

Python基础Day11 文件与IO

基本文件操作

1. 创建和打开文件

在Python中，想要操作文件需要先创建或者打开指定的文件并创建文件对象。可以通过内置的open()函数实现。语法格式：

```
file = open(文件名,[mode[,buffering]])
```

file:被创建的文件对象

文件名:要创建或打开的文件，如果在同级目录下可以直接写文件名和后缀。如果不在同级目录需要写完整路径。

mode:可选参数，用于指定打开模式。默认为只读(r).

buffering:可选参数，用于指定读写文件的缓冲模式，值为0是表示不缓存，值为1时表示缓存，大于1时，表示缓存大小。默认为缓存模式。

值	说明
r	以只读模式打开。光标位于文件开头
rb	以二进制打开，同时也是只读模式，一般用于非文本，如图片，声音等。
r+	文件打开后可以读取内容，同时也可以添加内容并覆盖掉原来的内容
rb+	同上，但一般用于图片声音等
w	以只写模式打开
wb	同上，但一般用于图片，声音等。
w+	打开文件后，先清空原有文件内容，对这个空文件有读写权限。
wb+	同上，一般用于图片声音等
a	以追加模式打开，如果文件存在那么将内容添加到末尾，文件不存在则创建一个新的文件用于写入
ab	同上，但一般用于图片，声音等。
a+	以读写模式打开
ab+	同上，但一般用于图片，声音等。

```
# 打开一个图片文件。
file = open("pic.jpg", 'rb') # 以二进制打开图片
print(file) # 输出创建的文件对象

# 打开文件时指定编码格式
# open打开文件时默认用的时GBK编码，当打开不是GBK编码的文件时，会报错。
file = open('demo.txt', 'r', encoding='utf-8')
print(file)
```

2. 关闭文件

打开文件后，需要及时关闭，以免对文件造成不必要的破坏。关闭文件时可以使用文件对象的close()方法实现。

```
file.close()
```

3. 打开文件时使用with语句

在使用open()函数打开文件时，一定要记得使用close()方法将其关闭。为了避免忘记写close()方法，推荐使用with()语句处理文件。语法：

```
with expression as target:
    do something

expression :用于指定一个表达式，这里可以是打开文件的open()函数
target: 用于指定一个变量，并将expression 的结果保存在该变量中。
do something: 可以是对文件的操作等。
```

```
with open('demo.txt', 'w') as file:
    pass
print("文件操作结束")
```

4. 写入文件内容

可以通过write方法向文件写入内容。

```
with open('demo.txt', 'w') as file:
    file.write("abcdefghijklmn") # 将"abcdefghijklmn"写入文件，并覆盖掉原来的内容
print("文件操作结束")
```

5. 读取文件

○ 读取指定字符

文件对象提供了read()方法读取指定个数的字符。语法：

```
file.read([size])
```

demo2.py

```
# 读取文件中的指定个数的字符
with open("demo.txt", 'r') as file:
    string = file.read(3) # 读取前3个字符
    print("前3个字符为:", string) # 运行结果: 前3个字符为: abc
```

read()方法是从文件开头开始数获取多少字符的。如果向获取中间的内容, 可以使用seek()方法。

file.seek(offset[, whence])

file:操作的文件对象

offset:用于指定移动的字符个数, 其具体位置和whence有关

whence: 用于指定从什么位置开始结算。值为0时表示从开头开始计算, 值为1时表示从当前位置开始结束, 值为2时表示从文件末尾开始计算。默认为0。

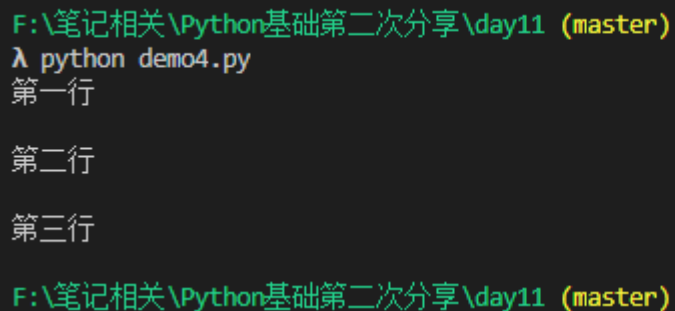
```
'''demo3.py'''
with open("demo.txt", 'r') as file:
    file.seek(3) # 将光标移动到新位置
    string = file.read(8)
    print(string) # 运行结果: defghijk
```

o 读取一行

file.readline()

demo4.py

```
with open("msg.txt", 'r', encoding='utf-8') as file:
    while True:
        line = file.readline()
        if line == '':
            break
        print(line)
```



```
F:\笔记相关\Python基础第二次分享\day11 (master)
λ python demo4.py
第一行

第二行

第三行

F:\笔记相关\Python基础第二次分享\day11 (master)
```

o 读取全部行

file.readlines()

demo5.py

```
with open("msg.txt", 'r', encoding='utf-8') as file:
    msg = file.readlines()
    print(msg)
```

```
F:\笔记相关\Python基础第二次分享\day11 (master)
λ python demo5.py
['第一行\n', '第二行\n', '第三行']

F:\笔记相关\Python基础第二次分享\day11 (master)
λ
```

