Python专题教程：字符串和字符编码

版本：v1.2.1

Crifan Li

摘要

本文是针对Python的中级开发人员，介绍Python中的字符串和字符编码方面的知识，主要包括Python 2.x的str和unicode以及Python 3.x的bytes和str，Python中常见字符编码和解码方面的错误及其解决办法，以及其他一些常见的字符串方面的处理，比如格式化为树形输出，HTML的Entity实体等等

本文提供多种格式供：

在线阅读	HTML [1]	HTMLs [2]	PDF [3]	CHM [4]	TXT [5]	RTF [6]	WEBHELP [7]
下载（7zip压缩包）	HTML [8]	HTMLs [9]	PDF [10]	CHM [11]	TXT [12]	RTF [13]	WEBHELP [14]

HTML版本的在线地址为：

http://www.crifan.com/files/doc/docbook/python_topic_str_encoding/release/html/python_topic_str_encoding.html

有任何意见，建议，提交bug等，都欢迎去讨论组发帖讨论：

http://www.crifan.com/bbs/categories/python_topic_str_encoding/

修订历史		
修订 1.2.1	2015-05-26	crl
1. 把之前教程的地址整理过来 2. 添加新帖子的链接 3. 增加新的章节：总结		

[1] http://www.crifan.com/files/doc/docbook/python_topic_str_encoding/release/html/python_topic_str_encoding.html
[2] http://www.crifan.com/files/doc/docbook/python_topic_str_encoding/release/htmls/index.html
[3] http://www.crifan.com/files/doc/docbook/python_topic_str_encoding/release/pdf/python_topic_str_encoding.pdf
[4] http://www.crifan.com/files/doc/docbook/python_topic_str_encoding/release/chm/python_topic_str_encoding.chm
[5] http://www.crifan.com/files/doc/docbook/python_topic_str_encoding/release/txt/python_topic_str_encoding.txt
[6] http://www.crifan.com/files/doc/docbook/python_topic_str_encoding/release/rtf/python_topic_str_encoding.rtf
[7] http://www.crifan.com/files/doc/docbook/python_topic_str_encoding/release/webhelp/index.html
[8] http://www.crifan.com/files/doc/docbook/python_topic_str_encoding/release/html/python_topic_str_encoding.html.7z
[9] http://www.crifan.com/files/doc/docbook/python_topic_str_encoding/release/htmls/index.html.7z
[10] http://www.crifan.com/files/doc/docbook/python_topic_str_encoding/release/pdf/python_topic_str_encoding.pdf.7z
[11] http://www.crifan.com/files/doc/docbook/python_topic_str_encoding/release/chm/python_topic_str_encoding.chm.7z
[12] http://www.crifan.com/files/doc/docbook/python_topic_str_encoding/release/txt/python_topic_str_encoding.txt.7z
[13] http://www.crifan.com/files/doc/docbook/python_topic_str_encoding/release/rtf/python_topic_str_encoding.rtf.7z
[14] http://www.crifan.com/files/doc/docbook/python_topic_str_encoding/release/webhelp/python_topic_str_encoding.webhelp.7z

Python专题教程：字符串和字符编码:

Crifan Li

版本：v1.2.1

出版日期 2015-05-26

[15] http://www.crifan.com/files/doc/docbook/soft_dev_basic/release/html/soft_dev_basic.html#cc_by_nc

目录

表格清单

前言

1. 本文目的

本文目的在于，通过此文，对于Python中的字符串和字符编码方面的知识，有个系统的了解。

以及了解Python中常见的字符编码方面的问题，背后的逻辑，以及如何解决。

2. 前提

在学习Python中的字符编码方面的内容之前，需要自己本身已经对于字符串编码本身的基础知识，比较了解，然后才能真正明白Python中的字符编码是怎么回事。

如果不了解字符编码，请先去参考：

字符编码详解 [1]

[1] http://www.crifan.com/files/doc/docbook/char_encoding/release/html/char_encoding.html

