# 【整理】Python中实际上已经得到了正确的Unicode或某种编码的字符，但是看起来或打印出来却是乱码

2013 年 7 月 19 日 下午 4:26 crifan 已有9683人围观 4个评论



【背景】

Python中的字符编码，其实的确有点复杂。

再加上，不同的开发环境和工具中，显示的逻辑和效果又不太相同，尤其是，中文的，初级用户，最常遇到的：

（1）在Python自带的IDE：IDLE中折腾中文字符，结果看到的差不多都是乱码类的东西，比如：'\xd6\xd0\xce\xc4'

（2）将一个中文字符，打印输出到windows的cmd命令行中，看到的是乱码

对此，此处专门整理一下，这些常见的现象，和现象背后的根本原因，以及如何解决这类问题。

## 背景知识

其实，看下面问题之前，最好是已经了解相关的背景知识，才更容易看懂的：

1.字符编码的基本知识

对于字符编码本身，比如UTF-8，GBK等等，不熟悉的，不了解是啥的话，先去看：

字符编码详解

2.Windows的cmd中的默认是GBK编码

这方面不了解的，也需要先去看：

Windows的命令行工具: cmd

中的：

设置字符编码：简体中文GBK/英文

3.关于IDLE

其实也要先大概了解：

Python内部，默认的字符编码是，是根据操作系统，我们多数都是Windows的中文系统，默认是GBK编码。

而IDLE中，直接输入中文字符，其实就是GBK编码的。

4.Python中的字符串的设计

主要是：Python 2.x中的str和unicode ，和，Python 3.x中的bytes和str，之间的逻辑，转换，和区别。

不了解的，也要先去看：

【整理】Python中字符编码的总结和对比：Python 2.x的str和unicode vs Python 3.x的bytes和str

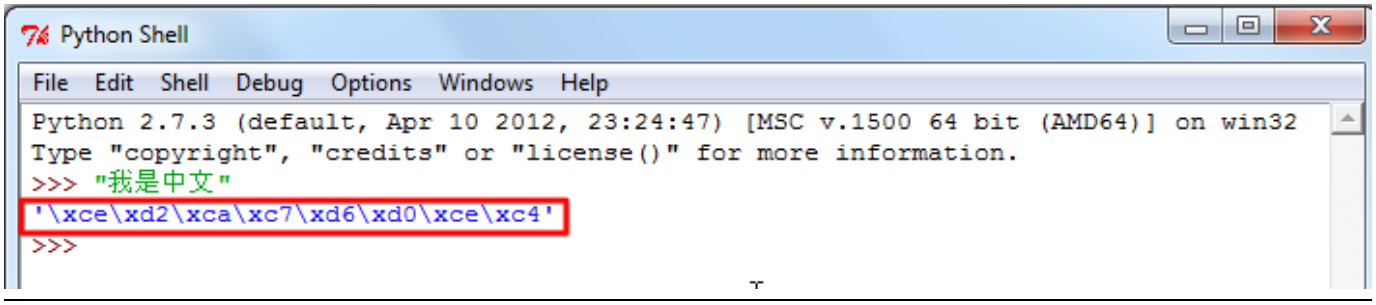
# 常见问题：IDLE中看到类似于'\xd6\xd0\xce\xc4'，而不是我想要的中文字符

初学者，最容易遇到的问题就是：

中文用户，用了Python自带的IDLE，在里面输入中文后，结果显示出，类似于：

```
'\xce\xd2\xca\xc7\xd6\xd0\xce\xc4'
```

的内容，而不是希望看到输出的中文字符，比如：




此现象的解释是：

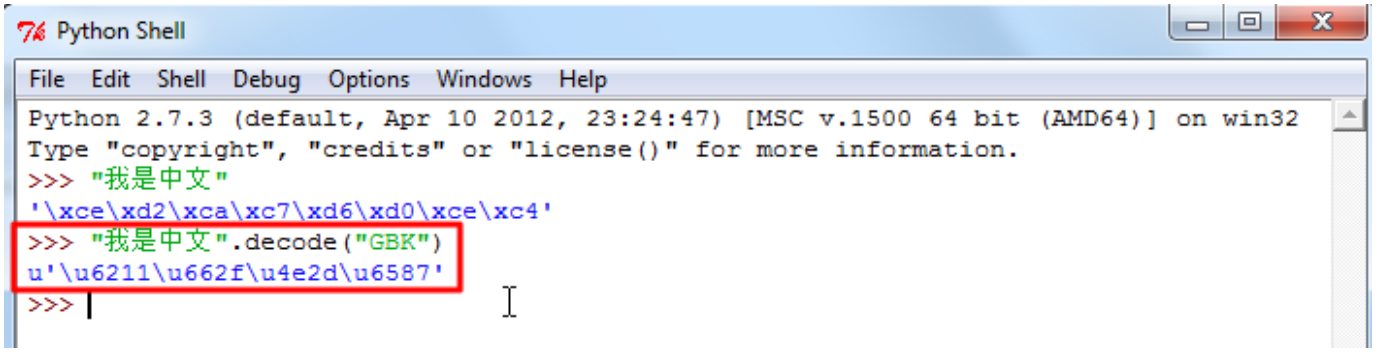
实际上，此处你，本身就已经得到了，正确的，默认的GBK编码的，中文字符串："我是中文"

