## 1. 打印到屏幕

最简单的输出方法是用print语句，你可以给它传递零个或多个用逗号隔开的表达式。

此函数把你传递的表达式转换成一个字符串表达式，并将结果写到标准输出。

print "雪凤小仙女！！！"

你的标准屏幕上会产生以下结果：

# 雪凤小仙女！！！

## 2. 读取键盘输入

Python提供了两个内置函数从标准输入读入一行文本，默认的标准输入是键盘。

### 2.1. aw\_input函数

raw\_input([prompt]) 函数从标准输入读取一个行，并返回一个字符串（去掉结尾的换行符）：

str = raw\_input("请输入：")  
  
print "你输入的内容是: ", str

这将提示你输入任意字符串，然后在屏幕上显示相同的字符串。当我输入"雪凤小仙女!!!"，它的输出如下：

请输入：雪凤小仙女!!!  
你输入的内容是: 雪凤小仙女!!!

### 2.2. input函数

**input([prompt])** 函数和 **raw\_input([prompt])** 函数基本类似，但是 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

这会产生如下的对应着输入的结果：

请输入：[x\*5 for x in range(2,10,2)]  
你输入的内容是: [10, 20, 30, 40]

## 3. 打开和关闭文件

Python 提供了必要的函数和方法进行默认情况下的文件基本操作。你可以用 **file** 对象做大部分的文件操作。

### 3.1. open 函数

你必须先用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。

语法：

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

各个参数的细节如下：

* file\_name：file\_name变量是一个包含了你要访问的文件名称的字符串值。
* access\_mode：access\_mode决定了打开文件的模式：只读，写入，追加等。所有可取值见如下的完全列表。这个参数是非强制的，默认文件访问模式为只读(r)。
* buffering: 如果buffering的值被设为0，就不会有寄存。如果buffering的值取1，访问文件时会寄存行。如果将buffering的值设为大于1的整数，表明了这就是的寄存区的缓冲大小。如果取负值，寄存区的缓冲大小则为系统默认。

### 3.2. File对象的属性

一个文件被打开后，你有一个file对象，你可以得到有关该文件的各种信息。

| 属性 | 描述 |
| --- | --- |
| 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。 |
| file.softspace | 如果用print输出后，必须跟一个空格符，则返回false。否则返回true。 |

# 打开一个文件  
fo = open("foo.txt", "w")  
  
print "文件名: ", fo.name # 文件名: foo.txt  
print "是否已关闭 : ", fo.closed # 是否已关闭 : False  
print "访问模式 : ", fo.mode # 访问模式 : w  
print "末尾是否强制加空格 : ", fo.softspace # 末尾是否强制加空格 : 0

### 3.3. close()方法

File 对象的 close（）方法刷新缓冲区里任何还没写入的信息，并关闭该文件，这之后便不能再进行写入。

当一个文件对象的引用被重新指定给另一个文件时，Python 会关闭之前的文件。用 close（）方法关闭文件是一个很好的习惯。

语法：

fileObject.close()

# 打开一个文件  
fo = open("foo.txt", "w")  
print "文件名: ", fo.name # 文件名: foo.txt  
   
# 关闭打开的文件  
fo.close()

### 3.4. write()方法

write()方法可将任何字符串写入一个打开的文件。需要重点注意的是，Python字符串可以是二进制数据，而不是仅仅是文字。

write()方法不会在字符串的结尾添加换行符('\n')：

语法：

fileObject.write(string)

# 打开一个文件  
fo = open("foo.txt", "w")  
fo.write( "wuQinHao like zhengXueFeng!!!\n")  
   
# 关闭打开的文件  
fo.close()

上述方法会创建foo.txt文件，并将收到的内容写入该文件，并最终关闭文件。如果你打开这个文件，将看到以下内容:

wuQinHao like zhengXueFeng!!!

### 3.5. read()方法

read（）方法从一个打开的文件中读取一个字符串。需要重点注意的是，Python字符串可以是二进制数据，而不是仅仅是文字。

语法：

fileObject.read([count])

在这里，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。

# 打开一个文件  
fo = open("foo.txt", "r+")  
str = fo.read(13)  
print "读取的字符串是 : ", str # 读取的字符串是 : wuQinHao like  
  
# 关闭打开的文件  
fo.close()

## 4. 文件定位

tell()方法告诉你文件内的当前位置, 换句话说，下一次的读写会发生在文件开头这么多字节之后。

seek（offset [,from]）方法改变当前文件的位置。Offset变量表示要移动的字节数。From变量指定开始移动字节的参考位置。

# 打开一个文件  
fo = open("foo.txt", "r+")  
str = fo.read(13)  
print "读取的字符串是 : ", str # 读取的字符串是 : wuQinHao like  
   
# 查找当前位置  
position = fo.tell()  
print "当前文件位置 : ", position # 当前文件位置 : 13  
   
# 把指针再次重新定位到文件开头  
position = fo.seek(0, 0)  
str = fo.read(13)  
print "重新读取字符串 : ", str # 重新读取字符串 : wuQinHao like  
  
# 关闭打开的文件  
fo.close()

## 5. 重命名和删除文件

### 5.1. rename() 方法

rename() 方法需要两个参数，当前的文件名和新文件名。

语法：

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

import os  
   
# 重命名文件test1.txt到test2.txt。  
os.rename( "test1.txt", "test2.txt" )

### 5.2. remove()方法

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。

语法：

import os  
  
os.remove(file\_name)

import os  
   
# 删除一个已经存在的文件test2.txt  
os.remove("test2.txt")

## 6. Python里的目录

所有文件都包含在各个不同的目录下，不过Python也能轻松处理。os模块有许多方法能帮你创建，删除和更改目录。

### 6.1. mkdir()方法

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。

语法：

import os  
  
os.mkdir("newdir")

import os  
   
# 将在当前目录下创建一个新目录test  
os.mkdir("test")

### 6.2. chdir()方法

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

语法：

import os  
  
os.chdir("newdir")

import os  
  
# 将当前目录改为"/home/newdir"  
os.chdir("/home/newdir")

### 6.3. getcwd()方法

getcwd()方法显示当前的工作目录

语法：

import os  
  
os.getcwd()

import os  
   
# 给出当前的目录  
print os.getcwd()

### 6.4. rmdir()方法

rmdir()方法删除目录，目录名称以参数传递。在删除这个目录之前，它的所有内容应该先被清除。

语法：

import os  
  
os.rmdir('dirname')

以下是删除" /tmp/test"目录的例子。目录的完全合规的名称必须被给出，否则会在当前目录下搜索该目录。

import os  
   
# 删除”/tmp/test”目录  
os.rmdir( "/tmp/test" )

## 7. 文件、目录相关的方法

File 对象和 OS 对象提供了很多文件与目录的操作方法，可以通过点击下面链接查看详情：

* [File 对象方法](https://www.runoob.com/python/file-methods.html): file 对象提供了操作文件的一系列方法。
* [OS 对象方法](https://www.runoob.com/python/os-file-methods.html): 提供了处理文件及目录的一系列方法。