Python 程序设计基础

江西财经大学-朱文强

第八章

文件操作

文件操作

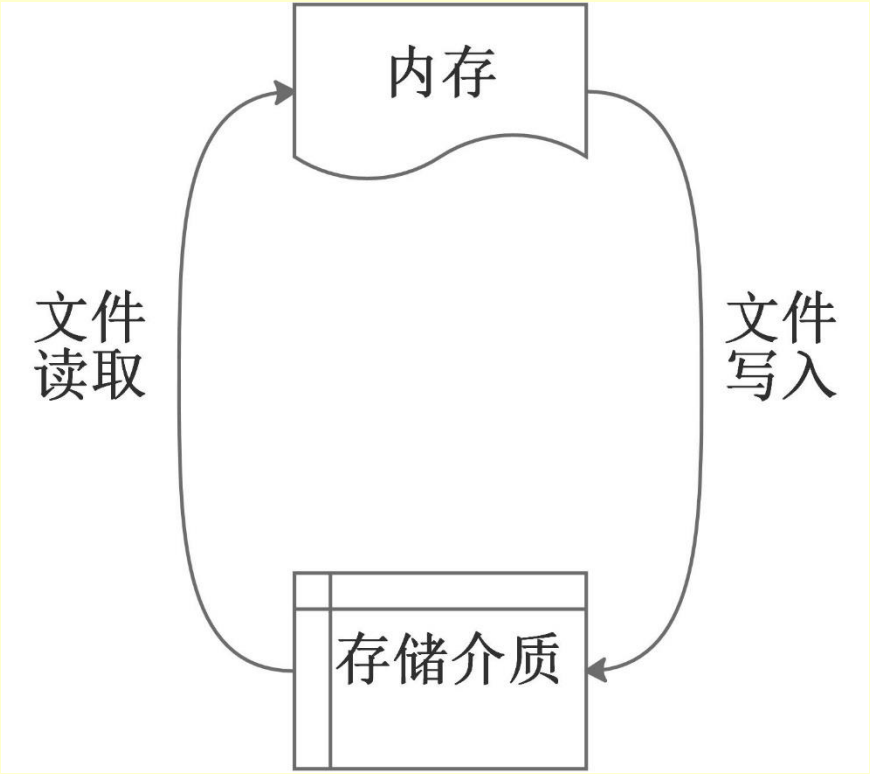
文件操作在程序开发中有着广泛的应用。一个程序项目，常常都需要数据文件的支持，数据的处理就涉及到文件的存取。**对于计算量大，业务繁杂的系统，为了避免重复计算，往往也会将程序执行的结果保存到文件中**。此外,从互联网下载图片、视频、音频保存到本地，也会涉及到文件操作。

本章首先讲述是普通文件读写；然后是对文件和文件夹的一些常见操作，比如对文件夹的遍历、创建文件夹等；最后是Excel文件的读写。

Excel文件是存储数据的较为普遍的方式，掌握对Excel文件的数据存取，是数据分析的重要基础。

1 文件操作及方法

数据通常以文件的形式存储在磁盘、U盘等外部存储介质中。需要对文件中的数据进行操作时，再将存储介质中的文件读取到内存中



文件与存储介质的关系

1 文件操作及方法

根据编码不同，可将**文件分为两类：文本文件和二进制文件**。

文本文件基于字符编码，存储的内容为普通字符串，不包括字体、字号、样式、颜色等信息，**可通过文本编辑器进行显示和编辑**。常见的.txt、.py、.csv文件等都是文本文件。

二进制文件基于值编码的，以字节形式存储数据内容，其编码长度根据值的大小长度可变。二进制文件通常会在文件头部的相关属性中定义表示值的编码长度。常见的视频、音频文件都是二进制文件。

1 文件操作及方法

Python中对**文本文件的操作通常按照以下三个步骤进行**(其他类型文件操作类似)。

(1)使用open()函数打开(或建立)文件, 返回一个File对象;

(2)使用File对象的读/写方法对文件进行读/写的操作;

(3)使用File对象的close()方法关闭文件。

1 文件操作及方法

open()函数的语法格式:

