【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

2012 年 11 月 29 日 下午 6:38 crifan 已有17362人围观 8个评论

Python 2.x中的字符编码，设计的的确不好，导致初学者，甚至是即使用Python很长时间的人，都会经常遇到字符编解码方面的错误。

下面就把一些常见情，尽量的都整理出来，并给出相应的解决办法。

看此文之前

Python中字符编码所涉及的背后逻辑（从你输入字符，到终端显示字符的背后过程）

在去了解Python编码解码之前，还有个更加重要，但是很多时候却被其他解释相关知识的人所忽略的问题，那就是：

对于Python中字符串，输入输出的背后逻辑。

即，知其所以然。

此处就简单介绍一下，在Python中，从你所输入的字符串，到显示出字符串，这背后的过程是什么样的。

只有了解了这个大概的过程，和背后的逻辑，你才能真正理解后面的所解释的，字符串编码解码方面的错误，以及如何解决这样的错误。

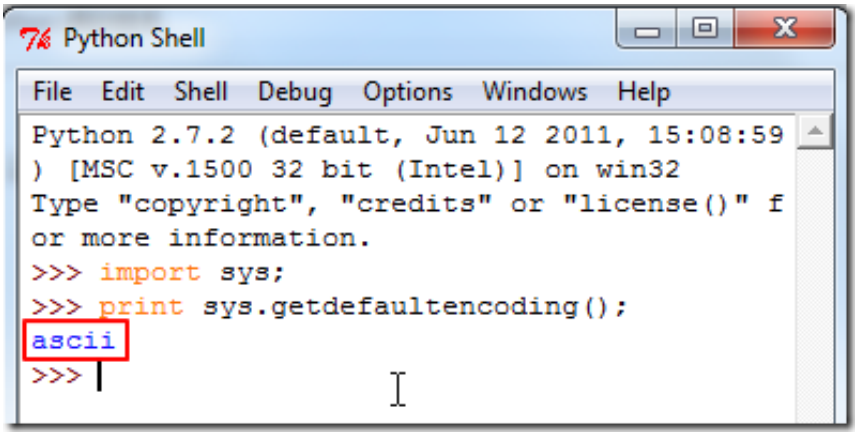
对于你

你只是看到的是：

- 你输入了字符串
  - 不论是从Python的IDLE中输入的
  - 还是写入到Python文件中的
- 然后你去运行的对应代码，该Python代码，经过Python系统，（此处及之后，我称其为**Python**解析器），的处理
- 使得你可以看到最终所输出的字符串
  - 不论是在Python的IDLE中看到的
  - 还是在windows的cmd中看到的。

对于**Python**解析器

而Python解析器所干的事情，就是：

1. Python解析器，根据当前的所用的字符串编码类型
   - 此字符串编码类型，是你自己所设置的
     - 不论是在Python的IDLE中，还是Python文件中
     - 都是你自己显示指定对应的编码类型的
   - 当然你没显示的指定的话，那就用默认的配置
     - 如果是Python的IDLE，如果你没修改defaultencoding，那么就使用默认的字符编码
       - 可以通过sys.getdefaultencoding()而获得，比如此处获得是：**ascii**
       - 

     - 如果是Python文件，如果你没声明文件编码，则使用默认的编码：**UTF-8**
       - 常见的做法是指定为对应的UTF-8类型：# -*- coding: utf-8 -*-
         - 相关内容，不了解的可参考：【整理】关于Python脚本开头两行的：#!/usr/bin/python和# -*- coding: utf-8 -*-的作用 – 指定文件编码类型
2. 去执行你的Python代码
   - 其中，很常见的几种动作是：
     - 打印print对应的所获得的字符

