# 第十一章 Python 文件操作

本节所讲内容：

11.1 文件对象

11.2 文件权限

11.3 文件操作

11.4 os 模块

## 11.1 文件对象

### 11.1.1 文件介绍

什么是文件？

示例如下：



狭义说：文本文件；广义说：超文本文件, 图片，声音，超链接，视频

Python中可以使用一个文件对象file来做大部分文件操作，它有两个函数file()和open()

Open函数是file函数的再封装！

今天我们重点介绍open()

open()想要读取或写入文件，必须使用python内置的open()函数来打开它，该函数创建一个文件对象，这将用来调用与之关联的其他支持方式

语法：

File object = open(path,mode,[buffering])

如何来创建一个文件对象

f = open()

path

文件路经: 前期的时候我们写成固定的（绝对和相对路经），也可以使用os模块来辅助完成

mode权限: r（read）读权限，w（write）写权限，a（add）追加写

buffering(了解)

是否启用缓存 系统默认(全缓存) -1 ； 0 不启用； 1启用行缓存

行缓冲就是说每一行刷新一次缓冲区，就是见到换行符的时候把缓冲区的内容送到指定位置

### 11.1.2 文件的作用

大家应该听说过一句话：“好记性不如烂笔头”。不仅人的大脑会遗忘事情，计算机也会如此，比如一个程序在运行过程中用了九牛二虎之力终于计算出了结果，试想一下如果不把这些数据存放起来，相比重启电脑之后，“哭了……”。可见，在把数据存储起来有做么大的价值。

使用文件的目的：

把一些数据存放起来，可以让程序下一次执行的时候直接使用，而不必重新制作一份，省时省力。

## 11.2 文件权限

案例：open('1.txt','r')

说明:

|  |  |
| --- | --- |
| 访问模式 | 说明 |
| r | 以只读方式打开文件。文件的指针将会放在文件的开头。这是默认模式。 |
| w | 打开一个文件只用于写入。如果该文件已存在则将其覆盖。如果该文件不存在，创建新文件。 |
| a | 打开一个文件用于追加。如果该文件已存在，文件指针将会放在文件的结尾。也就是说，新的内容将会被写入到已有内容之后。如果该文件不存在，创建新文件进行写入。 |
| rb | 以二进制格式打开一个文件用于只读。文件指针将会放在文件的开头。这是默认模式。 |
| wb | 以二进制格式打开一个文件只用于写入。如果该文件已存在则将其覆盖。如果该文件不存在，创建新文件。 |
| ab | 以二进制格式打开一个文件用于追加。如果该文件已存在，文件指针将会放在文件的结尾。也就是说，新的内容将会被写入到已有内容之后。如果该文件不存在，创建新文件进行写入。 |
| r+ | 打开一个文件用于读写。文件指针将会放在文件的开头。 |
| w+ | 打开一个文件用于读写。如果该文件已存在则将其覆盖。如果该文件不存在，创建新文件。 |
| a+ | 打开一个文件用于读写。如果该文件已存在，文件指针将会放在文件的结尾。文件打开时会是追加模式。如果该文件不存在，创建新文件用于读写。 |
| rb+ | 以二进制格式打开一个文件用于读写。文件指针将会放在文件的开头。 |
| wb+ | 以二进制格式打开一个文件用于读写。如果该文件已存在则将其覆盖。如果该文件不存在，创建新文件。 |
| ab+ | 以二进制格式打开一个文件用于追加。如果该文件已存在，文件指针将会放在文件的结尾。如果该文件不存在，创建新文件用于读写。 |

文件对象的属性（方法）:

|  |  |
| --- | --- |
| File.closed | 如果文件被关闭返回true，否则返回false |
| File.mode | 返回文件被打开的访问模式 |
| File.name | 返回文件的名称 |

f = open('1.txt','r',encoding='utf-8')  
*#判断是否关闭*print(f.closed)  
*#打印权限*print(f.mode)  
*#打印文件的name*print(f.name)  
*#文件关闭*f.close()

运行结果如下：

False  
r  
1.txt

## 11.3 对文件的操作

### 11.3.1 文件读取（面试题）

f = open('1.py','r',encoding='utf-8')  
*#遍历进行读取*for i in f:  
 print(i)  
f.close()

