bb\cc\dd*  
os.removedirs(**r'D:\python1809\day08\hello'**) *#可删单层hello*  
  
os.removedirs(**r'aa\bb\cc'**)

os.remove(**r'abc.html'**) *#删除文件*

**rename(src,dest)：重命名目录或者文件**os.rename(**r'aa.html'**,**r'bb.html'**) *#相对路径下，把文件名为aa.html改为bb.html*  
os.rename(**r'D:\python1809\day06\note.txt'**,**r'D:\python1809\day06\笔记.txt'**) *#绝对路径下，把文件名note改为笔记*  
  **演示os模块下的path模块的使用：**

**join(first,second)：将first和second两部分内容(字符串数据)拼接得到一个新的串(描述物理地址)；【注意】：不关注路径是否真实存在**

**import** osp = os.path.join(**r'D:\python1809\day07'**,**'note.txt'**)  
p1 = os.path.join(**r'D:\python1809\day07'**,**'note1.txt'**)  
print(p) *#D:\python1809\day07\note.txt*print(p1) *#D:\python1809\day07\note1.txt* **getsize(path)：返回path路径(锁定到文件层面)所包含的字节量***print(os.path.getsize(r'D:\python1809\day06\冒泡排序思路分析图解.png'))  
print(os.path.getsize(r'D:\python1809\day04'))*

**exists(path)：判断path描述的物理路径是否真实存在；如果存在，返回True；反之，返回False  
【注意】：以下两个函数它们都包含了exists的作用  
isfile(path)：判断path描述的物理路径是否是一个文件；如果存在，返回True；反之，返回False  
isdir(path)：判断path描述的物理路径是否是一个目录；如果存在，返回True；反之，返回False**path = **r'D:\python1809\day07\note1.txt'**print(os.path.exists(path))  
print(os.path.isfile(path))  
print(os.path.isdir(path))  
 **dirname(path)：返回path中最后一个\前面的部分(串)，最终以字符串的形式返回  
basename(path)：返回path中最后一个\后面的部分(串)，最终以字符串的形式返回**path = **r'D:\python1809\day07\note.txt'**str1 = os.path.dirname(path)  
print(str1,type(str1)) *#D:\python1809\day07 <class 'str'>*str2 = os.path.basename(path)  
print(str2,type(str2)) *#note.txt <class 'str'>* **split(path)：将path变量中最后一个\前面的部分(串)，放入到元祖的第一个元素位；将path变量中最后一个\后面的部分(串)，放入到元祖的第二个元素位；最终返回元祖对象  
splitext(path)： 将path变量中最后一个.前面的部分(串)，放入到元祖的第一个元素位；将path变量中最后一个.以及其后面的部分(串)，放入到元祖的第二个元素位；最终返回元祖对象**path = **r'D:\python1809\day07\note.txt'**tp1 = os.path.split(path)  
print(tp1,type(tp1)) *#('D:\\python1809\\day07','note.txt') <class 'tuple'>*tp2 = os.path.splitext(path)  
print(tp2,type(tp2)) *#('D:\\python1809\\day07\\note','.txt') <class 'tuple'>*

**2.内存和硬盘：**

内存：计算机硬件的组成部分之一，它是一种容器，用来存储数据，处理数据速度快，

存储的数据量小，断电死机内部数据就丢失了，短暂性存储数据；

硬盘：计算机硬件的组成部分之一，它是一种容器，用来存储数据，处理数据速度慢，

存储的数据量大，断电死机内部数据不丢失了，持久存储数据；

3.字节和字符：

字节：作为计算机存储数据的单位(默认)，底层唯一能够识别并且运算的都是字节

字符：人类将字节封装为另一个能够直接识别的数据单位，底层还是字节

总结：**计算机中一切数据皆字节**

4.如何区分字节文件和字符文件

如果使用文本编辑器打开能看得懂(不会乱码)，那就说明是字符文件；

反之，是字节文件

问题一：常见的字节和字符文件有哪些？(枚举出一些来)

**字节文件**：图片文件(.jpg、.png...)、视频文件(.avi、.mp4、rmb...)、音频文件(.mp3...)

**字符文件**：.txt、.py、.js、.java、.html、.css、.php...

问题二：.doc结尾的文件是字节还是字符文件？

是字节文件，因为.doc结尾的文件中既可以包含字符内容，也可以包含图片、颜色设置...

