文件读写

1 引入

- 什么是文件

  文件是保存在持久化存储设备(硬盘、U盘、光盘..)上的一段数据，一个文本、一个py文件、一张图片、一段视频音频等这些都是文件。

- 文件分类
  - 文本文件：打开后会自动解码为字符，如txt文件、word文件、py程序文件。
  - 二进制文件：内部编码为二进制码，无法通过文字编码解析，如压缩包、音频、视频、图片等。
- 字节串类型
  - 概念：在python3中引入了字节串的概念，与str不同，字节串以字节序列值表达数据，更方便用来处理二进程数据。
  - 字符串与字节串相互转化方法
    - 普通的英文字符字符串常量可以在前面加b转换为字节串，例如：b'hello'
    - 变量或者包含非英文字符的字符串转换为字节串方法：str.encode()
    - 字节串转换为字符串方法：bytes.decode()

    注意：python字符串用来表达utf8字符，因为并不是所有二进制内容都可以转化为utf8字符，所以不是所有字节串都能转化为字符串，但是所有字符串都能转化成二进制，所以所有字符串都能转换为字节串。

2 文件读写操作

使用程序操作文件，无外乎对文件进行读或者写

- 读 ： 即从文件中获取内容

- 写：即修改文件中的内容

对文件进行读写的基本操作步骤为：打开文件，读写文件，关闭文件。

2.1 打开文件

```
file_object = open(file, mode='r', buffering=-1,
encoding=None)
功能：打开一个文件，返回一个文件对象。
参数： file  文件名
     mode  打开文件的方式，如果不写默认为‘r’
     buffering  1表示有行缓冲，默认则表示使用系统默认提供的缓冲机制
     encoding='UTF-8'  设置打开文件的编码方式，一般Linux下不需要
返回值：成功返回文件操作对象
```

打开模式	效果
r	以读方式打开，文件必须存在
w	以写方式打开，文件不存在则创建，存在清空原有内容
a	以追加模式打开，文件不存在则创建，存在则继续进行写操作
r+	以读写模式打开，文件必须存在
w+	以读写模式打开文件，不存在则创建，存在清空原有内容
a+	追加并可读模式，文件不存在则创建，存在则继续进行写操作
rb	以二进制读模式打开，同r
wb	以二进制写模式打开，同w
ab	以二进制追加模式打开，同a
rb+	以二进制读写模式打开，同r+
wb+	以二进制读写模式打开，同w+
ab+	以二进制读写模式打开，同a+

> 注意：
>
> 1. 以二进制方式打开文件，读取内容为字节串，写入也需要写入字节串
> 2. 无论什么文件都可以使用二进制方式打开，但二进制文件不能以文本方式打开，否则后续读写会报错

文件打开代码示例：open_file.py

```
# 读方式打开文件
# file = open("3.txt","r")

# 写方式打开文件
# file = open("file.txt","w") # 清除原来内容
file = open("file.txt","a") # 不会清除原来内容

# 操作文件

# 关闭
file.close()
```

2.2 读取文件

- 方法1

```
read(size=-1)
功能： 读取文件中字符
参数： 默认值为-1，此时文件将被读取直至末尾，给定size表示最多读取给定个字符（字节）
返回值： 返回读取到的内容
```

> 注意：1. 文件过大时不建议直接读取到文件结尾，占用内存较多，效率较低
>
> 2. 读到文件结尾，如果继续进行读操作，会返回空字符串

- 方法2

readline(size=-1)
功能： 读取文件中一行
参数： 默认值为-1,表示读取一行，给定size表示最多读取指定的字符（字节）或到换行位置
返回值： 返回读取到的内容

- 方法3

readlines(hint=-1)
功能： 读取文件中的每一行作为列表中的一项
参数： 默认值为-1，文件将被读取直至末尾，给定hint表示读取到hint个字符所在行为止
返回值： 返回读取到的内容列表

- 方法4

```
# 文件对象本身也是一个可迭代对象，在for循环中可以迭代文件的每一行
for line in fr:
     print(line)
```

