

cur.execute("SET NAMES 'gbk'")

temp = "SELECT loginname,salt,userid FROM user WHERE username = '%s'" % (teachername\_post)

sql = temp.decode('utf8').encode('gb18030')

这个是数据库

teachername\_post = i.teachername.encode('utf8') 这个是webpy的

# 五分钟战胜 Python 字符编码

2016/06/27 · [基础知识](http://python.jobbole.com/category/basic/) · [编码](http://python.jobbole.com/tag/%e7%bc%96%e7%a0%81/)

分享到：**3**

本文作者： [伯乐在线](http://python.jobbole.com/) - [LittleCoder](http://www.jobbole.com/members/wjLcGhKwuCH) 。未经作者许可，禁止转载！  
欢迎加入伯乐在线 [专栏作者](http://blog.jobbole.com/99322" \t "http://python.jobbole.com/85482/_blank)。

对于很多接触Python的人而言，字符的处理和语言整体的温顺可靠相比显得格外桀骜不驯难以驾驭。

本文不谈复杂的理论，就经验教你字符处理八字真言：确定编码，同类交互。

文章针对Python 2.7，主要因为3对的编码已经有了很大的改善并且实际原理一样，更改一下操作命令即可。

了解完本文，你可以轻松解决文字处理，特殊平台（Windows?）下的编码，爬虫编码等问题。

## **阅读建议**

本文分为如下几个部分：

* 原理
* 具体操作
* 建议的使用习惯
* 疑难问题解答

如果想要了解我给出的使用习惯，可以直接跳到建议的使用习惯。

如果只想要解决相关问题可以直接跳到疑难问题解答。

希望本文能够帮到你。

## **原理**

为了理解方便，这里不谈理论只做类比，具体想要进一步了解各种编码的理论的搜狗一下好了。

首先说一下我们为什么会碰到各式各样的编码问题：

* 因为我们没有统一编码
* 因为我们没有用对命令（传对数据）

再说一下编码是什么，Python的编码看似复杂，实际上可以看做只有两类编码：Unicode，二进制

* Unicode 相信都很熟悉：，就是\u0000这样的
* 二进制编码也很简单，就是\x00\x00这样的，平常看到的utf-8,cp936都是二进制编码
* 二进制编码是具象的，10001100原样就可以存储，而Unicode是抽象的，不能这样存

Python

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11 | *#coding=utf8*    *# Unicode编码演示*  print('Unicode:')  print(repr('Unicode编码'))`    *# 二进制编码演示*  print(u'二进制编码:')  print(repr('Unicode编码'))`    *# 只是看个样子，代码不必去深究* |

再说怎么做，就是只有同种编码之间才可以操作

* 举个简单的类比

Python

|  |  |
| --- | --- |
| 1  2  3  4 | 就把一串数据比为烤鸭，我们作为人和鸭子不同种看待烤鸭的态度完全不一样。  我们看到的是晚上的配菜，鸭子看到的是自己二舅。  那么我在逛烤鸭店的时候用错编码就会报错。  因为我在烤鸭店看到了满世界的二舅。 |

* 这里说的同种就是我们熟悉的各种编码方式：utf-8,unicode,ucs-bom
* 这也就是编码问题的核心，非常重要。

最后说一下Python的环境

* 本身代码是用Ascii解码的，文件里有Ascii无法解码的内容的话要告知Python怎么解码
* 内部大量命令都是默认接受Unicode

Python

|  |  |
| --- | --- |
| 1  2  3 | *# 告知的命令就是下面这一行，删掉就会报错*  *#coding=utf8*  print(u'测试编码') |

## **具体操作**

拿到各种编码的内容自然是不用说，那么如果我们想要自己构造怎么做呢，看下面：

Python

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9 | *#coding=utf8*    *# 字符串前面加u会默认构造出Unicode的字符串*  unicodeString = u'Unicode字符串'    *# 字符串前面什么都不加会构造出默认编码（首行限定了现在的utf8）的字符串*  utf8String = 'Utf-8字符串'    *# 当然，没有首行，默认的编码是Ascii* |

那么他们之间怎么转换呢，同样很简单：

Python

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10 | *# 接上一段程序*    *# Unicode转化为二进制编码中的一种：utf8*  unicodeString.encode('utf8')    *# 二进制编码根据自己的编码种类转化为Unicode*  utf8String.decode('utf8')    *# 如果二进制编码中混进了奇怪的东西可以根据需求用特殊的decode策略*  print(repr('u8字\x00符串'.decode('utf8', 'replace'))) |

