文件处理

1.1 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…… 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，字节串以不可变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。
一个汉字3个字节。
```

1.2 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

对文件实现读写的基本操作步骤为：打开文件，读写文件，关闭文件。

1.2.1 打开文件

```
file_object = open(file_name, access_mode='r', buffering=-1)
功能：打开一个文件，返回一个文件对象。
参数：file_name   文件名；
     access_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。
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，但是二进制文件则不能以文本方式打开，否则后续读写会报错。

1.2.2 读取文件

- 方法1

```
read([size])
功能： 来直接读取文件中字符。
参数： 如果没有给定size参数（默认值为-1）或者size值为负，文件将被读取直至末尾，给定size最多读取给定数目个字符（字节）。
返回值： 返回读取到的内容 ---字符串
```

> 注意：文件过大时候不建议直接读取到文件结尾，读到文件结尾会返回空字符串或空字节串。

```python
# 大文件循环读取文件
while True:
    data = f.read(100)  # 每次读取100个字节，读到文件结尾后会返回空字符串
    if not data:
        break
    print(data)
```

- 方法2

```
readline([size])
功能： 用来读取文件中一行
参数： 如果没有给定size参数（默认值为-1）或者size值为负，表示读取一行，给定size表示最多读取制定的字符（字节）。
返回值： 返回读取到的内容 ---字符串
```

- 方法3

```
readlines([sizeint])
功能： 读取文件中的每一行作为列表中的一项
参数： 如果没有给定size参数（默认值为-1）或者size值为负，文件将被读取直至末尾，给定size表示读取到size字符所在行为止。
返回值： 返回读取到的内容列表 ---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 f:
    print(line)
```

1.2.3 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
```

> 读-写：4-2

1.2.4 关闭文件

打开一个文件后我们就可以通过文件对象对文件进行操作了，当操作结束后可以关闭文件操作

- 方法

```
file_object.close()
```

- 好处

1. 可以销毁对象节省资源，（当然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。

1.2.5 with操作

python中的with语句也可以用于访问文件，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

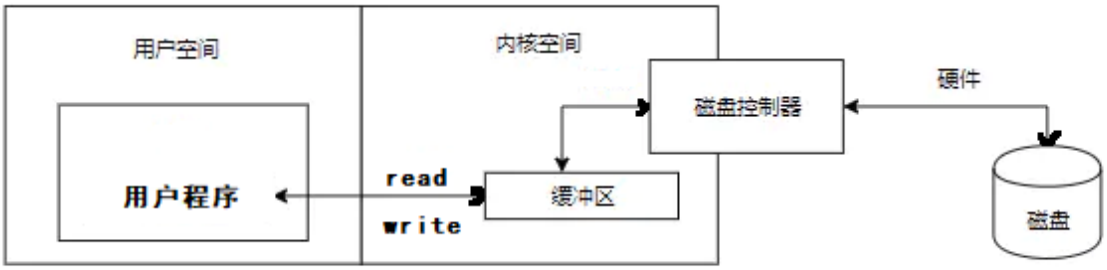
```
with open('file','r+') as f:
    f.read()
```

> 注意： with语句块结束后会自动释放f所以不再需要close().

1.2.6 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()
```

1.2.7 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是基准位置的默认值为 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

```
"""
重点代码
练习4：  文件的拷贝
        有一个文件  timg.jpeg,将其拷贝一份到主目录下，命名为 mm.jpg

        从源文件读取，写入到目标文件

思路 ：  从timg.jpeg中读取内容，原样写入到mm.jpg
"""

fr = open('timg.jpeg','rb')  # 源文件
fw = open('/home/tarena/mm.jpg','wb') # 新文件

while True:
    data = fr.read(1024)
    # 读到文件结尾返回空字符串
    if not data:
        break
    fw.write(data)
```

```
20
21 fr.close()
22 fw.close()
23
```