最简单的输出：

input 可接受表达式作为输入，并返回表达式的值

\_\_repr\_\_ 解释器的读取

\_\_str\_\_ print输出的适合人阅读形式

直接在shell中输出调用的是\_\_repr\_\_的方法，使用print输出是调用了\_\_str\_\_

函数str()用于将值转化为适于人阅读的形式，而repr()转化 为供解释器读取的形式。

打开文件open()函数基本语法：

一、open（）打开

fileObj = open(file\_name , mode , encoding)

file\_name： 文件名(当前目录下的文件名，可写绝对路径的文件名)

mode：文件的打开模式

encoding用来设置编码 utf-8

write( )写 close( )关闭

二、使用with 可以自动调用close()方法

with open(filename,mode,encoding) as f: #把filename直接赋值给f重命名

Print f.read()

打开文件的文件名

相对路径和绝对路径

打开当前目录下的文件可直接写文件名

打开其他路径下的文件则使用绝对路径

例：

>>> f1 = open('text') #text 文件与Python 3.0在同一目录下(相对路径)

>>> f2 = open(r'C:\Users\Win8\Desktop\m.py') #打开桌面的m.py文件(绝对路径)

<\_io.TextIOWrapper name='C:\\Users\\Win8\\Desktop\\m.py' mode='r' encoding='cp936'>

>>> f2.name #访问路径

'C:\\Users\\Win8\\Desktop\\m.py'

>>> f2.mode #访问属性

'r'

文件名是以字符串参数的形式存在的，所以在前面加r字符，防止字符串的转义而导致文件打开失败。打开不存在文件会报错。

打开模式：

r 只读模式，文件不存在时会报错、

w 写入模式，文件存在会清空之前的内容，文件不存在则会新建文件

a 追加写入模式，不清空之前的文件，直接将写入的内容添加到后面。

b 以二进制模式读写文件，rb,wb,ab,

+ 可读写模式（可添加到其他模式中使用）

r+,w+,a+,这几种模式还遵循了r,w,a的基本原则。

总结区别：

r只读，r+读写 ，不创建

w新建只写，w+新建读写，二者会将原来的内容清零

r+ 可读写，若文件不存在报错，w+可读可写，若文件不存在，创建

r+ 是覆盖写，a+是追加写

不可读打开方式：w，a

若不存在会新建文件的打开方式：a , a+ , w , w+

>>> f1 =open(r'C:\Users\Win8\Desktop\xx.py','w',encoding='utf8') #新建xx.py文件

>>> f1.write('aaaaaa') #写入aaaaa字符

>>> f1.close() #关闭保存，用flush刷新也可以。

文件对象的属性和方法：

f.open() 打开

f.flush() 刷新

f.close() 关闭文件

f.closed 查看文件是否关闭

f.read(size) 读取文件的所有字符，size是可选的数值，读取字符的个数

f.readline() 从文件中读取并返回一行，也可设置读取字符的个数

f.readlines() 以列表形式返回，每行一个元素，用for循环迭代输出每一行的信息。

f.write(str) 将字符串str写入文件

f.writelines(seq) 向文件写入字符串序列seq，seq是返回字符串的可迭代对象

f.tell() 返回当前在文件中的位置

f.seek(offset,from) 在文件中移动文件指针，从from(0)代表起始位置（默认），1代表当前位置，2代表文件末尾)偏移offset个字节

>>> f2 =open(r'C:\Users\Win8\Desktop\python习题（自己）.py')

>>> f2.tell() #返回当前在文件中的位置

0

>>> f2.read(12) #读取文件前12个字符

'def fun1(a):'

>>> f2.tell()

12

>>> print (f2.read()) #读取文件的所有字符

a=list(a)

a.reverse()

return ''.join(a)

>>> f2.tell() #返回当前在文件中的位置

69

>>> f2.seek(0) #把指针移动至起始位置

0

>>> f2.tell() #返回当前在文件中的位置

0

>>> f2.readline() #从文件中读取并返回一行

'def fun1(a):\n'

>>> f2.readline() #从文件中读取并返回一行

' a=list(a)\n'

>>> f2.readlines() #以列表形式返回

[' a.reverse()\n', " return ''.join(a)\n"]

文件对象方法

1.**读取文件内容**：read()，readline()，readlines()

f.read(size) 读取文件的内容，将文件的内容以字符串形式返回。

size是可选的数值，指定字符串长度，如果没有指定size或者指定为负数，就会读取并返回整个文件。

当文件大小为当前机器内存两倍时就会产生问题，反之就尽可能大的size读取和返回数据，如果到了文件末尾，会返回空字符串。

f.readline() 从文件中读取单独一行，字符串结尾会自动加上一个换行符\n，只有当文件最后遗憾没有以换行符结尾时，这一操作才会被忽略，这样返回值就不会有混淆。

如果返回空字符串，表示到达率文件末尾，如果是空行，就会描述为\n,一个只有换行符的字符串。

f.readlines() 返回一个列表，列表的元素为文件行的内容。可以通过列表索引的方式将文件的每一行的内容输出。可以通过for循环迭代输出每一行的信息。

1. **文件的写入**：write() , writelines()

f.write() 将要写入的内容以字符串的形式通过write方法写入文件中。

f.writelines() 括号里必须是由字符串元素组成的序列：f.writelines(())或f.writelines([])。

1. **内容的保存和关闭：**flush() , close()

f.flush()在读写模式下，当写完的数据想要读取出来时，要先缓存区的内容保存到文件当中。

f.close() 关闭文件。对一个已经关闭的文件进行操作会报错

1. **光标位置：**tell() , seek()

f.tell() 返回光标在文件中的位置

f.seek(offset,from)在文件中移动文件指针，从from(0代表起始位置，1代表当前位置，2代表文件末尾)偏移offset个字节。

1. **查看文件信息**：closed , mode , name

closed 查看文件是否已经关闭，返回布尔值

mode 返回文件打开模式

name 返回文件名

Python3和python2区别案例：

m.py文件：

try:

a = input('请输入一个数字类型的数：')

b = input('请输入一个字符串型的数：')

c = a+b

>>> f2 =open(r'C:\Users\Win8\Desktop\m.py')

>>> f2.read(4) #因为m.py中有中文所以报错

Traceback (most recent call last):

File "<pyshell#67>", line 1, in <module>

f2.read(4)

UnicodeDecodeError: 'gbk' codec can't decode byte 0xaa in position 50: illegal multibyte sequence

>>> f2 =open(r'C:\Users\Win8\Desktop\m.py',encoding='utf8')

>>> f2.read(4) #加上’utf8’不会报错

'try:'

#python2如果不设置编码是不支持中文的，python3是utf-8的编码所以不需要管中文的问题，但这里用的是cp936，你的程序是用的utf-8，所以要改成一样的才有用

python3里的这个open函数和2的不一样，3中open有个encoding用来设置编码，

这个encoding 的默认值是cp936

utf-8是国际编码