了。只是：

IDLE这个，Python自带的IDE，不是很好用的IDE，给你显示出来，其内部的16进制的值而已。

1. 对于此点，你可以去用decode去验证一下：

```
Python 2.7.3 (default, Apr 10 2012, 23:24:47) [MSC v.1500 64 bit (AMD64)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> "我是中文"
'\xce\xd2\xca\xc7\xd6\xd0\xce\xc4'
>>> "我是中文".decode("GBK")
u'\u6211\u662f\u4e2d\u6587'
>>>
```




其中，GBK的字符串，经过解码后，就可以得到Unicode的字符串了，对应的显示出来的是：

```
u'\u6211\u662f\u4e2d\u6587'
```

此处的：

\u6211，\u662f，\u4e2d，\u6587，分别对应着，四个中文字符："我"，"是"，"中"，"文"

2. 有人会问，我怎么知道这些值，是对应着这四个中文字符的呢？

答案是：

那是因为你不熟悉Unicode。且也不会去查Unicode表格。

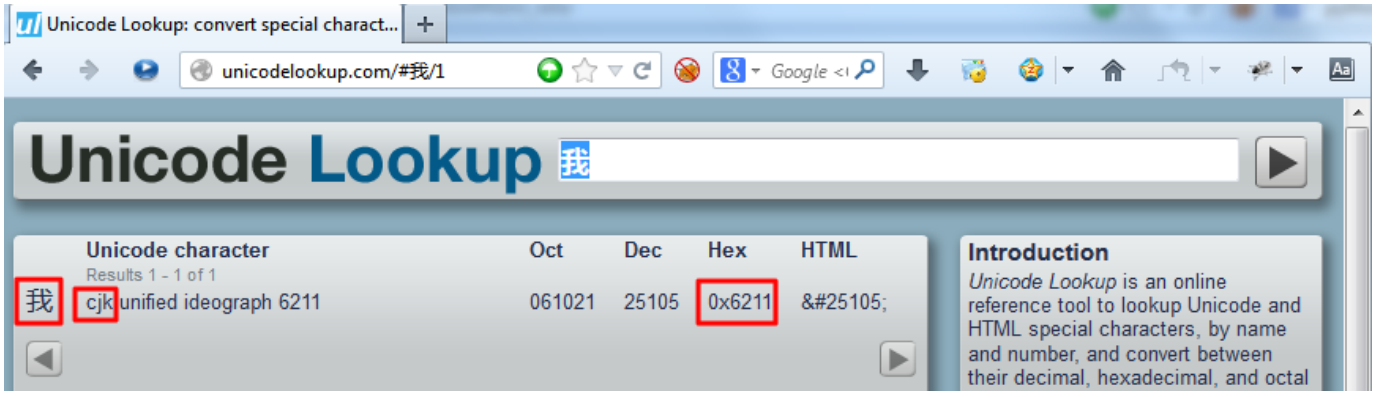
等你看了之前告诉你的：

字符编码详解

然后再去参考我的：

HTML相关的参考资料

去查Unicode值，就可以查到“我”对应的Unicode值是0x6211：




---

同理，可以查得剩下的：

0x662f="是"=\u662f

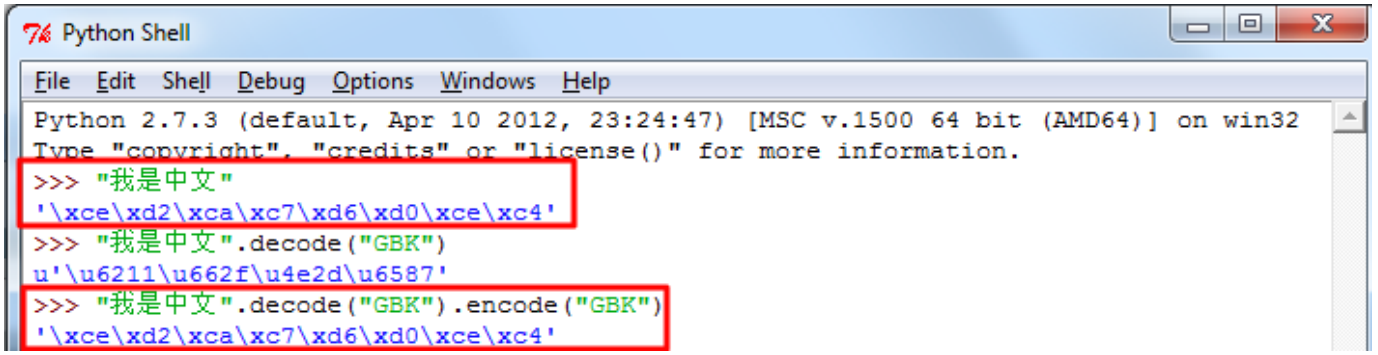
0x4e2d="中"=\u4e2d

0x6587="文"=\u6587

3.回到上面的问题，接着，还可以接着进一步验证，之前的字符串，的确是GBK：

```