可以看出返回值为一个列表。

目录操作

1. os和os.path模块

在python中内置了os模块，以及它的子模块os.path,用于对目录或文件进行操作。

```
import os
```

```
import os

# name:用于获取操作系统的类型。 nt:windows posix,则表示为linux,Unix或者mac
print(os.name) # nt

# linesep: 用于获取当前操作系统的换行符
print(str(os.linesep)) # windows正常应该返回'\r\n',我这里由于某种原因没有显示出来

# sep用于获取当期系统的路径分割符
print(os.sep)
```

2. 路径

◦ 相对路径

```
# 获取当前路径
print(os.getcwd()) # F:\笔记相关\Python基础第二次分享\day11
```

相对路径就是在当前路径的基础上直接输入文件加的路径，比如获取子文件的内容

```
with open("子文件夹/msg.txt") as file:
    pass
```

◦ 绝对路径

在Python中通过abspath()来获取绝对路径

```
os.path.abspath("路径") # 来获取指定的路径的绝对路径
```

◦ 拼接路径

```
os.path.join(path1)
```

注意这个函数拼接路径时，不会对文件路径的存在与否进行检查。

3. 判断目录是否存在

```
os.path.exists(path)
```

如果存在则返回True，否则返回False

```
# 检查路径是否存在
myPath = os.path.join(r"C:\user\code", 'text.txt') # 进行路径拼接
print(os.path.exists(myPath)) # 返回值: False
```

4. 创建目录 删除目录

◦ 创建一级目录

```
os.mkdir("文件夹路径")
```

如果文件夹存在会报错。

◦ 创建多级目录

```
os.makedirs("文件夹路径")
```

当文件夹路径中的某个文件夹不存在时，也会自动被创建

◦ 删除目录

```
os.rmdir("要删除的文件夹路径")
```

如果文件夹不为空则会报错

rmdir智能删除空文件夹，如果文件夹有内容可以用Python的标准模块"shutil"的rmtree函数实现

```
import shutil
shutil.rmtree("文件夹路径")
```

5. 遍历目录

```
os.walk()
```

该函数可以遍历指定文件夹下的目录

```
# 遍历文件夹
dirs = os.walk(r"F:\笔记相关\Python基础第二次分享\day11")
for mydir in dirs:
    print(mydir)
```

```
('F:\\笔记相关\\Python基础第二次分享\\day11', [], ['demo.py', 'demo.txt', 'demo1.py', 'demo2.py', 'demo3.py', 'demo4.py', 'demo5.py', 'demo6.py', 'msg.txt', 'pic.jpg', 'Python基础day11 文件与IO.md'])
```

高级文件操作

1. 删除文件

```
os.remove("path")
```

```
os.remove("test.txt") # 该文件位于同级目录下
```

2. 重命名文件和目录

```
os.rename("源文件", "新文件")
```

```
# 重命名
src = r'F:\笔记相关\Python基础第二次分享\day11\old.txt'
newName = r'F:\笔记相关\Python基础第二次分享\day11\new.txt'
if os.path.exists(src): # 判断文件是否存在
    os.rename(src, newName)
    print("重命名完成")
else:
    print("文件不存在")
```

3. 获取文件基本信息

```
os.stat(path)
```

用来获取文件的基本信息，可以通过如下属性来获取

属性	说明	属性	说明
st_mode	保护模式	st_dev	设备名
st_ino	索引号	st_uid	用户ID
st_nlink	硬链接号（被连接数目）	st_gid	组ID
st_size	文件大小，单位为字节	st_atime	最后一次访问时间
st_mtime	最后一次修改时间	st_ctime	最后一次状态变化时间

```
# 文件信息
if os.path.exists("msg.txt"):
    info = os.stat("msg.txt")
    print("文件大小:", info.st_size, "字节")
    print("最后一次修改时间:", info.st_mtime)
```

```
文件大小: 31 字节
最后一次修改时间: 1547888349.578518
```