-----------------------------------------------------------------------------------------

day08(下午)：

**1.字符编码(字符集)：**

ascii码表：u.s.a设计出来的，字符范围非常少；不包含很多其它国家的文字，英文字符占用内存1个字节

utf-8码表：现今比较通用的一张编码表，包含世界上所有国家的文字内容，范围：0~65535之间， 1个汉字占用内存3个字节，1个英文字符占用内存1个字节

gbk码表：gbk属于gb2312的扩充版，兼容gb2312中的所有字符，并且加入了更多的一些汉字内容；1个汉字占用内存2个字节，1个英文字符占用内存1个字节

在我们当前中国环境下，大多数情况默认就是使用gbk作为编码

**2.编码和解码：**

编码：使数据从看得懂到看不懂，就是编码；

解码：使数据从看不懂到看得懂，就是解码；

【注意事项】：

如果编码和解码不一致会怎么样？

1).会出现乱码现象

2).会出现报错现象

总结：不管是乱码还是报错，都是不理想的结果，我们都需要避免

字符串中的两个函数引入：

encode(encoding,errors)：对字符串数据进行编码操作，得到一个bytes类型的对象

decode(encoding,errors)：对字符串数据进行解码操作，理解：将字节还原为str类型的数据**演示str类型中encode()和decode()函数的使用：**

str1 = **'哈哈呵呵嘿嘿'***#编码：encode()*b = str1.encode(encoding=**'gbk'**)  
print(b,type(b))  
*#解码：decode()*str2 = b.decode(encoding=**'gbk'**)  
print(str2,type(str2))

*#b'\xb9\xfe\xb9\xfe\xba\xc7\xba\xc7\xba\xd9\xba\xd9'<class 'bytes'>*

*#哈哈呵呵嘿嘿 <class 'str'>*

str1 = **'哈哈呵呵嘿嘿'***#编码：encode()*b = str1.encode(encoding=**'utf-8'**)  
print(b,type(b))  
*#解码：decode()*str2 = b.decode(encoding=**'utf-8'**)  
print(str2,type(str2))  
*# b'\xe5\x93\x88\xe5\x93\x88\xe5\x91\xb5\xe5\x91\xb5\xe5\x98\xbf\xe5*

*\x98\xbf' <class 'bytes'>  
# 哈哈呵呵嘿嘿 <class 'str'>*

str1 = **'哈哈呵呵嘿嘿'***#编码：encode()*b = str1.encode(encoding=**'utf-8'**)  
print(b,type(b))  
*#解码：decode()*str2 = b.decode(encoding=**'gbk'**)  
print(str2,type(str2))  
*# b'\xe5\x93\x88\xe5\x93\x88\xe5\x91\xb5\xe5\x91\xb5\xe5\x98\xbf\xe5*

*\x98\xbf' <class 'bytes'>  
#鍝堝搱鍛靛懙鍢垮樋 <class 'str'>*

str1 = **'哈哈呵呵嘿嘿'***#编码：encode()*b = str1.encode(encoding=**'gbk'**)  
print(b,type(b))  
*#解码：decode()*str2 = b.decode(encoding=**'utf-8'**,errors=**'ignore'**)*#不加errors就会报错*print(str2,type(str2))

*# b'\xb9\xfe\xb9\xfe\xba\xc7\xba\xc7\xba\xd9\xba\xd9'<class 'bytes'>  
# ǺǺٺ <class 'str'>*

**3.文件读写**

满足以下3个步骤：

1).打开文件

2).操作数据(读、写)

3).关闭文件 --> 不要忘记

分析：打开文件：

格式：**open（path,打开方式,[encoding=],[errors=]）**

参数解释：path：文件的路径(字符串体现)