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

其部分**参数含义如下**:

file:表示要打开的文件名;

mode: 文件打开的模式, 默认为只读模式;

buffering: 设置缓存;

encoding: 设置文件的编码, 一般使用 “utf-8”编码;

1 文件操作及方法

errors: 设置编码错误的处理方式(忽略或报错);

newline: 设置新行处理方式;

closefd: 设置文件关闭时是否关闭文件描述符。

注意: file参数是必须的, 其他参数都是可选的。实际应用中通常传递三个参数: file、 mode和encoding。

程序演示:读取上节课程的.py文件(三步骤)

1 文件操作及方法

其中, **mode可以为下表的实参形式, 如下表所示**(可略过)。

Mode取值	权限			是否以二进制读写?	是否删除原内容?	文件不存在时是否产生异常?	文件指针的初始位置?
	读	写	附加				
r	是					是	头
r+	是	是				是	头
rb+	是	是		是		是	头
w		是			是	否，新建文件	头
w+	是	是			是	否，新建文件	头
wb+	是	是		是	是	否，新建文件	头
a			是			否，新建文件	尾
a+	是	是	是			否，新建文件	尾
ab+	是	是	是	是		否，新建文件	尾

1 文件操作及方法

文件的常用操作方法如下(可略过)。

方法	作用
read([size])	默认为文本文件读取，表示读取文件数据，将所有内容作为一个字符串返回；若给定正整数n，将返回n个的字符或字节(若不足n个字符，则返回所有内容)
readline()	单独读取文件的一行字符，包括"\n"字符
readlines()	把文件中的每行字符作为一个元素存入列表中，并返回该列表
write(str)	默认为文本文件写入，表示写入文本数据，返回值为写入的字节数
writelines([str])	列表中每个元素作为一行，逐个写入列表中所有的元素，不会自动换行，没有返回值

1 文件操作及方法

文件的常用操作方法如下。

方法	作用
close()	刷新缓冲区里还没写入的信息, 并关闭该文件
flush()	刷新文件内部缓冲，把内部缓冲区的数据立刻写入文件, 但不关闭文件
next()	返回文件下一行
tell()	返回文件指针的当前位置
seek(offset[,whence])	用于移动文件指针到指定位置，offset为需要移动的字节数；whence指定从哪个位置开移动，默认值为0。0代表从文件开头开始，1代表从当前位置开始，2代表从文件末尾开始

1.1 文件的读取

readme.txt:

The first line!

The second line.

The end line.

file_reader.py:

```
with open('work\readme.txt') as file_object:
    contents = file_object.read()
print(contents)
```

1.1 文件的读取

在这个例子中:

1、**open函数用于打开当前目录下指定路径的文件**，并将其作为File对象存储在file_object变量中，然后通过该文件对象对文件内容进行操作;

2、**关键字with可自动管理资源**，不论何种原因跳出with代码块，都能保证文件被正确关闭，且可在代码块执行完毕后自动还原进入该代码块时的现场(你也可以调用close方法来直接关闭文件，但一般**建议使用with来管理文件对象，以免因未妥善关闭文件而导致数据受损**);

1.1 文件的读取

3、**如果需要读取系统中任意位置的文件，则需要使用绝对路径(**windows系统有时不能正常解读路径中的斜杠，这时**应以原始字符串方式指定路径，**即在路径单引号前加上**r，**如果不加r字符，也可把路径字符串中的”\”改为”/”，还可以使用windows的相对路径”..”);

4、 fileinput.input(filename)方法也可以打开文件，但一般建议使用open函数来打开文件，其执行效率和稳定性都更好。

1.1 文件的读取-read()简化版

File对象的read([n])方法将整个文件的内容读取为一个字符串值, 如果给出参数n, 则读入前n个字符或字节流内容。

```
filename = ‘readme.txt’

f = open(filename)   # 需要显式关闭文件

print(f.read())    # print(f.read(20))

f.close()
```

1.1 文件的读取-readlines()

```
filename = ‘readme.txt’

with open(filename) as file_object:

    lines = file_object.readlines()

for line in lines:

    print(line, end=‘’)
```

