文件： 计算机中由操作系统管理的具有名字的存储区域

Python内置的open函数会创建一个Python文件对象 可以作为计算机上的一个文件链接

操作
f = open(‘123.log’, ‘w’) 创建一个文件 w 指的是写入 此时调用f.read()报错
f = open(‘123.log’, ‘w+’) #打开一个文件 如果文件不存在则创建 可同时读写

f = open(‘123.log’, ‘r’) #只读方式打开文件
f = open(‘123.log’) 默认就是r
f= open(‘123.log’, ‘a’) #代表向文件追加内容
f= open(‘123.log’, ‘a+’) #对文件进行读写

注意写和追加的区别

常用方法

In [2]: f.

f.close f.flush f.next f.seek f.writelines
f.closed f.isatty f.read f.softspace f.xreadlines
f.encoding f.mode f.readinto f.tell
f.errors f.name f.readline f.truncate
f.fileno f.newlines f.readlines f.write

1. Read() ##把整个文件读进到一个字符串

In [4]: f.read()
Out[4]: 'sfs\nsdf\nsdf\nff\nsd\nsa\na\n\n'

In [6]: f.seek(0) ###跳转文件文件内容的操作光标到指定位置

In [7]: f.read(2)
Out[7]: 'sf'

In [10]: f.tell() ##获取当前文件操作光标的位置
Out[10]: 2

astring = f.readline() #读取下一行
astring = f.readlines() #读取整个文件到字符串列表 当处理大文件时慎用

flush() ##刷新缓冲区

方法是用来刷新缓冲区的，即将缓冲区中的数据立刻写入文件，同时清空缓冲区，不需要是被动的等待输出缓冲区写入。

一般情况下，文件关闭后会自动刷新缓冲区，但有时你需要在关闭前刷新它，这时就可以使用 flush() 方法。

f.write(string) #写入字符串到文件
f.writelines(alist) #把列表内的所有字符串写入文件
f.close() #关闭文件 ##对文件操作完成后， 需要关闭文件
f.closed ##判断文件是否已经关闭

In [30]: f.closed
Out[30]: False

f.flush() #刷新但不关闭文件

for j in open('123.log') #使用迭代器读文件 最快的读文件的方式

Python程序中的文本文件都采用的是字符串的形式 读取文件时返回字符串形式的文本

注意：

1. 文件迭代器是最好的读取行的工具
2. 内容是字符串而不是对象 从文本读取的数据回到脚本时是一个字符串 如果字符串不是程序员所需的对象类型， 我们可以将其进行转化
3. close是通常的选项 调用close()方法会终止对文件的链接
4. 默认情况下输出文件总是缓冲的 这意味着写入文本可能不会立即自动从内存转换到硬盘 可以通过关闭文件或者flush方法迫使缓存的数据进入硬盘

如果想要一行一行的扫描文件 最好的方式是使用文件迭代器

```
for j in open('myfile'):
    print j
```

以这种方式进行编码 open临时创建的文件对象将自动在每次循环迭代的时候读入并返回一行 这种形式通常很容易编写
占用内存更少