第 1 章 Python 2.x的str和unicode vs. Python 3.x的bytes和str

相关旧帖

【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 [1]

【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 [2]

[1] http://www.crifan.com/summary_python_string_encoding_decoding_difference_and_comparation_python_2_x_str_unicode_vs_python_3_x_bytes_str

[2] http://www.crifan.com/python_3_x_auto_handle_string_encode_and_decode_to_and_from_unicode_then_output_to_cmd

第 2 章 Python中常见字符编码和解码方面的错误及其解决办法

相关旧帖

【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 [1]

【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]

【整理】Python中遇到"UnicodeDecodeError: 'gbk' codec can't decode bytes in position 2-3: illegal multibyte sequence"之类的编码或解码的错误时如何处理 [3]

【已解决】Python字符串处理出现错误：UnicodeDecodeError: 'ascii' codec can't decode byte 0xe6 in position 0: ordinal not in range(128) [4]

2.1. Python中的 UnicodeEncodeError

Python中很容易出现一些关于编码方面的错误。

其中，最常见的一个就是UnicodeEncodeError

下面，总结一下各种出现的原因，以及相应的解决办法。

2.1.1. 如果打印显示终端中字符编码不支持所打印字符的话，也会出现错误UnicodeEncodeError

Python中的编码问题，的确很容易让人头疼。

之前已经遇到过N个类似的UnicodeEncodeError，并且也都一一解决了。

但是今天又遇到一个：

```
UnicodeEncodeError: 'gbk' codec can't encode character u'\u2665' in position 160: illegal multibyte sequence
```

最后发现，原来是在用：

```
print "footerUni=",footerUni;
```

去打印此unicode类型的字符串footerUni，由于其中包含一些字符，其无法在当前命令行下的GBK编码环境中找到对应字符，所以才报了以上的错误的。

所以，真的是，对于Python中的字符编码方面，连使用print都要小心。

在这点上，真的很让人无语。。。

2.1.2. 在处理中文简体和中文繁体的的时候，使用目标编码中不存在的中文字符，也会导致UnicodeEncodeError

在一个UTF-8的Python文件中，有如下代码：

[1] http://www.crifan.com/summary_python_2_x_common_string_encode_decode_error_reason_and_solution
[2] http://www.crifan.com/python_string_encoding_declare_encoding_vs_file_real_encoding/
[3] http://www.crifan.com/summary_python_unicodedecode_error_possible_reasons_and_solutions/
[4] http://www.crifan.com/python_unicodedecodeerror_codec_can_not_decode_byte_in_position_ordinal_not_in_range

```
str = '电脑';
......
goods.append(urllib.quote(str.decode('utf-8').encode('big5')));
```

此时，就会出现错误：

```
UnicodeEncodeError: 'big5' codec can't encode character u'\u7535' in position 0: illegal
 multibyte sequence
```

此问题的原因在于，对于所输入的str类型的中文简体字符”电脑“来说，虽然通过str.decode('utf-8')可以获得了Unicode的中文简体字符”电脑“

但是将此Unicode类型的，简体中文字符”电脑“，去进行BIG5编码的时候，结果由于BIG5编码中，根本就不存在上述这两个简体中文的汉字”电脑“，由此出现上述UnicodeEncodeError的错误。

即，无法将简体中文的”电脑“，从Unicode转换为BIG5编码，因为BIG5编码字符集中，就不存在”电脑“这两个字符。

对此问题，详细去探究，就可以更了解中文简体和中文繁体之间的差别。

此处，先来明确一下我们的目标，即，希望是在BIG5编码中，能显示”电脑“这两个字符。

而对于BIG5编码所对应的所有字符，可以去Big5 (Traditional Chinese) character code table[5]中查找

我们会发现，BIG5编码中，根据就找不到这两个字符。

那如何才能让”电脑“这两个字符，在繁体中显示呢？

那就需要，将中文简体的”电脑“，去先转换为对应的中文繁体字，然后就可以编码为BIG5，可以显示了。

而关于中文简体和中文繁体之间的转换，可以去这里：

简繁转换 [6]