注意：使用readlines方法可以将文件内容先读取出来，存储在一个列表中, 然后文件会自动关闭(以保护文件的内容)。

1.1 文件的读取-逐行读取

```
filename = ‘readme.txt’   #  from collections import Iterable
with open(filename) as file_object:
    for line in file_object:   # 迭代读取, 类似调用readlines()
        print(line)        # isinstance(file_object, Iterable)
```

除了readlines()方法，**还可以使用readline([n])方法来一行一行的读取数据，并可指定读取的字符数量**。假如somefile.readline()返回的是‘Hello, world!\n’，那么somefile.readline(5)返回的将是’Hello’。

1.1 文件的读取-pprint()

```
import pprint

filename = ‘readme.txt’

with open(filename) as file_object:

    lines = file_object.readlines()

pprint.pprint(lines)   # 打印原始字符内容
```

1.2 文件写入

```
filename = ‘write_test.txt’

with open(filename, ‘w’) as file_object:

    file_object.write(‘Writes the first line. \n’)

    file_object.write(‘Writes the second line. \n’)
```

1.2 文件写入

```
filename = ‘programming.txt’

a = [‘abc’, ‘def’, ‘ghi’]

with open(filename, ‘w’) as file_object:

    file_object.writelines(a)
```

注意：1、mode参数可为：只读模式(‘r’)、只写(‘w’)、只写(x)、追加(‘a’)、**读写(‘r+’，从当前指针处开始插入)、读写模式(+，需与‘rwa’结合使用)**，以及读写内容形式：二进制(b)及文本模式(t, 缺省为rt模式, t和b需要与其他模式结合使用)

1.2 文件写入

2、如果你省略了模式实参，Python将**以默认的只读(rt)模式打开文件，并自动将字节解码为Unicode字符；**

3、如果你要写入的文件不存在，**函数open()的'w'\'a'参数将自动创建它；如果指定的文件已经存在，参数'w'将在返回文件对象前清空该文件，'r+'会覆盖写入；参数'x'也会创建新文件，但如果目录下已存在该文件时，则报错(r+则是不存在则报错)；**

4、open函数还可以通过encoding参数指定读写文件时使用的编码格式（默认为Unicode），如encoding='iso-8859-1', encoding='gbk', encoding="utf-8", encoding="utf-8-sig"等(中文文件解码)；

1.2 文件写入

5、**Python只能将字符串写入文本文件**，要将数值数据存储到文本文件中，需要先使用函数str()将其转换为字符串格式；

6、与readlines()方法对应，**writelines()方法接受一个字符串列表**，并将这些字符串都写入到文件中(该方法写入时不会添加换行符，因此需要自行添加)；

7、**write(s)方法也可用来向文件写入一个字符串或字节流内容**，且更加灵活。

1.2 文件写入(迭代写入)

Python对于写入和读取文件是非常灵活的，还可以使用迭代的方式进行读写。

```
>>> f = open('somefile1.txt', 'w')
>>> print('First', ' line', file=f)      #会先写在缓冲区
>>> print('Third', ' and final', ' line', file=f)
>>> f.close()
>>> print(list(open('somefile1.txt')))  # or .read()
['First  line\n', 'Third  and final  line\n']
```

1.3 文件对象其他常用方法

seek(offset[，whence])：将文件指针移动到新的位置，offset表示相对于whence参数的偏移量，单位为字节。 whence取值为0(默认值)、1、 2, 分别表示文件开头位置, 当前位置和文件末尾位置(要使用1, 2参数, 打开文件需使用’rb’模式)。

1.3 文件对象其他常用方法

```
>>> f = open(r'c:\text\somefile.txt', 'w')
>>> f.write('0123456789012345678 9')
20
>>> f.seek(5)
5
>>> f.write(‘Hello, world!’)     #也会先写在缓冲区
13
>>> f.close()
```

1.3 文件对象其他常用方法

tell(): 返回文件指针的当前位置, 单位为字节 。

```
>>> f = open(r'c:\text\somefile.txt')
>>> f.read(3)  #read()如果不指定参数, 将读取整个文件
>>> f.read(7)
>>> f.tell()
```

