 input 可以接收一个Python表达式作为输入，并将运算结果返回

str = input("请输入：")

print "你输入的内容是: ", str

请输入：[x\*5 for x in range(2,10,2)]

你输入的内容是: [10, 20, 30, 40]

判断文件是否存在os.path.exists(filename)

先用Python内置的open()函数打开一个文件，创建一个file对象，相关的方法才可以调用它进行读写。

file object = open(file\_name [, access\_mode][, buffering])

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。

b 二进制模式，一般用于分文本文件如图片

w 打开一个文件只用于写入。如果该文件已存在则打开文件，并从开头开始编辑，即原有内容会被删除。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

## File对象的属性

file.closed 返回true如果文件已被关闭，否则返回false。

file.mode 返回被打开文件的访问模式

file.name 返回文件的名称

### **close()方法**

### **write()方法**

write()方法不会在字符串的结尾添加换行符('\n')

### **read()方法**

fileObject.read([count]）

被传递的参数是要从已打开文件中读取的字节计数。该方法从文件的开头开始读入，如果没有传入count，它会尝试尽可能多地读取更多的内容，很可能是直到文件的末尾

**重命名和删除文件**

Python的os模块提供了帮你执行文件处理操作的方法，比如重命名和删除文件。

要使用这个模块，你必须先导入它，然后才可以调用相关的各种功能。

rename()方法：

rename()方法需要两个参数，当前的文件名和新文件名。

os.rename(current\_file\_name, new\_file\_name)

你可以用remove()方法删除文件，需要提供要删除的文件名作为参数。

os.remove(file\_name)

**python里的目录**

os模块有许多方法能帮你创建，删除和更改目录

可以使用os模块的mkdir()方法在当前目录下创建新的目录们。你需要提供一个包含了要创建的目录名称的参数

os.mkdir("newdir")

可以用chdir()方法来改变当前的目录。chdir()方法需要的一个参数是你想设成当前目录的目录名称

os.chdir("newdir")

由于文件读写时都有可能产生IOError，一旦出错，后面的f.close()就不会调用。所以，为了保证无论是否出错都能正确地关闭文件，我们可以使用try ... finally来实现：

try:

f = open('/path/to/file', 'r')

print(f.read())

finally:

if f:

f.close()

但是每次都这么写实在太繁琐，所以，Python引入了with语句来自动帮我们调用close()方法：

with open('/path/to/file', 'r') as f:

print(f.read())

with open('/Users/michael/test.txt', 'w') as f:

f.write('Hello, world!')

**要写入特定编码的文本文件，请给open()函数传入encoding参数，将字符串自动转换成指定编码**

**遇到有些编码不规范的文件，你可能会遇到UnicodeDecodeError，因为在文本文件中可能夹杂了一些非法编码的字符。遇到这种情况，open()函数还接收一个errors参数，表示如果遇到编码错误后如何处理。最简单的方式是直接忽略：**

f = open('/Users/michael/gbk.txt', 'r', encoding='gbk', errors='ignore')

前面讲的默认都是读取文本文件，并且是UTF-8编码的文本文件。要读取二进制文件，比如图片、视频等等，用'rb'模式打开文件即可：

f = open('/Users/michael/test.jpg', 'rb')