文件读操作代码示例：read_file.py

```
# 打开文件
# file = open("file.txt","r")

file = open("file.txt","rb")

# 读取内容
data = file.read()
print(data.decode())

# 每次读取一个字符,将文件内容原样打印出来
# while True:
#     data = file.read(1)
#     if data == "":
#         break
#     print(data,end="")

# 按行读取
# data = file.readline()
```

```
# print(data)

# 读取所有行内容
# data_list = file.readlines()
# print(data_list) # 内容列表

# 迭代每次获取一行
# for line in file:
#     print(line)

file.close()
```

随堂练习：基于 dict.txt 完成
编写一个函数,参数是一个单词,查询这个单词的解释
提示 ： 单词有可能查不到，显示None
     字符串切片split()
思路 ： 使用 word 逐行比对

```
def find_word(word):
    file = open("dict.txt", "r")
    # 每次读取一行
    for line in file:
        tmp = line.split(' ')[0] # 提取单词
        # 如果遍历的单词大于word就不用再找了
        if tmp > word:
            file.close()
            return
        elif word == tmp:
            file.close()
            return line

print(find_word("abc"))
```

2.3 写入文件

- 方法1

```
write(data)
功能：把文本数据或二进制数据块的字符串写入到文件中去
参数：要写入的内容
返回值：写入的字符个数
```

> 注意： 如果需要换行要自己在写入内容中添加\n

- 方法2

```
writelines(str_list)
功能：接受一个字符串列表作为参数，将它们写入文件
参数：要写入的内容列表
```

文件写操作代码示例：

```
# 写打开
file = open("file.txt","w")
# file = open("file.txt","a") # 追加

# 写操作
# n = file.write("hello,死鬼\n".encode())
# file.write("哎呀,干啥\n".encode())
# print("写入字符数:",n)

# 将列表每一项写入到文件
data = [
    "接着奏乐\n",
    "接着舞\n"
]

file.writelines(data)

file.close()
```

```
随堂练习：编写一个函数,参数传入一个指定的文件,
将这个文件复制到程序运行目录下,注意不确定文件类型
思路 ： 边读边写

def copy(filename):
    """
    将指定文件复制到这个文件夹下
    :param filename: 指定的要复制的文件
    """
    new_file = filename.split('/')[-1] # 取出文件名
    fr = open(filename,"rb") # 原文件
    fw = open(new_file,"wb") # 新文件
    # 边读边写
    while True:
        data = fr.read(1024)
        if not data:
            break
        fw.write(data)
    fr.close()
    fw.close()


copy("/home/robin/b.jpg")
```

2.4 关闭文件

打开一个文件后，就可以通过文件对象对文件进行操作了，操作结束后需要关闭文件

- 方法

```
file_object.close()
```

- 好处

1. 可以销毁对象节省资源（如果不关闭程序结束后对象也会被销毁）。
2. 防止后面对这个对象的误操作。

2.5 with操作

python中的with语句可以用于访问文件，在语句块结束后会自动释放资源。

- with语句格式

```
with context_expression [as obj]:
    with-body
```

- with访问文件

```
with open('file','r+') as fr:
    fr.read()
```

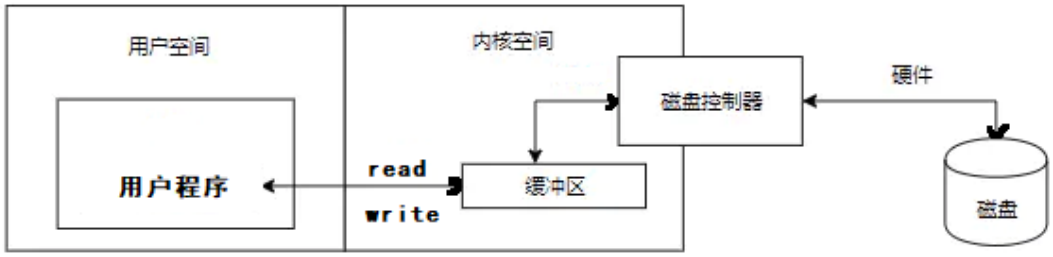
> 注意： with语句块结束后会自动释放文件，所以不需要再close()

```
with 使用示例：

# 临时打开文件简单使用 file = open()
with open("file.txt") as file:
    data = file.read()
    print(data)

# 语句块结束 file 被销毁
```