注意：一般推荐使用with关键字来对文件进行操作，从而避免手动添加close()方法关闭文件时容易引发的异常，或用户忘记关闭文件时引发的问题。

1.3 文件对象其他常用方法

```
>>>s = '江西财经大学jufe’

>>>f = open('test.txt', 'w’)

>>>f.write(s)     #  10

>>>f.close()

>>>fp = open('test.txt’)

>>>print(fp.read(3))   # 江西财

>>>fp.seek(2)     #  2  将文件指针移到文件开头2字节处

>>>fp.read(2)     #  ‘西财’     读取2个字符
```

1.3 文件对象其他常用方法

```
>>>fp.seek(12)   # 12   将文件指针移动到离文件头12字节处
>>>print(fp.read(1))     #  j
>>>fp.seek(3)   # 3   将文件指针移动到离文件头3字节处
>>>print(fp.read(1))   # 报错, 文件指针在’财’字中间
>>>fp.close()
```

1.3 文件对象其他常用方法

flush(): 把缓冲区的内容写入文件, 但不关闭文件;

close(): 把缓冲区的内容写入文件, 同时关闭文件, 且释放文件对象;

truncate([size]): 删除从当前指针位置到文件末尾的内容。如果指定了size，则不论文件指针在什么位置都只留下前size字节内容, 其余删除。

2 常见文件的操作

2.1 JSON文件的操作

在Python中，**模块json提供了一种简单的方式进行数据存取，并方便的实现与其他程序共享这些数据**。

json(JavaScript Object Notation)格式最初是为JavaScript语言开发的，**是一种常见的轻量级数据格式，被包括Python在内的众多语言采用**。其采用键值对的方式进行数据存储:

“key”: “value”

json格式的文件扩展名为.json。

2.1 json文件的操作

对于数据的保存格式, **json有如下一些约定**:

(1) 数据保存在键值对中, 键与值用冒号分开;

(2) 键值对之间用逗号分隔;

(3) 大括号用于保存键值对数据组成的对象;

(4) 方括号用于保存键值对数据组成的数组.

json数据展示:

https://api.github.com/search/repositories?q=language:python&sort=stars

2.1 json文件的操作(例子)

{"本书作者": [

{“姓氏”: “朱",

“名字”: “文强",

“单位”: “江西财经大学", },

{“姓氏”: “钟",

“名字”: “元生",

“单位”: “江西财经大学", },

] }

2.1 json文件的操作

json库是处理json格式的Python标准库，**导入方式如下**：

```
import json
```

一般来说，**json格式的对象将被json库解析为字典，json格式的数组将被解析为列表**。

2.1 json文件的操作

json库处理json包含两个过程：编码(encoding)和解码(decoding)。编码是将Python数据类型变换成json格式的过程, 解码是对json文件解析数据, 对应到Python数据类型的过程。也即序列化和反序列化的过程。

注意：json可存储的数据类型只包括，字符串、整形、浮点型、布尔型、列表、字典和None。

2.1 json文件的操作

json库的操作函数主要有:

json.dumps(obj, sort_keys=False, indent=None, ensure_ascii),
将Python的数据类型转换为json格式字符串;
其中obj可以是列表或字典类型，当输入为字典类型时，dumps将其转换为json格式**字符串**;
sort_keys对字典元素按**key**进行排序; indent参数用于增加数据缩进, 使生成的字符串更具可读性;
ensure_ascii默认为True，为了避免网络传输中因编码不同带来的问题，其采用Unicode编码处理非西文字符；**如果要使json库输出中文字符, 可将其设置为False**。

2.1 json文件的操作

json.dump(obj, fp, sort_keys=Flase, indent=False), 与dumps()功能一致, 输出到文件fp;

json.loads(string), 将json格式字符串转换为Python的数据类型, 解码过程;

json.load(fp), 与loads()功能一致, 从文件fp读入。

2.1 json文件的操作

```
>>> import json
>>> dt = {'b': 2, 'c': 4, 'a': 6}
>>> s1 = json.dumps(dt)
>>> s2 = json.dumps(dt, sort_keys=True, indent=4)
>>> print(s1)     # s1, s2为字符串类型
>>> print(s2)  # s2试试
```