进入该网站后，输入”电脑“，然后点击”TW Unicode“或”TW BIG“，就可以转换为对应的繁体字”電腦“了。

很明显，”電腦“这两个中文繁体字，那肯定是能在Big5 (Traditional Chinese) character code table[7]中找到的

而对应的，这几个中文字符所对应的Unicode的值，可以去Unicode Lookup[8]查到。

此处整理如下：

表 2.1. ”电脑“和”電腦“所对应的Unicode值

Unicode character	Oct	Dec	Hex	HTML
电 cjk unified ideograph 7535	072465	30005	0x7535	&#30005;
脑 cjk unified ideograph 8111	0100421	33041	0x8111	&#33041;
電 cjk unified ideograph 96fb	0113373	38651	0x96FB	&#38651;
腦 cjk unified ideograph 8166	0100546	33126	0x8166	&#33126;

[5] http://ash.jp/code/cn/big5tbl.htm
[6] http://www.j4.com.tw/big-gb/
[7] http://ash.jp/code/cn/big5tbl.htm
[8] http://unicodelookup.com/

此处，“电”所对应的Unicode值为7535，Unicode写法为\u7535，就是对应着上述出错的内容中所指示的，

BIG5编码器，无法对于Unicode为\u7535的字符进行编码，即BIG5无法编码“电”，因为其本身就没这个字符。

对应的，如果用GB2312/GBK/GB18030去编码“电”，那么肯定是可以的，但是肯定也是无法编码“電”的。

所以，对于中文简体和繁体之间，如果想要正确显示，需要先转换为对应繁体或简体，然后才能用正确的编码，去解析其所支持的字符的。

而有人看到这里，可能会疑惑了，因为比如对于有些中文字符，比如“手机”，

虽然没有去用什么简体转换为繁体，而直接用上述BIG5去编码，会发现程序可以正常执行，不会出现那个UnicodeEncodeError的

对此，你去简繁转换[9]中搜一下“手”和“机”这两个汉字，发现BIG5编码中，也的确是存在这两个汉字的，

所以原因就很清楚了：对于有些字符，中文简体和中文繁体的写法，是一样的，所以才可以在GB212/GBK/GB18030和BIG5之间来回转换，而不会出现问题。

换句话说，如果你想要在BIG5编码中显示（编码）某个中文简体的话，那前提要确保该字符是可以在简繁转换[10]中能查找到的。

否则，说明BIG5中没有改汉字，需要用到简繁转换[11]去先进行简繁体转换，得到转换后的繁体字后，才能得到BIG5编码的字符，用BIG5去编解码，去显示对应的繁体中文。

上述步骤，很明显，可以省略了去BIG5编码表中查找，而直接利用上述简繁体转换的网站去实现转换，或者本地建立一个转换表，直接从输入的中文简体，得到输出的中文繁体，即可继续后续处理，而无需关注，在简体和繁体中，哪些是一样的写法，哪些是不一样的写法。

2.2. str的解码decode

decode函数，是str类型变量本身就有的函数，用于实现将某种编码的字符，解码为Unicode类型字符。

将某种编码的str字符解码为Unicode字符，通常做法为：

```
defCmtCharset = "GB18030";
dataJsonStrUni = dataJsonStr.decode(defCmtCharset);
```

不过，上述用法，是在你知道了字符编码的前提下，才能这么做的。

如果是对于输入的字符串，可能是多种不同的编码，即无法确定输入字符编码的前提下，想要对其解码的话，可以这么实现：

先利用chardet判断字符编码类型，然后再去解码：

```
possibleCharset = crifanLib.getStrPossibleCharset(dataJsonStr);
dataJsonStrUni = dataJsonStr.decode(possibleCharset);
```

[9] http://www.j4.com.tw/big-gb/
[10] http://www.j4.com.tw/big-gb/
[11] http://www.j4.com.tw/big-gb/