打开方式：

r： 只读字符

r+： 在只读的基础上加入写操作

w： 只写字符

w+：在只写的基础上加入读操作

a： 追加字符(保留原本的，不会覆盖掉)

a+： 在追加的基础上加入一个读操作

rb： 只读字节

wb：只写字节

总结：所有带+号的方式，我们不去采用

encoding：确定字符集(编码、解码)；如果不定义，默认使用gbk进行编码和解码使用

errors：取值：'ignore'，定义了errors = 'ignore'；假设编码解码不一致，不会报错，但是会乱码，不定义的话，就直接报错了(出现异常)

操作数据：

读操作：

read()：一次将文件中所有的内部读到程序中

read(num)：一次将num个字符读入到程序中

readline()：每次读取文件中的一行数据(字符串数据)

readlines()：将文件中所有的行都以字符串的形式存入到一个列表中返回

tell()：返回文件描述符所定位到的已读字节数

seek(num)：将文件描述符回到num个字节位置

写操作：

'w'或者'wb'特点：会覆盖之前的内容

write(o)：将o中的数据写入到文件中

关闭文件：

记住一行代码搞定：调用close()函数即可

**演示读取文件中的数据：****#假设a.txt文件中的内容是 abcdefg1234567**

path = **r'D:\python1809\day08\3-文件读写\a.txt'***#1.打开文件*fr = open(path,**'r'**) *#打开方式都要单引号引起来 'r'*  
*#2.读取文件中的数据*content = fr.read(3)  
print(content) *#得到 abc 读的是字符*  
content = fr.read(4)  
print(content) *#得到 defg 读的是字符*  
content = fr.read()  
print(content) *#得到 1234567*

*#以下代码的可得，文件描述符已经读取到了文件末尾，再无内容可读...*

content = fr.read()  
print(content) *#再来以下read（）得到的是空行*

*#tell()：返回已读的字节数*print(fr.tell()) *#接上面的代码的到 14*  
*#seek(num)：将文件描述符移动到num字节位*fr.seek(4) *#表示起始位置从第四位后面的开始*

print(fr.read()) *#得到efg1234567*

1. **演示read()、tell()、seek()此三个函数参数&返回值的理解**

**#假设b.txt文件中的内容是 哈哈呵呵嘿嘿**

path = **r'b.txt'**fr = open(path,**'r'**,encoding=**'gbk'**) *#以gbk的方式进行解码*  
*#read(num)： num记录的是字符数*content = fr.read(2)  
print(content) *#得到 哈哈*  
*#tell()：返回已读的字节数 注意：不是字符*print(fr.tell())  *#得到 4*  
fr.read()  *#将内容全部读完*print(fr.tell()) *#得到 12*

*#seek(num)：num记录了字节位*  
fr.seek(4) *#标识符停在第二个哈上*   
print(fr.read()) *#得到 呵呵嘿嘿*

*#以下代码会报错，相当于停在第三个字节上，把第二个哈字一分为二，不可以*

*#fr.seek(3)  
#print(fr.read())*

fr.close()

**演示读文件时发生的各种报错现象：  
1).编码和解码不一致导致报错或者乱码现象 --> errors = 'ignore' 不会报错，显示乱码(相对友好)  
2).encoding如果不定义，默认使用gbk进行解码  
3).读操作：如果打开的文件不存在(路径有问题)，直接报错：FileNotFoundError**path = **r'b.txt'** *#文件b.txt要是不存在，就会报错*fr = open(path,**'r'**,encoding=**'utf-8'**,errors=**'ignore'**)  
print(fr.read())  
  
fr.close()

**演示readline()和readlines()的使用：  
readline()：读取文件中的每一行数据到程序(以字符串的形式返回)  
readlines()：将文件中的每一行数据以字符串的形式存入到列表中返回**

**#假设c.txt文件里面的内容是：**

**laoguo is a good man**

**laoguo is a nice man**

**laoguo is a handsome man**

**laoguo is a cool man**

**laoguo is a real man**

path = **r'c.txt'**fr = open(path,**'r'**)  
print(fr.readline())  
print(fr.readline())  
print(fr.readline())  
print(fr.readline())  
print(fr.readline())

fr.close()