2.1 json文件的操作

```
>>> dt1 = json.loads(s1)

>>> dt2 = json.loads(s2)

>>> print(dt2)    # dt1, dt2为字典类型

{'a': 6, 'b': 2, 'c': 4}
```

2.1 json文件的操作

number_writer.py

```
import json

chars = {‘b’: 2, ‘c’: 4, ‘a’: 6}  # 也可试试列表

filename = ‘chars.json’

with open(filename, ‘w’) as file_object:

    #存储数据到json文件中(以字符串形式)

    json.dump(chars, file_object)
```

2.1 json文件的操作

number_reader.py

```
import json

filename = ‘chars.json’

with open(filename) as file_object:

    #加载json文件中的数据, 并将其转换为字典

    chars = json.load(file_object)

    # 也可以使用read()读取(字符串形式)

print(chars)
```

2.1 json文件的操作(例子)

```
import json # 加载json库

file = "name.json"

def get_stored_name():
    try:
        with open(file) as fr:
            name = json.load(fr)
    except FileNotFoundError:
        return None
```

2.1 json文件的操作(例子)

```
else:
        return name
def get_new_name():
    with open(file, 'w') as fw:
        name = input("请输入您的姓名:")
        json.dump(name, fw)
    return name
```

2.1 json文件的操作(例子)

```
def greet_user():
    name = get_stored_name()
    if name:
        print(f"欢迎回来, {name}!")
    else:
        name = get_new_name()
        print(f"欢迎首次登录:{name}.")
greet_user()
```

2.2 Excel文件的操作

Excel文件是一种二进制文件，它在数据分析、处理过程中使用非常频繁。Python官方发布版本中没有读写excel文件的模块，需要安装第三方模块来实现对excel文件的操作。较为简单而又**常用的第三方库有：xlrd(用于读取Excel文件)、xlwt(用于向Excel文件写入内容)，openpyxl(可用于Excel文件的读写)**。

在Anaconda集成开发环境中，则自带了这两个模块，可以直接使用。

2.2 Excel文件的操作(读取)

Excel文件读取涉及到的常用方法和属性如下表所示。

方法	属性
xlrd.open_workbook(文件名)	打开Excel文件发, 返回工作簿对象
sheet_by_index(索引)	根据索引, 获取工作簿对象的表单
sheet_by_name(名称)	根据名称, 获取工作簿对象的表单
nrows	返回表单对象的行数
ncols	返回表单对象的列数
cell_value(行序,列序)	获取表单对象的单元格内容
row_values(行序)	获取表单对象的某一行内容

2.2 Excel文件的操作(读取)

Excel文件读取的操作步骤:

(1)导入模块xlrd;

(2)打开Excel工作簿Book;

(3)指定工作簿中的表单Sheet;

(4)根据行列序号读取内容。

注意：索引是从“0”开始，即“0”是指第一个表单，“1”表示第二个表单。

2.2 Excel文件的操作(读取)

```
import xlrd    # 读取Excel文件中的内容, 并显示输出

wb = xlrd.open_workbook("school.xls")

sheet = wb.sheet_by_index(0)

schools = []

for row in range(sheet.nrows):

    school = []

    for col in range(1, sheet.ncols):

        content = sheet.cell_value(row,col)
```

2.2 Excel文件的操作(读取)

```
        school.append(content)

    schools.append(school)

for school in schools:

    print(school)
```

2.2 Excel文件的操作(写入)

Excel文件写入涉及到的常用方法和属性如下表所示。

方法	说明
xlwt.Workbook()	创建Excel文件
add_sheet(名称)	添加表单(Workbook类)
xlwt.XFStyle()	定义样式
xlwt.Font()	定义字体
xlwt.Alignment()	定义对齐方式
write(行序,列序,内容,样式)	向单元格添加内容(Worksheet类)
write_merge(行序1,行序2,列序1,列序2,内容,样式)	合并指定范围单元格,并指定内容