2.6 读写缓冲区

- 定义

  系统自动的在内存中为每一个正在使用的文件开辟一个空间，在对文件读写时都是先将文件内容加载到缓冲区，再进行读写。



- 作用
    1. 减少和磁盘的交互次数，保护磁盘。
    2. 提高了对文件的读写效率。
- 缓冲区设置

类型	设置方法	注意事项
系统自定义	buffering=-1	
行缓冲	buffering=1	当遇到\n时会刷新缓冲
指定缓冲区大小	buffering>1	必须以二进制方式打开

- 刷新缓冲区条件

1. 缓冲区被写满
2. 程序执行结束或者文件对象被关闭
3. 程序中调用flush()函数

```
file_obj.flush()
```

文件读写缓冲示例：buffer.py

```
# buffering=1 行缓冲
# file = open("file.txt","w",buffering=1)

# buffering > 1 指定缓冲大小
file = open("file.txt","wb",buffering=10)
```

```
while True:
    data = input(">>")
    if not data:
        break
    file.write(data.encode())
    # file.flush() # 刷新缓冲

file.close()
```

2.7 文件偏移量

- 定义

  打开一个文件进行操作时，系统会自动生成一个记录，记录每次读写操作时所处的文件位置，每次文件的读写操作都是从这个位置开始进行的。

  > 注意：
  >
  > 1. r或者w方式打开，文件偏移量在文件开始位置
  > 2. a方式打开，文件偏移量在文件结尾位置

- 文件偏移量控制

  ```
  tell()
  功能：获取文件偏移量大小
  返回值：文件偏移量
  ```

  ```
  seek(offset[,whence])
  功能： 移动文件偏移量位置
  参数：offset代表相对于某个位置移动的字节数。负数表示向前移动，正数表示向后移动。
      whence是基准位置，默认值为 0，代表从文件开头算起，1代表从当前位置算起，2代表从文件末尾算起。
  ```

  > 注意：必须以二进制方式打开文件时，基准位置才能是1或者2

```
"""
```

```
随堂练习
编写一个程序，启动后循环向文件 my.log 中写入
如下内容：
1. Tue Aug  2 14:38:57 2022
2. Tue Aug  2 14:39:00 2022
3. Tue Aug  2 14:39:03 2022
4. Tue Aug  2 14:39:06 2022
5. Tue Aug  2 14:39:09 2022

要求每条内容占一行，每行写入后可以实时在文件中查看
每隔3秒写入一行，当程序终止后重新启动能够继续向下写，并且序号衔接
提示：  获取当前时间 time.ctime()
      时间间隔：  time.sleep(3)
"""
import time

# 保证每行及时刷新
f = open("my.log","a+",encoding="UTF-8",buffering=1)

f.seek(0,0) # 偏移量到开头才能读取
# n = 行数 + 1
n = len(f.readlines()) + 1

while True:
    line = f'{n}. {time.ctime()}\n'
    f.write(line)
    n += 1
    time.sleep(3) # 时间间隔0
```

3 os模块

os模块是Python标准库模块，包含了大量的文件处理函数。

- 获取文件大小

```
os.path.getsize(file)
功能：  获取文件大小
参数：  指定文件
返回值：  文件大小
```

- 查看文件列表

```
os.listdir(dir)
功能： 查看文件列表
参数： 指定目录
返回值：目录中的文件名列表
```

- 判断文件是否存在

```
os.path.exists(file)
功能： 判断文件是否存在
参数： 指定文件
返回值： 布尔值
```

```
os模块使用示例：file.py

import os

print("文件大小:",os.path.getsize("file.txt"))
print("文件列表:",os.listdir("."))
print("文件是否存在:",os.path.exists("file.txt"))
```