- 对于字符串打印,Python的逻辑:
  - 如果是Unicode字符串,则可以,自动地,编码为对应的终端所用编码,然后正确的显示出来
    - 比如unicode的字符串,输出到windows的默认编码为GBK的cmd中,则Python可以自动将Unicode编码为GBK,然后输出到cmd中
    - 个别特殊情况,也会出错:
      - 当此unicode字符串中包含某特殊字符,而目标终端的编码集合中,没有此字符,则很明显也是无法实现将Unicode编码为对应的特定编码的字符串,无法正确显示的
  - 如果是某种编码类型的str,则需要该str的编码类型,和目标终端编码匹配
    - 比如GBK的字符串,输出到windows的默认编码为GBK的cmd,则是可以正常输出的
      - 此处后来经过代码测试，就发现一个有趣或者说诡异的问题，虽然我们python文件声明的UTF-8编码，但是实际上实际上是用GBK编码，而此时，文件中的字符串，很明显是用GBK存储的，所以，将此GBK字符，输出到GBK的cmd中，是可以正常输出的。即，此处字符串的类型，很明显只和文件所用的实际编码有关，而和文件所声明的代码无关。
    - 如果是UTF-8的字符串,输出到windows的默认编码为GBK的cmd,就会出错
- 对相应的字符，进行编码（为某种特定类型的字符str），或解码（为对应的unicode类型的字符）
  - 比如将当前的某种编码的字符串，解码为Unicode字符串
    - 很明显，也是要保证，你字符串本身的编码和所指定的编码，两者之间要一致的
      - 比如：decodedUnicode = someUtf8Str.decode("UTF-8")
      - 而如果用这样的：decodedUnicode = someGbkStr.decode("UTF-8")，那就会出现错误

所以你要

- 确保当前输入的字符串的编码，和对应的编码设置一致，才能使得解析器正常解析你的字符串
- 确保输出字符串时，所用字符（的编码）和目标输出所用的编码一致，才能正常在输出终端显示
- 确保字符串本身的编码，和你去解码等操作所用的编码一致，否则很明显会由于编码不一致而无法进行解码（等操作）

常见错误简介

很明显，如果你不遵守上述规则，出现前面编码不一致的情况，那么就会出现一些常见的编码解码方面的错误了。

此处只是简单举例如下：

- 你python文件本身是GBK的，对应的字符串也是GBK的，然后你指定按照UTF-8解码为对应的Unicode，那么当然会导致人家Python解析器，按照UTF-8编码的方式，去解析你的，实际上是GBK的字符，当然会出错，无法解析了。
- 你所要打印的字符，本身是UTF-8的，但是要打印输出的终端是，windows的cmd，其默认编码为GBK，即将UTF-8的字符，显示到GBK编码的cmd中，当然也会出现错误了

- 你所要打印出来的字符，虽然是unicode，但是其中包含了某些特殊字符，而对应的特殊字符在终端所用的编码中不存在，比如将含某些特殊字符的Unicode字符串，打印到windows的默认编码为GBK的cmd中，而GBK编码集合中，本身就没有这些字符，所以，当然也是无法显示，会出现对应的UnicodeEncodeError的错误的。

所以，诸如此类的问题，如果搞懂了之前的逻辑，那么自然很容易理解其错误的根源，并找到解决办法。

即，知其所以然，之后，更容易，知其然。

下面就来，故意的：

- 再现，出各种Python 2.x中所常见的字符的编码解码等方面的错误；
- 然后帮你找到问题的根本原因；
- 进而找到问题所对应的解决办法；

Python 2.x的字符编码本身的设计的逻辑：str和unicode

想要了解Python 2.x中，字符串编解码的问题和原因，首先搞懂Python 2.x中在字符串的方面是如何设计的，其主要分两大类：str和unicode。

对于str和unicode的确切含义，以及如何互相转换，以及Python 2.x和Python 3.x中的bytes和str，有何区别等内容

之前已经详尽的总结了：

【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
不了解的，也是需要去看懂，然后才能真正明白下面的问题的原因的。