2.2 Excel文件的操作(写入)

Excel文件写入的操作步骤。

(1)导入模块xlwt;

(2)构造工作簿Workbook;

(3)为工作簿添加表单Worksheet;

(4)根据行列序号写入内容;

(5)保存文件。

案例见:

ch08_18.py; ch08_19.py; ch08_20.py

3 文件与文件夹的操作(进阶)

Python提供的对文件和文件路径的常用操作方法，大部分都在os库以及os.path这个库中，这两个库是Python自带的标准模块, 不需要额外安装。

os库让用户能够访问文件系统提供的服务，能够进行文件的重命名、更换当前路径等操作。os.path库提供了对文件路径的判断、切分、连接等操作方法。

3 文件与文件夹的操作(进阶)

os.environ: 查看系统环境变量, 如os.environ[‘pythonpath’]

os.system(‘command’): 运行指令, 如os.system(‘dir’)

os.sep: 查看操作系统所使用的路径分隔符; # \\

os.pathsep: 查看不同路径同时存在时的分隔符; # ;

os.linesep: 查看文本文件中的行分隔符; # \r\n

os.urandom(size): 查看系统自带的随机源;

3 文件与文件夹的操作(进阶)

os.chmod(): 改变文件的访问权限;

os.remove(path): 删除指定的文件;

os.removedirs(path1/path2…): 删除多级目录;

os.rename(src, dst): 重命名文件或目录;

os.stat(path): 返回文件的所有属性;

os.listdir(path): 返回path目录下的文件和目录列表;

os.startfile(filepath[, operation]): 使用关联的应用程序打开指定文件;

3 文件与文件夹的操作(进阶)

os.getcwd(): 获得当前工作目录;

os.chdir(path): 更换当前目录为path目录;

os.mkdir(path): 创建path目录;

os.makedirs(path1/path2…): 创建多级path目录;

os.rmdir(path): 删除path目录;

os.get_exec_path(): 返回可执行文件的搜索路径;

3 文件与文件夹的操作(进阶)

os.walk(path，topdown=True，onerror=None)：对被传入的文件夹进行遍历, 该方法返回一个元组生成器, 用于生成3个值。

1. 当前文件夹的字符串名称;

2. 当前文件夹中子文件夹的字符串名称列表;

3. 当前文件夹中文件的字符串名称列表。

遍历结束后, 重新回到最初的当前文件夹。

3 文件与文件夹的操作(进阶)

os.path.abspath(path): 返回绝对路径;

os.path.exists(path): 判断文件是否存在;

os.path.getatime(filename): 返回文件的最后访问时间;

os.path.getctime(filename): 返回文件的创建时间;

os.path.getmtime(filename): 返回文件的最后修改时间;

os.path.getsize(filename): 返回文件大小，以字节为单位;

os.path.isabs(path): 判断path是否为绝对路径;

os.path.isdir(path): 判断path是否为目录;

3 文件与文件夹的操作(进阶)

os.path.isfile(path): 判断path是否为文件;

os.path.join(path, *paths): 连接两个或多个path, 如将文件夹和文件组合得到绝对路径;

os.path.split(path): 对路径进行切分, 以列表形式返回;

os.path.splitext(path): 从路径中分割文件的扩展名;

os.path.splitdrive(path): 从路径中分割驱动器名称;

os.path.basename(path): 返回path的文件名;

os.path.dirname(path): 返回path的目录名。

3 文件与文件夹的操作(进阶-例子)

```
>>> import os
>>> os.getcwd()        #获得当前目录
'C:\\Users\\Administrator'  # 最后没有'\\'
>>> os.chdir('D:\\Python\\automate')        #更换当前目录
>>> os.getcwd()
'D:\\Python\\automate'
```

3 文件与文件夹的操作(进阶-例子)

```
>>> import os   # mkdir()不能创建整个路径
>>> os.makedirs('c:\\delicious\\walnut') #创建整个路径
>>> os.path.abspath('.')        # 返回当前目录的绝对路径
'C:\\Users\\Administrator'
>>> os.path.isabs('.')          #判断当前目录是否为绝对路径
False
>>> os.path.isabs(os.path.abspath('.'))
```