其中getStrPossibleCharset是我自己写的函数

详见：getStrPossibleCharset函数详解[12]

但是，有时候，你会发现，即使如此，也可能遇到decode失败的情况。

因为有时候所输入的字符串，本身不完全是某种单一的编码，而是2种或更多种不同的编码的混合体，此时，此处通过getStrPossibleCharset中的chardet所得到的值，就未必是0.99,而可能是某个很低的值，比如0.6，此时用此编码去解码，就可能遇到失败的情况了。

这种变态的，多种编码混合的字符串，我之前就在折腾给BlogsToWordpress[13]添加QQ空间支持的过程中，处理QQ空间帖子的评论数据中，就遇到过。

当时很是郁闷，无法有效解决此问题，导致对于有些帖子的评论数据，无法继续处理。

直到后来，直到对于decode函数来说，还有个ignore参数，可以实现，在解码过程中，对于那些（用当前编码）无法解码的，不支持的字符，采取忽略的策略，而使得不会出现解码失败，然后最终可以成功解码整个字符串。

对应的函数详细解释，在Python的帮助文档中可以找到：

> **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.

然后，上述的代码，改为：

```
defCmtCharset = "GB18030";
dataJsonStrUni = dataJsonStr.decode(defCmtCharset, 'ignore');
```

即可实现，对于绝大多数的GB18030的中文字符，都可以正确解码为Unicode字符了。

即使遇到一些变态的混合型编码的字符（比如其中一部分是ISO-8859-2编码，其他部分是其他的某种编码），也不会出现decode出错，而使得代码可以继续运行了。

当然，需要注意一点的是，我这里，其中被ignore的个别特殊字符，由于是在评论的数据中的个别特殊字符，所以不重要，忽略了也无所谓，所以才可以使用ignore参数的。要是你的代码中，不允许忽略任何内容，那你就得想其他办法了。

[12] http://www.crifan.com/files/doc/docbook/crifanlib_python/release/html/crifanlib_python.html#getstrpossiblecharset
[13] http://www.crifan.com/crifan_released_all/website/python/blogstowordpress/

第 3 章 Python中常见的字符串和编码等相关的处理

相关旧帖

【已解决】Python中，将字符串转换为函数，并且实现带参数的函数调用 [1]

3.1. Python中的反斜杠u类型（\uXXXX）的字符串

相关旧帖

【整理】Python中，如何将反斜杠u类型（\uXXXX）的字符串，转换为对应的unicode的字符 [2]

3.2. Python中字符串的格式化

相关旧帖

【整理】Python中将（字典，列表等）变量格式化成（漂亮的，树形的，带缩进的，JSON方式的）字符串输出 [3]

【记录】折腾Python中的pprint [4]

【已解决】Python中，带填充和设置对齐方式的，格式化字符串输出 [5]

3.3. Python中处理HTML

相关旧帖

【整理】Python中解码（decode）HTML中的实体（entity）+ 将name entity转为code point entity + 将code point entity转为name entity [6]

[1] http://www.crifan.com/python_convert_string_to_function_then_call
[2] http://www.crifan.com/python_decode_slash_u_unicode_escape_string_into_unicode_chars
[3] http://www.crifan.com/format_dictionary_list_variable_into_prettified_tree_like_with_indent_json_string_then_output
[4] http://www.crifan.com/python_module_pprint
[5] http://www.crifan.com/python_string_format_fill_with_chars_and_set_alignment
[6] http://www.crifan.com/python_decode_html_entity_and_convert_between_name_entity_and_code_point_entity

参考书目

[1] Big5 (Traditional Chinese) character code table[1]

[2] 简繁转换[2]

[3] Unicode Lookup[3]

[4] 【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[4]

[1] http://ash.jp/code/cn/big5tbl.htm
[2] http://www.j4.com.tw/big-gb/
[3] http://unicodelookup.com/
[4] http://www.crifan.com/summary_python_string_encoding_decoding_difference_and_comparation_python_2_x_str_unicode_vs_python_3_x_bytes_str