2、读取方法

调用方法read()会一次性读取文件的全部内容，但是如果文件有10G，内存就爆了，会导致程序卡死，所以，要保险起见，可以反复调用read(size)方法，每次最多读取size个字节的内容。

另外，调用readline()可以每次读取一行内容，调用readlines()一次读取所有内容并按行返回list。因此，要根据需要决定怎么调用。

f = open('1.py','r',encoding='utf-8')  
*#读取所有内容  
# print(f.read(10))  
#读取一行  
# print(f.readline())  
#读取所有 并且按照list承载*print(f.readlines())  
f.close()

### 11.3.2 写（write）

f = open('2.txt','w',encoding='utf-8')  
*# f.write('你好呀，hello,world')  
# f.writelines(str(i)+'\n' for i in range(10))*f.writelines(['a','b','c'])  
f.close()

两者的区别在哪呢？

file.write(str)的参数是一个字符串，就是你要写入文件的内容.  
file.writelines(sequence)的参数是序列，比如列表，它会迭代帮你写入文件。

3、close

close（）方法关闭一个已经打开的文件对象，文件关闭后不能再执行读写操作。

1、如果不去手动关闭文件，会一直占用我们的内存；

思考：如果读写文件出错，出现IOError的错误，我们后面的close（）方法就不会调用，也就是说我们的文件就不会关闭，这个该如何？

try:  
 f = open('2.txt', 'r',encoding='utf-8')  
 print(f.read())  
except Exception as e:  
 print(e)  
finally:  
 *# f.close()#关闭* pass  
print(f.closed)

2、with

Python引入了with语句来自动帮我们调用close（）方法

with open('2.txt','r',encoding='utf-8') as f:  
 print(f.read())  
print(f.closed)

with简单的导入原理（了解一下）：  
（１）紧跟with后面的语句被求值后，返回对象的“\_\_enter\_\_()”方法被调用，这个方法的返回值将被赋值给as后面的变量；   
（２）当with后面的代码块全部被执行完之后，将调用前面返回对象的“\_\_exit\_\_()”方法。

class A:  
 def \_\_enter\_\_(self):  
 print('\_\_enter\_\_() is called')  
 def \_\_exit\_\_(self, exc\_type, exc\_val, exc\_tb):  
 print('\_\_exit\_\_() is called ')  
with A() as a:  
 print('got instance')

运行结果如下：

\_\_enter\_\_() is called

got instance

\_\_exit\_\_() is called

3、对图片文件进行操作

#图片文件以二进制形式打开，写入

*#对图片文件操作*with open('for.png','rb') as png1:  
 with open('shu.png','wb') as png2:  
 png2.write(png1.read())  
#第二种（不推荐）  
with open('for.png','rb') as png1, open('shu.png','wb') as png2:  
 png2.write(png1.read())

### 11.3.3 指针



1. tell() 方法返回文件的当前位置，即文件指针当前位置。

语法

tell() 方法语法：fileObject.tell()

f = open('1.txt','r+',encoding='gbk')  
print(f.name)  
*#读取一行*line = f.readline()  
print(line)  
*#获取为你家指针当前的位置*pos = f.tell()  
print('当前只针的位置%s'%pos)  
*#关闭*f.close()

运行结果如下：

1.txt

当我还只有六岁的时候，在一本描写原始森林的名叫《真实的故事》的书中， 看到了一副精彩的插画，画的是一条蟒蛇正在吞食一只大野兽。页头上就是那副 画的摹本。

当前只针的位置152

2. seek（）

seek()方法用于移动文件读取指针到指定位置。  
 seek()语法格式：fileObject.seek(offset[, whence])

offset 移动的长度（字节）

whence 相对位置；0从开头（默认），1从当前，2从末尾

如果offset为负数，表示从后往前移动n个字节

f = open('1.txt','r+',encoding='gbk')  
print(f.name)  
*#赌气一行*line = f.readline()  
print(line)  
f.seek(0,0)  
print(f.readline())  
*#获取为你家指针当前的位置*pos = f.tell()  
print('当前只针的位置%s'%pos)  
*#关闭*f.close()

