<https://wiki.python.org/moin/UnicodeEncodeError>

Unicode\_str.encode(‘utf-8’): Unicode -> utf-8

Str.decode(): utf-8:Unicode

<https://www.joelonsoftware.com/2003/10/08/the-absolute-minimum-every-software-developer-absolutely-positively-must-know-about-unicode-and-character-sets-no-excuses/>

python中的ord，chr函数 <http://www.cnblogs.com/hushaojun/p/4532635.html>

chr()、unichr()和ord()

chr()函数用一个范围在range（256）内的（就是0～255）整数作参数，返回一个对应的字符。unichr()跟它一样，只不过返回的是Unicode字符，这个从Python 2.0才加入的unichr()的参数范围依赖于你的Python是如何被编译的。如果是配置为USC2的Unicode，那么它的允许范围就是range（65536）或0x0000-0xFFFF；如果配置为UCS4，那么这个值应该是range（1114112）或0x000000-0x110000。如果提供的参数不在允许的范围内，则会报一个ValueError的异常。

ord()函数是chr()函数（对于8位的ASCII字符串）或unichr()函数（对于Unicode对象）的配对函数，它以一个字符（长度为1的字符串）作为参数，返回对应的ASCII数值，或者Unicode数值，如果所给的Unicode字符超出了你的Python定义范围，则会引发一个TypeError的异常。

常用汉字的UTF-8编码及编码范围

<http://blog.csdn.net/fatherican/article/details/7198502>

UTF8的是u4E00 - u9FFF

<http://blog.csdn.net/u014431852/article/details/48003321>

UTF-8

互联网的普及，强烈要求出现一种统一的编码方式。UTF-8就是在互联网上使用最广的一种Unicode的实现方式。其他实现方式还包括UTF-16（字符用两个字节或四个字节表示）和UTF-32（字符用四个字节表示），不过在互联网上基本不用。重复一遍，这里的关系是，UTF-8是Unicode的实现方式之一。

UTF-8最大的一个特点，就是它是一种变长的编码方式。它可以使用1~4个字节表示一个符号，根据不同的符号而变化字节长度。

UTF-8的编码规则很简单，只有二条：

1）对于单字节的符号，字节的第一位设为0，后面7位为这个符号的unicode码。因此对于英语字母，UTF-8编码和ASCII码是相同的。

2）对于n字节的符号（n>1），第一个字节的前n位都设为1，第n+1位设为0，后面字节的前两位一律设为10。剩下的没有提及的二进制位，全部为这个符号的unicode码。

下表总结了编码规则，字母x表示可用编码的位。

Unicode符号范围 | UTF-8编码方式

(十六进制) | （二进制）

——————–+———————————————

0000 0000-0000 007F | 0xxxxxxx

0000 0080-0000 07FF | 110xxxxx 10xxxxxx

0000 0800-0000 FFFF | 1110xxxx 10xxxxxx 10xxxxxx

0001 0000-0010 FFFF | 11110xxx 10xxxxxx 10xxxxxx 10xxxxxx

跟据上表，解读UTF-8编码非常简单。如果一个字节的第一位是0，则这个字节单独就是一个字符；如果第一位是1，则连续有多少个1，就表示当前字符占用多少个字节。

下面，还是以汉字”严”为例，演示如何实现UTF-8编码。

已知”严”的unicode是4E25（100111000100101），根据上表，可以发现4E25处在第三行的范围内（0000 0800-0000 FFFF），因此”严”的UTF-8编码需要三个字节，即格式是”1110xxxx 10xxxxxx 10xxxxxx”。然后，从”严”的最后一个二进制位开始，依次从后向前填入格式中的x，多出的位补0。这样就得到了，”严”的UTF-8编码是”11100100 10111000 10100101”，转换成十六进制就是E4B8A5。

Python：过滤一些特殊字符

content.decode('utf-8').encode('gbk')

<https://segmentfault.com/q/1010000000645750>

针对这里的问题，我们可以先将文件名解码成unicode，然后用gbk编码，如果抛出异常，说明编码失败，即文件名中存在特殊字符。下面是一个小例子：

#! /usr/bin/env python

# -\*- coding: utf-8 -\*-

file\_names = ["œ®特殊字符.txt",

"没有特殊字符.txt",

"ßååå.txt"

]

for file\_name in file\_names:

try:

file\_name.decode("utf-8").encode("gbk")

print file\_name, "..."

except UnicodeEncodeError:

print file\_name, "中奖"

"""outputs:

œ®特殊字符.txt 中奖

没有特殊字符.txt ...

ßååå.txt 中奖

"""

<https://stackoverflow.com/questions/27836738/javas-getbyte-equivalent-in-python>

The **bytearray** you have created in Python contains the bytes you want. To see their decimal representation, print the bytes one by one:

>>> for x in bytearray('sr\_shah','windows-1252'): print(x)

.NET自动字符编码识别程序库 NChardet

<http://www.cnblogs.com/hhh/archive/2007/01/27/632251.html>

【已解决】python中文字符乱码（GB2312，GBK，GB18030相关的问题）

<https://www.thinksaas.cn/group/topic/493058/>

给你一个字符串，怎么判断该字符串是否含有乱码？

<http://www.oschina.net/question/927134_123511>