Python中常见字符的编码和解码方面的错误的现象，原因，及其解决办法

提醒：

1.下面的代码，如果想要拷贝粘贴到文件中去测试的话，请注意文本本身所用编码。

未必一定是声明的那个编码；

详情请自己看代码中的中文说明。

不过最简单的是，直接右键另存为对应的python文件，省去你拷贝粘贴转码等烦心事了。

2. 关于代码编辑器，推荐用Notepad++：

【crifan推荐】轻量级文本编辑器，Notepad最佳替代品：Notepad++

3. 其中对于文件转换编码等事宜，不了解的可以参考：

用Notepad++实现不同字符编码之间的转换

Python中，想要将某字符串解码为对应的Unicode，但是所使用的编码类型和字符串本身的编码不匹配

现象

字符串本身，是某种编码类型的字符串，但是结果将其解码为对应的Unicode时，却指定了另外一种编码类型，导致无法正常的解码为对应的Unicode，而出现UnicodeDecodeError之类的错误。

比如用UTF-8去解码GBK的字符串：

python_2.x_decode_gbk_use_utf8.py

```
#!/usr/bin/python
# -*- coding: utf-8 -*-
"""
--------------------------------------------------------------------------
Function:
【总结】Python中常见字符编码和解码方面的错误及其解决办法
http://www.crifan.com/summary_python_2_x_common_string_encode_decode_

Author:     Crifan
Verison:    2012-11-29
--------------------------------------------------------------------------
"""

def python2xDecodeGbkWithUtf8():
    zhcnActualGbk = "此处你所看到的中文字符所处的环境是：\r\n1.当前文件用
    print "zhcnActualGbk=",zhcnActualGbk; #此处是可以正常打印出上述中文
    zhcnUnicode = zhcnActualGbk.decode("UTF-8"); #UnicodeDecodeError:
    print "zhcnUnicode=",zhcnUnicode; #上述解码出错，更不可能执行到这里
```

```
19
20  ##############################################################################
21  if __name__=="__main__":
22      python2xDecodeGbkWithUtf8();
```

要说明的是，此处是，只是为了代码演示，所以很明显，就直接看出错误了。

但是实际的编程过程中，由于很多时候情况很复杂，未必立刻会意识到或注意到，编码类型弄错了，所以，也还是需要注意这种情况的。

原因

如上所述，想要把某种编码的字符串解码为Unicode；

字符串是A编码的（GBK），但是却用B编码（UTF-8）去解码，导致出现UnicodeDecodeError错误；

解决办法

使用和字符串本身编码相同的编码，去解码，就可以正常的解码为Unicode了。

修改后的代码如下：

python_2.x_decode_gbk_use_gbk.py

```
1   #!/usr/bin/python
2   # -*- coding: utf-8 -*-
3   """
4   ------------------------------------------------------------------------
5   Function:
6   【总结】Python中常见字符编码和解码方面的错误及其解决办法
7   http://www.crifan.com/summary_python_2_x_common_string_encode_decode_
8
9   Author:     Crifan
10  Verison:    2012-11-29
11  ------------------------------------------------------------------------
12  """
13
14  def python2xDecodeGbkWithGbk():
15      zhcnActualGbk = "此处你所看到的中文字符所处的环境是：\r\n1.当前文件用
16      print "zhcnActualGbk=",zhcnActualGbk; #此处是可以正常打印出上述中文
17      zhcnUnicode = zhcnActualGbk.decode("GBK"); #此处使用和字符串本身一
```

```
    print "zhcnUnicode=",zhcnUnicode; #此处就可以正常输出Unicode字符串

###############################################################################
if __name__=="__main__":
    python2xDecodeGbkWithGbk();
```

举一反三

如果你以后遇到类似的，UnicodeDecodeError，那么说明是Unicode在解码方面的错误，肯定是将某种编码的字符串，解码为Unicode的过程中，出现的错误。