那么怎么样会出现问题呢：

Python

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11 | *# 接上一段程序*    *# 如果我们把他们转化成同样的编码方式就可以操作（例如相加）*  print(repr(unicodeString + utf8String.decode('utf8')))  print(repr(unicodeString.encode('utf8') + utf8String))    *# 但如果不转化，当然就会出现满世界的烤鸭二舅啦*  unicodeString + utf8String    *# 所以另一方面也发现，编码转换是需要我们告诉程序怎么做的*  *# 所有`decode`操作都会生成Unicode编码，这是为了方便我之前说的大量接受Unicode的内部命令* |

所以我们需要确定程序使用的编码，这是我们需要告诉程序的东西

* 一方面在操作字符串的时候确定是同种编码
* 另一方面在使用非自己写的命令时，一般使用Unicode，或者使用接收二进制编码的命令

Python

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11 | *#coding=utf8*  *# 这里拿写入文件举例*    *# 一般使用Unicode*  **with** open('Unicode.txt', 'w') **as** f: f.write(u'Unicode测试')    *# 或者使用接收二进制编码的命令*  **with** open('Utf8.txt', 'wb') **as** f: f.write('Utf8测试')    *# 你可以反过来做个测试，自然会报错*  *# 二进制的命令方便了在不知道怎么解码的情况下也能进行操作（写入文件）* |

## **我建议的使用习惯**

相信到这里我已经把我对于编码的理解讲完了。

我们为什么会碰到各式各样的编码问题：

* 因为我们没有统一编码
* 因为我们没有用对命令（传对数据）

所以这里再重申一下八字真言：确定编码，同类交互

* 碰到问题，问一下自己，我现在是哪种编码
* 同一种编码才能交互，那我应该是哪种编码

这里给出我的使用习惯：

* 确定一种内部编码
* 内部编码的选择优先级如下：程序必须使用的编码、第三方包使用的编码、你喜欢的编码、Unicode
* 在输出时再更改到特定的编码

记得在开始整个程序之前确定内部的编码，否则编码一团糟会产生很多不必要的bug。

不要迷信内部Unicode，例如Evernote开发就应该根据第三方包使用的Utf8确定内部编码。

## **疑难问题解答**

### **编码识别**

说了要确定编码，那么拿到一串二进制要怎么确定编码呢？

最简单的方法是chardet：（需要安装）

Python

|  |  |
| --- | --- |
| 1 | python -m pip install chardet |

使用非常简单：

Python

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | *#coding=utf8*    **from** chardet **import** detect  print(detect('这是一串utf8的测试字符'))    *# 结果：`{'confidence': 0.99, 'encoding': 'utf-8'}`* |

另外例如抓取网站，那么头文件中很有可能有提示如何解码，记得不要忘记了。

### **编码转换**

很可能因为字符串中参杂了奇怪的东西，导致即使编码种类正确，依旧无法解码。

我知道我之前讲过了，但可能有人直接跳疑难问题解答嘛。

这里可以使用decode的第二个参数：

Python

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8 | *#coding=utf8*    *# 字符串中混进了\x00*  rubbishUtf8String = 'Utf-8字\x00符串'    print(repr(rubbishUtf8String.decode('utf8', 'replace')))    print(repr(rubbishUtf8String.decode('utf8', 'ignore'))) |

### **特殊平台下编码**

很多人都说Windows是个坑，即使在Python 3下面也一样。

因为中文文件名出来都是乱码。

这里使用一个取巧的方法：平台编码再特殊，起码命令行读取和创建一个文件夹不会出乱码吧。

Python

|  |  |
| --- | --- |
| 1  2  3  4 | **import** sys, os    **for** folder **in** os.walk('.').next()[1]:      print(folder.decode(sys.stdin.encoding)) |

同样的输入输出也可以这样做优化：

Python

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | **import** sys    **def** sys\_print(msg):      print(msg.encode(sys.stdin.encoding))    **def** sys\_input(msg):  **return** raw\_input(msg.encode(sys.stdin.encoding)).decode(sys.stdin.encoding) |

### **文件写入**

如果抓下来一个内容不知道怎么解码，但还是想要写入文件怎么办

写入文件的时候制定用二进制命令即可：

Python

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9 | *#coding=utf8*  **import** urllib    **with** open('Utf8.txt', 'wb') **as** f: f.write('Utf8测试')    *# 比如抓了个网页，不知道编码也可以写入文件进行一系列操作*    content = urllib.urlopen('http://www.baidu.com').read()  **with** open('baidu.txt', 'wb') **as** f: f.write(content) |

### **裸Unicode字符**

Unicode存成六个Ascii字符怎么办？其实也可以decode

Python

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15 | *#coding=utf8*  *# 这是普通的Unicode*  s = u'测'  **for** i **in** s: print(i)  print(repr(s))    *# 这是裸Unicode，实际存成了六个Ascii*  s = repr(s)[2:-1]  **for** i **in** s: print(i)  print(repr(s))    *# 转化其实也很简单*  s = s.decode('unicode-escape')  **for** i **in** s: print(i)  print(repr(s)) |