3 文件与文件夹的操作(进阶-例子)

```
>>>path = r"c:\windows\system32\expand.exe"
>>>os.path.exists(path)   # 判断是否存在路径
>>>True
>>>os.path.isfile(path)   # 判断该路径是否为文件
>>>True
>>>os.path.isdir(path)    # 判断该路径是否为文件目录
```

3 文件与文件夹的操作(进阶-例子)

```
>>> os.path.relpath(‘c:\\windows’, ‘c:\\users’)
#返回c:\\users目录到c:\\windows目录的相对路径的字符串
'..\\windows‘
>>> path = ‘C:\\windows\\system32\\explore.exe‘ # r或’/’也行
>>> os.path.basename(path)     # 返回路径下的文件名
'explore.exe'
>>> os.path.dirname(path)      # 返回除文件名外的路径
'C:\\windows\\system32'
```

3 文件与文件夹的操作(进阶-例子)

```
>>> os.path.split(path)   #返回路径和文件名的元组
('C:\\windows\\system32', 'explore.exe')
>>> path.split(os.path.sep)   #os.path.sep为系统的路径分隔符
['C:', 'windows', 'system32', 'explore.exe']
>>> path = "C:\\Users\\Default\\NTUSER.DAT.LOG1"
>>> os.path.getsize(path)      #返回指定文件的大小(Byte)
262144
```

3 文件与文件夹的操作(进阶-例子)

```
>>> os.path.join('usr', 'bin', 'spam')    #'usr\\bin\\spam‘
>>> myF = ['a.txt', 'b.txt', 'c.txt']
>>> os.makedirs(r“d:\users\aws”)  # 构建目录
for f in myF: # r”C:\users\aws” #
    path1 = os.path.join('d:\\users\\aws', f)
    print(path1)
    with open(path1, "w") as fw:
        fw.write(f"{path1}")
```

3 文件与文件夹的操作(进阶-例子)

```
>>> path = "D:\\Python\\automate"
>>> os.listdir(path)     #返回路径下所有的文件夹和文件名称
['chapter 2', 'chapter 3', 'chapter 6', 'chapter 7', 're.docx’]
>>>fdir = 'D:\Python\python_lesson\chapter7'
>>>for fn in os.listdir(fdir):
…     totalSize = totalSize + os.path.getsize(os.path.join(fdir, fn))
>>>print(totalSize)
```

3 文件与文件夹的操作(进阶-例子)

os.startfile(r'str'): 启动系统中的程序, 如:

>>>os.startfile(r'C:\Program Files\Python311\python.exe')

#以下方法也可, 但Python要在path环境变量中先配置

>>>os.system('jupyter notebook')

注意: 如果要在web浏览器中打开网站, 只需执行以下命令:

>>> import webbrowser

>>> webbrowser.open("http://www.python.org")

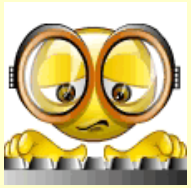
3 文件与文件夹的操作(进阶-例子)

```
# 返回当前目录下所有以’.py’结尾的文件名
>>>print([fname for fname in os.listdir(os.getcwd())\
        if os.path.isfile(fname) and fname.endswith('.py’)])
>>>f_ls = [fn for fn in os.listdir('.’)  if fn.endswith('.txt’)]
>>>for fn in f_ls:   # 将当前目录下所有后缀名为.txt的文件
        newn = fn[:-4] + ’.dat’    #  其后缀名更换为.dat
        os.rename(fn, newn)
        print(fn,"更名为:",newn)
```

3 文件与文件夹的操作(进阶-例子)

```
import os   # 使用os.walk()对D:\Python目录进行遍历
for folder, subfolders, filenames in os.walk(r'D:\Python'):
    print(f‘当前文件夹:{folder}')
    for subfolder in subfolders:
        print(f‘子文件夹:{subfolder}')
    for filename in filenames:
        print(f‘文件:{foldername}')
```

练习



01 完成教材和PPT上的例题和练习

02 完成课后编程题5

谢 谢