而错误原因，也往往都是编码类型不匹配，然后就可以去检查一下，是不是字符串本身的编码，和你在调用decode时所设置的编码不匹配。

Python中，打印字符串时，字符串本身的编码，与输出终端中所用编码不匹配

现象

字符串本身，是某种编码的，但是输出，显示，到终端时，终端所使用编码，和你字符串的编码不一样，导致无法正常显示。

比较常见的是，本身是UTF-8类型的字符串，但是却将其输出到Windows的cmd中，而cmd中默认是GBK编码的，导致两者不匹配，所以打印字符串时，出现乱码：

python_2.x_print_utf8_to_gbk.py

```
#!/usr/bin/python
# -*- coding: utf-8 -*-
"""
-------------------------------------------------------------------------------
Function:
【总结】Python中常见字符编码和解码方面的错误及其解决办法
http://www.crifan.com/summary_python_2_x_common_string_encode_decode_

Author:     Crifan
Verison:    2012-11-29
-------------------------------------------------------------------------------
"""

def python2xPrintUtf8ToGbk():
    zhcnUtf8 = "此处你所看到的中文字符所处的环境是：\r\n1.当前文件用声明的
    print "zhcnUtf8=",zhcnUtf8; #把UTF-8的字符串，输出到GBK的cmd中，结
    
    # zhcnUtf8= 姝ゅ浣犳墍鐪嬪埌鐨勪腑鏂囧瓧绗︽墍澶勭殑鐜鏄細
```

```
    # 1.褰撳墠鏂囦欢鐢ㄥ０鏄庣殑缂栫爜鏄疶F-8鍓備緥鏄悗鏉ョ粡杩囧瘬鏄庤緭
    # 欢鐨勫紡鐮佹湁鍏筹紱
    # 2.涓嶈繃鍛紱杩姝ゅ鏂囦欢鏄韩鐨勫紡鐮佹槸鍏朵欢UTF-8鐨勭紪鐮€涔ヤ綘鐨
    # 3.瀵逛簬UTF-8鐨勫瓧绗︿覆鍜屾垜浠寔鍥句綔鍚堟垚Windows鐨勬粯璁ょ紪鐮佷负

#######################################################################
if __name__=="__main__":
    python2xPrintUtf8ToGbk();
```

原因

把Python文件中的字符串，该Python文件是UTF-8的，所以该字符串也是UTF-8编码的，输出到Windows的cmd中，而cmd中默认编码为GBK，即

把UTF-8的字符串，在GBK的cmd上显示，则出现了乱码。

解决办法

目的是为了是要输出的字符串的编码，是Unicode，或者和目标输出终端的编码一致，就可以正常输出了。

所以可以把UTF-8的字符，解码为对应的Unicode，。

（也可以进一步的，把Unicode字符串，编码为GBK）

然后再输出到GBK的cmd中，就可以正常显示，不是乱码了：

python_2.x_dec_utf8_then_to_gbk.py

```
#!/usr/bin/python
# -*- coding: utf-8 -*-
"""
-----------------------------------------------------------------------
Function:
【总结】Python中常见字符编码和解码方面的错误及其解决办法
http://www.crifan.com/summary_python_2_x_common_string_encode_decode_

Author:      Crifan
Verison:     2012-11-29
-----------------------------------------------------------------------
"""

def python2xDecodeUtf8ThenOutputToGbk():
    zhcnUtf8 = "此处你所看到的中文字符所处的环境是：\r\n1.当前文件用声明的
    print "zhcnUtf8=",zhcnUtf8; #把UTF-8的字符串，输出到GBK的cmd中，结果
    # zhcnUtf8= 姝ゅ浣犳墍鐪嬪埌鐨勪腑鏂囧瓧绗︽墍澶勭殑鐜鏄細 ......

    decodedUnicode = zhcnUtf8.decode("UTF-8"); #用UTF-8解码UTF-8的字符
```

```
    print "decodedUnicode=",decodedUnicode; #此处就可以正常输出了，不会
    decodedUnicodeThenEncodedToGbk = decodedUnicode.encode("GBK"); #
    print "decodedUnicodeThenEncodedToGbk=",decodedUnicodeThenEncoded

##############################################################################
if __name__=="__main__":
    python2xDecodeUtf8ThenOutputToGbk();
```

