[【python】一个文件内容写入另一个](https://www.cnblogs.com/bashaowei/p/8761024.html)

with open('f1.txt','r') as f1:

with open('f2.txt','w') as f2:

f2.write(f1.read())

# [Python从文件中读取数据](https://www.cnblogs.com/tizer/p/11032642.html)

**一、读取整个文件内容**

在读取文件之前，我们先创建一个文本文件resource.txt作为源文件。

1 with open('resource.txt') as file\_obj:

2 content = file\_obj.read()

3 print(content)

4

**二、逐行读取文件内容**

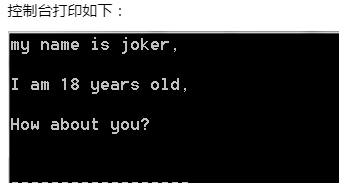
1 file\_name = 'resource.txt'

2

3 with open(file\_name) as file\_obj:

4 for content in file\_obj:

5 print(content)



解释：在上面的程序中，因为Python在读取文件之后将其存入对象file\_obj 中，我们通过对该对象进行循环来遍历文件中的每一行，但是却发现，多了空白行，因为在这个文件中，有看不见的换行符，且print语句语句也会加上一个换行符，因此每行的末尾会有两个换行符。要消除多于的空白行可在print语句中调用rstrip() 方法，如下：

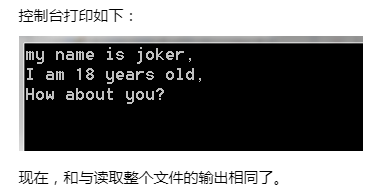
1 file\_name = 'resource.txt'

2

3 with open(file\_name) as file\_obj:

4 for content in file\_obj:

5 print(content.rstrip())



import json

import glob

result = []

for f in glob.glob("\*.json"):

with open(f, "rb") as infile:

result.append(json.load(infile))

with open("merged\_file.json", "wb") as outfile:

json.dump(result, outfile)

# Python从json中提取数据

s = json.loads('{"name":"test", "type":{"name":"seq", "parameter":["1", "2"]}}')

print s

print s.keys()

print s["name"]

print s["type"]["name"]

print s["type"]["parameter"][1]

通常我们比较容易想到如下的方式：

with open('foriginal,txt','r') as source:

with open('aim.txt','w') as destination:

destination.write(source.read())

然而，with支持多参数，于是有了更优雅的写法：

with open('foriginal,txt','r') as source,open('aim.txt','w') as destination:

destination.write(source.read())

Python2.x load文件内容时，会出现顺序乱

最后采用有序字典集合:collections.OrderedDict ,二者输出保持一致

dictStr = json.loads(jsonstr,object\_pairs\_hook=collections.OrderedDict)

参见：<https://www.jb51.net/article/181445.htm>

还有中文写入会乱码

              import sys  
              reload(sys)  
              sys.setdefaultencoding('utf8')

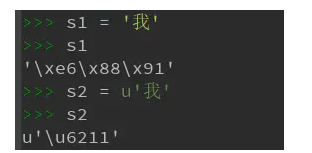
参见：<https://www.cnblogs.com/xiao-xue-di/p/11283496.html>

<https://www.jb51.net/article/138179.htm>

# python2.7 处理中文

## unicode 与 utf-8

默认的中文使用的是utf-8编码，前面加u的是Unicode编码：



写文件的字符串不能用Unicode编码，可以用utf-8编码，同理，其它的IO也应该使用utf-8：

# utf-8编码下可以正常写文件

s = '等等'

with open('testData.txt', 'w') as fOut:

fOut.write(s)

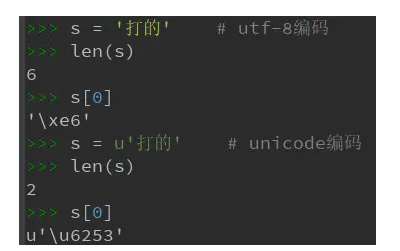
# unicode编码时写文件会抛出编码异常

s = u'等等'

with open('testData.txt', 'w') as fOut:

fOut.write(s)

unicode编码是python的工作编码，utf-8编码是python的IO编码。  
即字符串在内存中编辑时应该使用Unicode编码，字符串写入文件前应该转换成utf-8编码。文件中读取出的字符串是utf-8编码，对这字符串进行编辑前应该转换成Unicode编码。  
为什么编辑时要用Unicode编码而不直接用utf-8编码？且看下图：



如上，utf-8编码时字符串中的每个元素是字节，unicode 编码时字符串中的元素才是一个汉字。（这也是为什么要用utf-8写文件而不用unicode）

## encode 与 decode

unicode和utf-8的转换需要使用encode和decode，这里主动忽略gb编码。  
encode是将当前的字符串（字节串）编码成目标编码的字节串，decode是将当前的字节串解码为Unicode编码的字符串。所以才说Unicode是工作编码嘛。  
使用示例：

s = u'打的' # unicode字符串

s1 = s.encode('utf-8') # 将Unicode字符串编码成utf-8字节串

s2 = s1.decode('utf-8') # 将utf-8字节串解码成unicode字符串

s = '打的' # utf-8字节串

s2 = s.decode('utf-8') # 将utf-8字节串解码成Unicode字符串

s1 = s.encode('unicode escape') # 将utf-8字节串编码成Unicode字节串

s2 = s1.decode('utf-8') # 将Unicode字节串解码成Unicode字符串

用法分析：



一般来说，读文件出来的字符串是utf-8字符，想要对此进行处理，建议先将其decode成Unicode字符串。  
写文件时要十分注意这个字符串是不是utf-8。不是就得先encode。  
如果嫌麻烦，也可以直接在开头加这几句：

import sys

reload(sys)

sys.setdefaultencoding('utf-8')

## JSON 的 load 与 dump

用load函数加载json文件时，中文字符会被转换成\u开头的unicode码。

用dump函数写json文件时，需要注意一个参数 ensure\_ascii，此参数指定是否保证文件只含ASCII字符串，默认为True。所以要将其指定为False不然中文就被转码了：

with open("./test.json", "w") as f:

json.dump(elements, f, ensure\_ascii=False)

也可以不直接dump而是先dumps再write：

dataOut = json.dumps(dataOut)

with open('robot\_info\_2.json', 'w') as fOut:

fOut.write(dataOut.decode('unicode escape').encode('utf-8'))

## 案例1：将\u字符串转换成中文

s = '\u6253\u7684'

s1 = s.decode('unicode escape')