*因为文件中每行字节就有换行符，print()y也有换行效果，使用得到如下结果*

*#laoguo is a good man*

*#laoguo is a nice man*

*#laoguo is a handsome man*

*#laoguo is a cool man*

*#laoguo is a real man*

fr = open(path,**'r'**)  
print(fr.readlines())  
*#['laoguo is a good man\n','laoguo is a nice man\n','laoguo is a handsome man\n','laoguo is a cool man\n', 'laoguo is a real man']*

**演示向文件中写入数据：** *特点：  
1).如果需要写入的文件并不存在，先创建文件，再写入数据  
2).'w'操作不能追加数据到文件中，只能覆盖之前的内容*

path = **r'd.txt'***#1.打开文件*fw = open(path,**'w'**)  
*#2.写入数据到文件中*fw.write(**'老郭帮帮哒'**)  
fw.write(**'杜老师壮壮哒'**)  
*#此时上面代码执行后，文件只能显示 杜老师壮壮哒 把上面的那行覆盖调了*  
  
fw = open(path,**'a'**) *# a表示追加字符(保留原本的，不会覆盖掉)*  
fw.write(**'\n老郭又回来啦'**) *#write没有自动换行的功能，所以用\n*  
*#此时上面代码执行后，文件显示 杜老师壮壮哒*

*老郭又回来啦*

*#3.关闭文件*fw.close()

**自定义函数copyFile：实现文件复制  
有形参(2个物理地址)、没有返回值(复制完就复制完了)  
粗糙版(开发不用)**

**def** copyFile(src,dest): *#src表示原始文件，dest表示目标文件*

*#1.打开两个文件：1个关联读操作、1个关联写操作* fr = open(src,**'rb'**) *#打开字节文件 rb也可以读字符文件，是万能 的打开方式，字节也没有编码的概念了* fw = open(dest,**'wb'**)  
  
 *#2.实现读和写操作* content = fr.read() *#读进来*  
 fw.write(content) *#写出去*

*# 3.关闭两个文件*

fw.close()  
fr.close()

copyFile(**r'C:\Users\Administrator\Desktop\打飞机.gif',r'C:\Users\**

**Administrator\Desktop\打飞机1.gif'**)

*#相当于一次性复制了*

**重构copyFile()函数，精细版(开发可以用)  
def** copyFile(src,dest):  
 *# 1.打开两个文件：1个关联读操作、1个关联写操作* fr = open(src, **'rb'**)  
 fw = open(dest, **'wb'**)  
  
 *# 2.实现读和写操作（无限循环的读写，并不是一下子全读进去）* **while** 1:  
 content = fr.read(1024)  *#每次读1024字节大小*  
 **if not** content: *#not 取反*  
 **break  
 else**:  
 fw.write(content)  
  
 *# 3.关闭两个文件* fw.close()  
 fr.close()

copyFile(**r'C:\Users\Administrator\Desktop\a.avi'**,**r'C:\Users\Administrator\Desktop\b.avi'**)

*#当文件容量过大的时候，一次性复制，内存受不了，可以分成慢慢复制*

**内置函数：eval(str)**

参数：str类型的对象

作用：函数执行完毕后，将str字符串中字面量为iterable的内容，还原成为其相应的容器类型数据

举例：

'[1,2,3,4]' --> [1,2,3,4]

'(11,22,33)' --> (11,22,33)

str1 = **'[1,2,3,4]'**lt = list(str1)  
print(lt) *#['[','1',',','2',',','3',',','4',']']*

lt = eval(str1)  
print(lt,type(lt))  *#[1, 2, 3, 4] <class 'list'>*str2 = **"{'name':'李雷','age':25}"**dic1 = eval(str2)  
print(dic1,type(dic1))  *#{'name': '李雷','age': 25}<class 'dict'>*