举一反三

以后如果再遇到类似的乱码，先去确定你的字符串本身是什么编码的。

再去确定，你所要输出的终端目标中，所用的编码是什么样的。

其中，输出的终端，此处举例所用的是，windows的cmd，其他还有可能是输出内容到文件中，其中还涉及文件所用的编码是什么，这些都要自己搞清楚的。

总之，确保字符串编码类型，和输出所用的编码类型，两者是一致的，就不会出现乱码了。

Python中，打印含某些特殊字符的Unicode类型字符串，但是输出终端中字符编码集中不包含这些特殊字符

现象

虽然已经获得了Unicode的字符串了，但是当打印Unicode类型的字符串，到某些终端中时，结果却还是出错了。

比如，下面的例子中，就是把Unicode字符串，打印到Windows的cmd中，结果出错：

python_2.x_print_unicode_still_error.py

```
#!/usr/bin/python
# -*- coding: utf-8 -*-
"""
------------------------------------------------------------------------
Function:
【总结】Python中常见字符编码和解码方面的错误及其解决办法
http://www.crifan.com/summary_python_2_x_common_string_encode_decode_

Author:     Crifan
Verison:    2012-11-29
------------------------------------------------------------------------
"""

```

```
#任何字符，都可以在：
#http://unicodelookup.com/
#中，查找到对应的unicode的值

def python2xPrintUnicodeStillError():
    #http://zhidao.baidu.com/question/500133781.html
    slashUStr = "\\u3232\\u6674"; #(有) 晴
    decodedUniChars = slashUStr.decode("unicode-escape"); #此处已经可
    unicodeButContainSpecialChar = decodedUniChars;
    print "unicodeButContainSpecialChar=",unicodeButContainSpecialCha

    #此处在GBK编码的cmd中输出的话，会出现错误的：
    #UnicodeEncodeError: 'gbk' codec can't encode character u'\u3232'
    #那是因为，Unicode字符：0x3232，是个特殊字符，而此字符，在GBK编码字符

##############################################################################
if __name__=="__main__":
    python2xPrintUnicodeStillError();
```

原因

上述过程中，虽然已经获得了正确的unicode字符串了，但是由于此unicode字符串中包含了一个特殊字符，即那个\u3232，对应的字符，显示出来，像是左右括号中间一个"有"字，即类似于这样的：

(有)

而此特殊字符，GBK字符集中没有，不存在，所以无法将对应的Unicode字符，编码为对应的GBK字符，所以出现UnicodeEncodeError，更无法打印出来

注：

对于任何字符，都可以去这里：

http://unicodelookup.com/

而查到，对应的unicode的值，html中的写法。

提醒：

可以通过输入：

0x3232

而查到该特殊字符。

解决办法

解决办法，则是不同情况，不同处理：

（1）如果对于这些特殊字符，你不是很关心，即使不显示也无所谓，但是希望剩下的，其他大多数的正常的字符都能显示。

即，忽略掉特殊字符，显示哪些能显示的字符， 那么可以改为如下代码：

python_2.x_print_unicode_omit_special.py

```python
#!/usr/bin/python
# -*- coding: utf-8 -*-
"""
--------------------------------------------------------------------------
Function:
【总结】Python中常见字符编码和解码方面的错误及其解决办法
http://www.crifan.com/summary_python_2_x_common_string_encode_decode

Author:     Crifan
Verison:    2012-11-29
--------------------------------------------------------------------------
"""

#任何字符，都可以在：
#http://unicodelookup.com/
#中，查找到对应的unicode的值

def python2xPrintUnicodeOmitSpecial():
    #http://zhidao.baidu.com/question/500133781.html
    slashUStr = "\\u3232\\u6674"; #(有) 晴
    decodedUniChars = slashUStr.decode("unicode-escape"); #此处已经可
    unicodeButContainSpecialChar = decodedUniChars;
    #print "unicodeButContainSpecialChar=",unicodeButContainSpecialCh

    #此处在GBK编码的cmd中输出的话，会出现错误的：
    #UnicodeEncodeError: 'gbk' codec can't encode character u'\u3232'
    #那是因为，Unicode字符：0x3232，是个特殊字符，而此字符，在GBK编码字符

    #如果只是想要：
    #显示那些正常可以显示的字符，忽略个别特殊不能显示的字符
    #那么可以改为如下代码：
    encodedShowableGbk = unicodeButContainSpecialChar.encode("GBK", '
    print "encodedShowableGbk=",encodedShowableGbk; #encodedShowableG

##############################################################################
if __name__=="__main__":
    python2xPrintUnicodeOmitSpecial();
```