Python 2.7.3 (default, Apr 10 2012, 23:24:47) [MSC v.1500 64 bit (AMD64)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> "我是中文"
'\xce\xd2\xca\xc7\xd6\xd0\xce\xc4'
```

```
>>> "我是中文".decode("GBK")
u'\u6211\u662f\u4e2d\u6587'
>>> "我是中文".decode("GBK").encode("GBK")
'\xce\xd2\xca\xc7\xd6\xd0\xce\xc4'
```



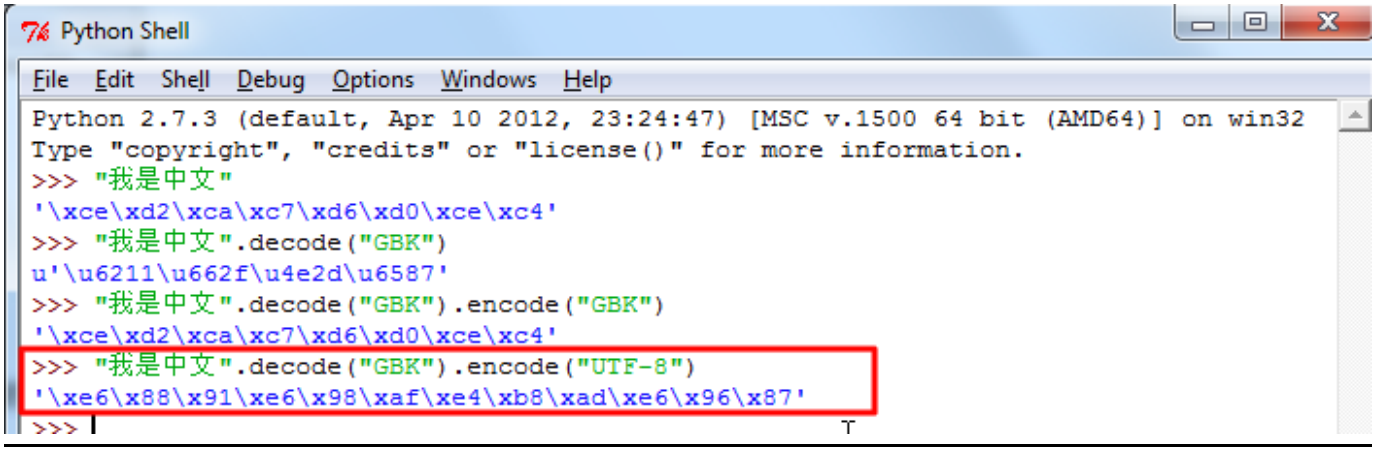

即：

之前直接输入中文字符所得到的16进制值，和通过GBK解码后得到Unicode，然后再编码为GBK的16进制的值，是一样的

-> 说明之前的中文字符的确是GBK的编码。

4.另外，也可以顺带看看，UTF-8的输出是啥：

```
Python 2.7.3 (default, Apr 10 2012, 23:24:47) [MSC v.1500 64 bit (AMD64)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> "我是中文"
'\xce\xd2\xca\xc7\xd6\xd0\xce\xc4'
>>> "我是中文".decode("GBK")
u'\u6211\u662f\u4e2d\u6587'
>>> "我是中文".decode("GBK").encode("GBK")
'\xce\xd2\xca\xc7\xd6\xd0\xce\xc4'
>>> "我是中文".decode("GBK").encode("UTF-8")
'\xe6\x88\x91\xe6\x98\xaf\xe4\xb8\xad\xe6\x96\x87'
```