运行结果如下：

1.txt

当我还只有六岁的时候，在一本描写原始森林的名叫《真实的故事》的书中， 看到了一副精彩的插画，画的是一条蟒蛇正在吞食一只大野兽。页头上就是那副 画的摹本。

当我还只有六岁的时候，在一本描写原始森林的名叫《真实的故事》的书中， 看到了一副精彩的插画，画的是一条蟒蛇正在吞食一只大野兽。页头上就是那副 画的摹本。

当前只针的位置152

### 11.3.4 实战

1、实战：电子书翻页功能

分析：电子书现在有自动翻页功能（死循环，深度循环）；有手动输入功能（在深度循环中加入判断条件）；

import time  
def reader(path,line = 3):  
 *#打开文件读取* with open(path,'r',encoding='gbk') as f:  
 *#我们把指针指向文件的末尾* f.seek(0,2)  
 *#告诉我文件末尾的指针的位置总字节量* end = f.tell()  
 *#指针指向开头* f.seek(0,0)  
 *#你是否要开启自动？* auto = input('是否开启自动:(y/n)?')  
 if auto == 'y':*#是* while True:*#深度循环  
 #循环三次 读取三行* for i in range(line):  
 print(f.readline())  
 time.sleep(2)*#美都区三行会sleep2两秒  
 # 判断如果你的文件指针指向文件末尾 我就强制break* if f.tell() == end:  
 break  
 else:  
 con = 'N'  
 f.seek(0,2)  
 end = f.tell()  
 f.seek(0,0)  
 while True:  
 if con == 'N':  
 for i in range(line):  
 print(f.readline())  
 else:  
 print('请输入N')  
 if f.tell() == end:  
 break  
 con = input('>>>')  
reader('1.txt')

运行结果如下：  
是否开启自动:(y/n)?n

当我还只有六岁的时候，在一本描写原始森林的名叫《真实的故事》的书中， 看到了一副精彩的插画，画的是一条蟒蛇正在吞食一只大野兽。页头上就是那副 画的摹本。

这本书中写道：“这些蟒蛇把它们的猎获物不加咀嚼地囫囵吞下，尔后就不 能再动弹了；它们就在长长的六个月的睡眠中消化这些食物。”

>>>N

## 11.4 os模块

### 11.4.1 Python系统路径和其他一些操作模块

|  |  |
| --- | --- |
| os常用功能 | |
| os.sep | 符合当前系统的路径分割符，Linux/windows\ |
| os.name | 返回操作系统类型windows“nt”Linux“posix” |
| os.rename | 更改文件名，传递两个参数（旧文件名，新文件名） |
| os.getcwd | 返回当前的工作目录 |
| os.listdir | 列出指定目录下的目录和文件 |
| os.chdir | 修改当前的工作路径 |
| os.mkdir | 创建目录 |
| os.makedirs | 递归创建目录 |
| os.remove | 删除文件 |
| os.rmdir | 删除文件夹（文件夹下内容为空） |
| os.removedirs | 递归删除文件夹 |
| os.system | 执行系统命令 |
| os.popen | 执行系统命令，会将结果以文件的形式返回 |
| os.walk | 通过在目录树中游走输出在目录中的文件名，向上或者向下 |
| os.path.join | 连接目录和文件名。os.path.join（path,name） |
| os.path.isfile | 判断指定对象是否为文件。是返回True,否则False |
| os.path.isdir | 判断指定对象是否为目录。是True,否则False |
| os.path.split | 返回路径的目录和文件名 |
| os.path.exists | 检验指定的对象是否存在。是True,否则False |
| os.path.getsize | 获取指定对象的文件大小 |
| os.path.abspath | 获得绝对路径 |
| os.path.basename | 返回文件名 |
| os.path.dirname | 返回文件所在目录 |

1. os.name()——判断现在正在实用的平台，Windows 返回 ‘nt'; Linux 返回’posix'

    rename(需要修改的文件名, 新的文件名) 也可以做剪切。

2. os.getcwd()——得到当前工作的目录。

3. os.listdir()——指定所有目录下所有的文件和目录名。

以列表的形式全部列举出来，其中没有区分目录和文件。

4. os.mkdir()——创建目录

>>>os.makedirs(‘a/b/c‘)

5. os.remove()——删除指定文件

6. os.rmdir()——删除指定目录（该目录不能为空）

注意：这样只能建立一层，要想递归建立可用：os.makedirs(‘x/y/z’)