注：我之前遇到的一个情况，就是通过添加ignore去处理的：

【已解决】UnicodeEncodeError: ‘gbk’ codec can’t encode character u’\u200e’ in position 43: illegal

multibyte sequence

（2）如果必须要显示这些字符，或者说必须要保留这些字符。那么本身对于打印这个需求来说，是可以不打印的，因为本身已获得了正常的Unicode字符了。

然后剩下的，只是尽量你自己所需要的后续的处理即可。

即， 已经得到了正确的unicode字符了，后续该咋办咋办，可以不打印的时候，就不打印，也就不会出错了。

举一反三

如果以后遇到这种，虽然已获得了Unicode字符串，但是还是无法打印等情况，则就要注意去调查一下，是否是由于此处有特殊字符，不存在于输出目标所用字符集中，才导致此问题的。

进一步的举一反三

如果你对于上述代码中的encode中的ignore不熟悉，那么你自然应该想到，去了解，去学习这方面的内容。

然后通过google搜：

unicode encode ignore

就可以找到Python官网的解释了：

Unicode HOWTO — Python v2.7.3 documentation

而如果你再稍微积极思考的话，就会想到：

既然encode有个ignore参数，那是不是还有其他参数？

对应的函数原型是啥？

由此，去通过Python的自带手册，就找到对应你所要的内容了：

> ***str.encode([encoding[, errors]])***
>
> *Return an encoded version of the string. Default encoding is the current default string encoding.* errors *may be given to set a different error handling scheme.*

> *The default for* errors *is 'strict', meaning that encoding errors raise a UnicodeError. Other possible values are 'ignore', 'replace','xmlcharrefreplace', 'backslashreplace' and any other name registered via codecs.register_error(), see section Codec Base Classes. For a list of possible encodings, see section Standard Encodings.*
>
> *New in version 2.0.*
>
> *Changed in version 2.3: Support for 'xmlcharrefreplace' and'backslashreplace' and other error handling schemes added.*
>
> *Changed in version 2.7: Support for keyword arguments added.*

然后再细心些的话，还会发现手册中，str.encode上面，还有个对应的decode函数，也是有一些相关的参数，比如'ignore', 'replace'的：

> ***str.decode([encoding[, errors]])***
>
> *Decodes the string using the codec registered for* encoding. encoding*defaults to the default string encoding.* errors *may be given to set a different error handling scheme. The default is 'strict', meaning that encoding errors raise UnicodeError. Other possible values are 'ignore','replace' and any other name registered via codecs.register_error(), see section Codec Base Classes.*
>
> *New in version 2.2.*
>
> *Changed in version 2.3: Support for other error handling schemes added.*
>
> *Changed in version 2.7: Support for keyword arguments added.*

由此，多去学习官网的手册，自然会了解到，函数的最权威的解释和用法。

总结

凡是都是，要积极思考，通过有限的经验，去努力获得更多的，相关知识的总结。

而此处上面的这些总结，相应地，也的确是必须经过一堆的错误，从开始的一头雾水，到最后的渐渐清晰，以及最终的，搞懂逻辑和背后的原因。

总之，想要学好Python或其他语言，都还是要足够的积累，足够的练习，以及及时的总结。