所以，总结此问题：

IDLE中输入中文字符，但是显示出来的是类似于'\xd6\xd0\xce\xc4'的值，而不是想要的中文字符

的答案就是：

其实本身已经是中文字符。

只是根据当前默认是GBK编码，所显示出来的GBK编码的内部的值而已。

其实，对此问题，更加终极的解决办法是：

由于IDLE不是很好用，所以不推荐用户，尤其是初学者，直接就用IDLE来开发Python。

而是推荐你用：

Notepad++ 加 cmd

具体的原因和解释，详见：

【整理】【多图详解】如何在Windows下开发Python：在cmd下运行Python脚本，如何使用Python Shell（command line模式和GUI模式），如何使用Python IDE

更更终极的办法是：

这类常见的错误，属于学习Python中所容易走的弯路。

而你要是按照我的教程去学习，不仅可以少走很多弯路，而且更容易明白很多基本的逻辑：

初级的：

python初级教程：入门详解

中级的：

python中级教程：开发总结

高级专题阐述：

Python专题教程：字符串和字符编码

Python专题教程：抓取网站，模拟登陆，抓取动态网页

# 常见问题：中文字符打印输出显示到命令行（Windows的cmd）显示乱码

和上面的现象类似的一个现象就是：

当用python代码，打印输出一个中文字符到命令中，结果却显示乱码。

（1）用如下代码：

```
#!/usr/bin/python
# -*- coding: utf-8 -*-
"""
--------------------------------------------------------------------------
[Function]
【整理】Python中实际上已经得到了正确的Unicode或某种编码的字符，但是看起来或
http://www.crifan.com/python_already_got_correct_encoding_string_but_

[Date]
2013-07-19

[Author]
Crifan Li

[Contact]
http://www.crifan.com/about/me/
--------------------------------------------------------------------------
"""

#----------------------------------import------------------------------

#--------------------------------------------------------------------------
def char_ok_but_show_messy():
    """
        Demo Python already got normal chinese char, with some encodi
    """
    #此处，当前Python文件是UTF-8编码的，所以如下的字符串，是UTf-8编码的
    cnUtf8Char = "我是UTF-8的中文字符串";
```

```
29      #所以，将UTF-8编码的字符串，打印输出到GBK编码的命令行（Windows的cmd）
30      print  "cnUtf8Char=",cnUtf8Char; #cnUtf8Char= 鎴戞槸UTF-8鐨勪腑鏂囧
31      #如果想要正确显示出中文字符，不显示乱码的话，则有两种选择：
32      #1. 把字符串转换为Unicode编码，则输出到GBK的命令行时，Python会自动将U
33      decodedUnicodeChar = cnUtf8Char.decode("UTF-8");
34      print "decodedUnicodeChar=",decodedUnicodeChar; #decodedUnicodeCh
35      #2. 让字符串的编码和输入目标（windows的cmd）的编码一致：把当前的字符串
36      reEncodedToGbkChar = decodedUnicodeChar.encode("GBK");
37      print "reEncodedToGbkChar=",reEncodedToGbkChar; #reEncodedToGbkCh
38
39
40  ##############################################################################
41  if __name__=="__main__":
42      char_ok_but_show_messy();
```

注意：

此时Python的文件编码是UTF-8。

不了解的，详见：

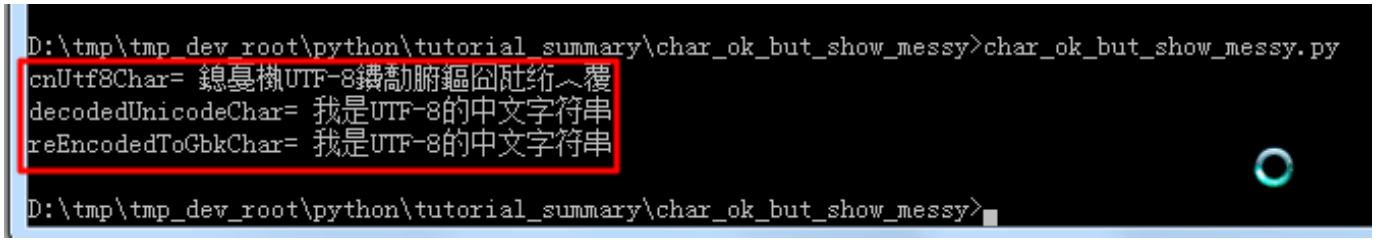
**【整理】Python中用encoding声明的文件编码和文件的实际编码之间的关系**

（2）当前代码下载（右键另存为）：

char_ok_but_show_messy.py

（3）还原现象

运行的结果是：




（4）解释

代码中已经解释的很清楚了。

不再啰嗦。

# 相关帖子

和此类的，python的字符串编码方面的相关内容，之前有更多的总结：

【总结】Python 2.x中常见字符编码和解码方面的错误及其解决办法

【整理】关于Python 3.x 中自动识别字符串编码，并正确在cmd中输出的各种情况的测试