7. os.path.isfile()——判断指定对象是否为文件。是返回True,否则False

>>>os.path.isfile(‘1.txt’)

True

8. os.path.isdir()——判断指定对象是否为目录。是True,否则False。例

>>>os.path.isdir(‘1.txt’)

False

9. os.path.exists()——检验指定的对象是否存在。是True,否则False.例：

>>>os.path.exists(‘1.txt’)

True

10. os.path.split()——返回路径的目录和文件名。

此处只是把前后两部分分开而已。就是找最后一个'/'。看例子：

os.path.split('E:/xuegod/python3')

('E:/xuegod', 'python3')

11. os.getcwd()——获得当前工作的目录（get current work dir)

12. os.system()——执行shell命令。

13. os.chdir()——改变目录到指定目录（谨慎更改）

案例：  
import os  
print('返回的本地路径为:',os.getcwd())  
path = './xuegod/'  
*#修改本地路径*os.chdir(path)  
print('新的本地路径为',os.getcwd())

运行结果如下：

E:\xuegod\_code\1\_12\_异常处理

E:\xuegod\_code\1\_12\_异常处理\xuegod

14. os.path.getsize()——获得文件的大小(字节)，如果为目录，返回0

>>>os.path.getsize('1.txt')

1644

15. os.path.abspath()——获得绝对路径。

print(os.path.abspath('1.txt'))

运行结果如下：

E:\xuegod\_code\1\_12\_异常处理\1.txt

16. os.path.join(path,name)——连接目录和文件名。

例：

import os  
file\_path = os.getcwd()  
new\_path = os.path.join(file\_path,'1.txt')  
print(new\_path)

运行结果如下：

E:\xuegod\_code\1\_12\_异常处理\1.txt

17.os.path.basename(path)——返回文件名

18. os.path.dirname(path)——返回文件所在目录

19. os.walk() 方法是一个简单易用的文件、目录遍历器，可以帮助我们高效的处理文件、目录方面的事情。在Unix，Windows中有效。

表达式：

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

top 代表的是你所要遍历的目录的地址, 返回的是一个三元组(root,dirs,files)。

root 所指的是当前正在遍历的这个文件夹的本身的地址。

dirs 是一个 list ，内容是该文件夹中所有的目录的名字(不包括子目录)。

files 同样是 list , 内容是该文件夹中所有的文件(不包括子目录)。

案例：

import os  
*#获取当前路径*path = os.getcwd()  
*#返回文件夹本身的地址，目录，文件*for root,dirs,files in os.walk(path):  
 for filename in files:  
 print(os.path.join(root,filename))

运行结果如下：

E:\workspace\importTest\1.png

E:\workspace\importTest\1.txt

E:\workspace\importTest\2.png

E:\workspace\importTest\2.py

……

### 11.4.2 如何批量修改文件夹下的文件名？（面试题）

import os  
m\_name = os.listdir('./xuegod/')  
*#对该目录下的文件遍历*for temp in m\_name:  
 *#新的文件名* new\_name = 'xuegod'+temp  
 *#更改文件名，前参数 代表老文件名，后面的参数代表新的文件名* os.rename('./xuegod/'+temp,'./xuegod/'+new\_name)

### 11.4.3 实战

传入一个文件路径，得到该路径下所有的文件目录?（面试题）

import os  
def iterbrowse(path):  
 *#返回路径，目录，文件；* for home,dirs,files in os.walk(path):  
 *#对文件进行遍历* for file in files:  
 *#拼接路径和name* print(os.path.join(home,file))  
*#处理当前的文件路径*new\_path = os.getcwd()  
*#调用*iterbrowse(new\_path)

输出结果如下：

E:\workspace\importTest\9\_1\_write\_test.py

E:\workspace\importTest\9\_2\_with\_test.py

E:\workspace\importTest\9\_3\_with\_test02.py

E:\workspace\importTest\9\_4\_tell\_test.py

E:\workspace\importTest\9\_5\_shizhan.py

E:\workspace\importTest\9\_6\_os\_test.py

E:\workspace\importTest\class\_test.py

E:\workspace\importTest\class\_test01.py

……

总结:

11.1 文件对象

11.2 文件权限

11.3 文件操作

11.4 os 模块