【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

2013 年 7 月 19 日 下午 1:37 crifan 已有11213人围观 2个评论

【背景】

python中的字符串编码，搞晕很多人，包括之前的我。

随着对于python的编码的深入了解，后来才算搞懂，关于python的，用encoding去声明的文件编码，和，python文件实际的编码，之间的关系。

Python中用encoding声明的文件编码和文件的实际编码之间的关系

1.解释声明的编码和真实的编码之前，需要了解，正常的情况下，python文件的话，如何声明文件的编码：

【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

背景知识

1.不了解各种编码，尤其是常见的UTF-8，GBK等编码的，去看：

字符编码详解

2.不会对文件进行编码转换的，可以去用Notepad++的编码转换功能：

用Notepad++实现不同字符编码之间的转换

3.对于windows的cmd的默认是GBK编码不熟悉的，去看：

Windows的命令行工具: cmd

然后再来说，关于声明的编码，和文件本身的编码事情。

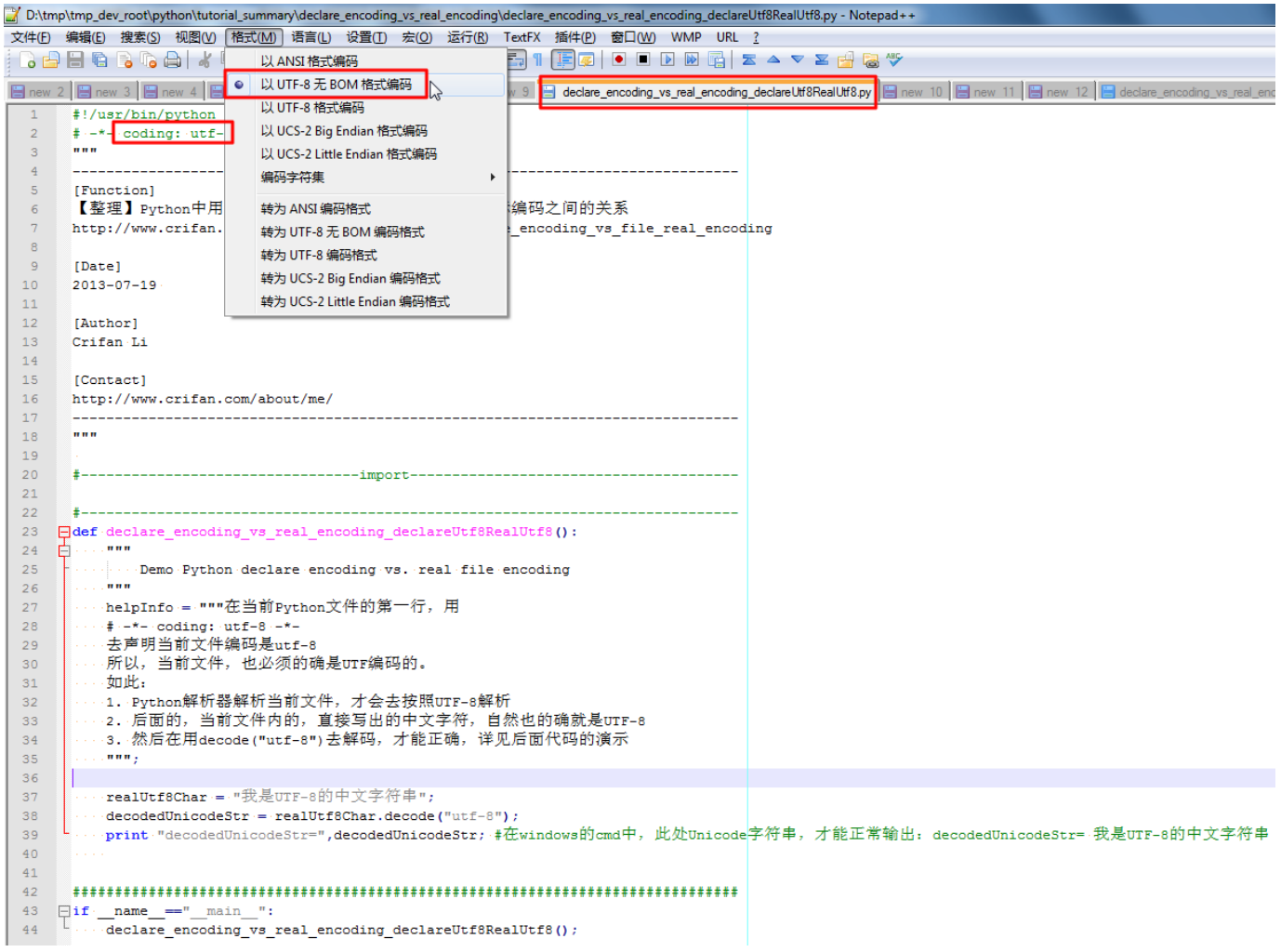
声明的编码和实际的编码匹配的时候：声明为UTF-8编码，文件实际编码也的确是UTF-8

（1）示例代码：

```
#!/usr/bin/python
# -*- coding: utf-8 -*-
"""
--------------------------------------------------------------------------
[Function]
【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
http://www.crifan.com/python_string_encoding_declare_encoding_vs_file

[Date]
2013-07-19

[Author]
Crifan Li

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

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

#----------------------------------------------------------------------
def declare_encoding_vs_real_encoding_declareUtf8RealUtf8():
    """
        Demo Python declare encoding vs. real file encoding
    """
    helpInfo = """在当前Python文件的第一行，用
    # -*- coding: utf-8 -*-
    去声明当前文件编码是utf-8
    所以，当前文件，也必须的确是UTF编码的。
    如此：
    1. Python解析器解析当前文件，才会去按照UTF-8解析
    2. 后面的，当前文件内的，直接写出的中文字符，自然也的确就是UTF-8
    3. 然后在用decode("utf-8")去解码，才能正确，详见后面代码的演示
    """;

    realUtf8Char = "我是UTF-8的中文字符串";
    decodedUnicodeStr = realUtf8Char.decode("utf-8");
    print "decodedUnicodeStr=",decodedUnicodeStr; #在windows的cmd中，
```

```
41
42  ##############################################################################
43  if __name__=="__main__":
44      declare_encoding_vs_real_encoding_declareUtf8RealUtf8();
```

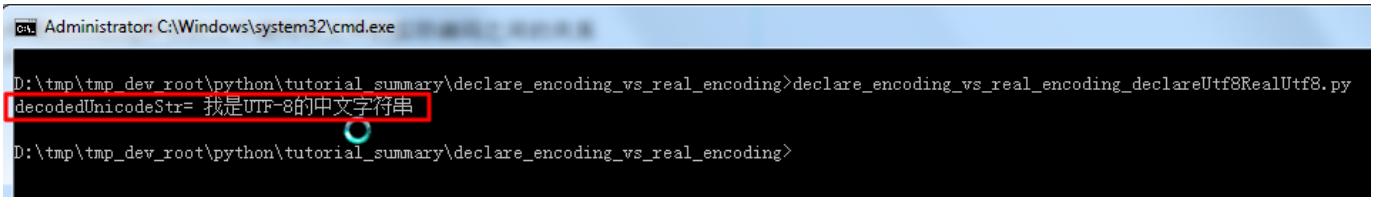
（2）在Notepad++中，可以看出，当前文件我的确已经设置成了UTF-8：



（3）代码下载（右键另存为）：

declare_encoding_vs_real_encoding_declareUtf8RealUtf8.py

（4）运行效果如下：



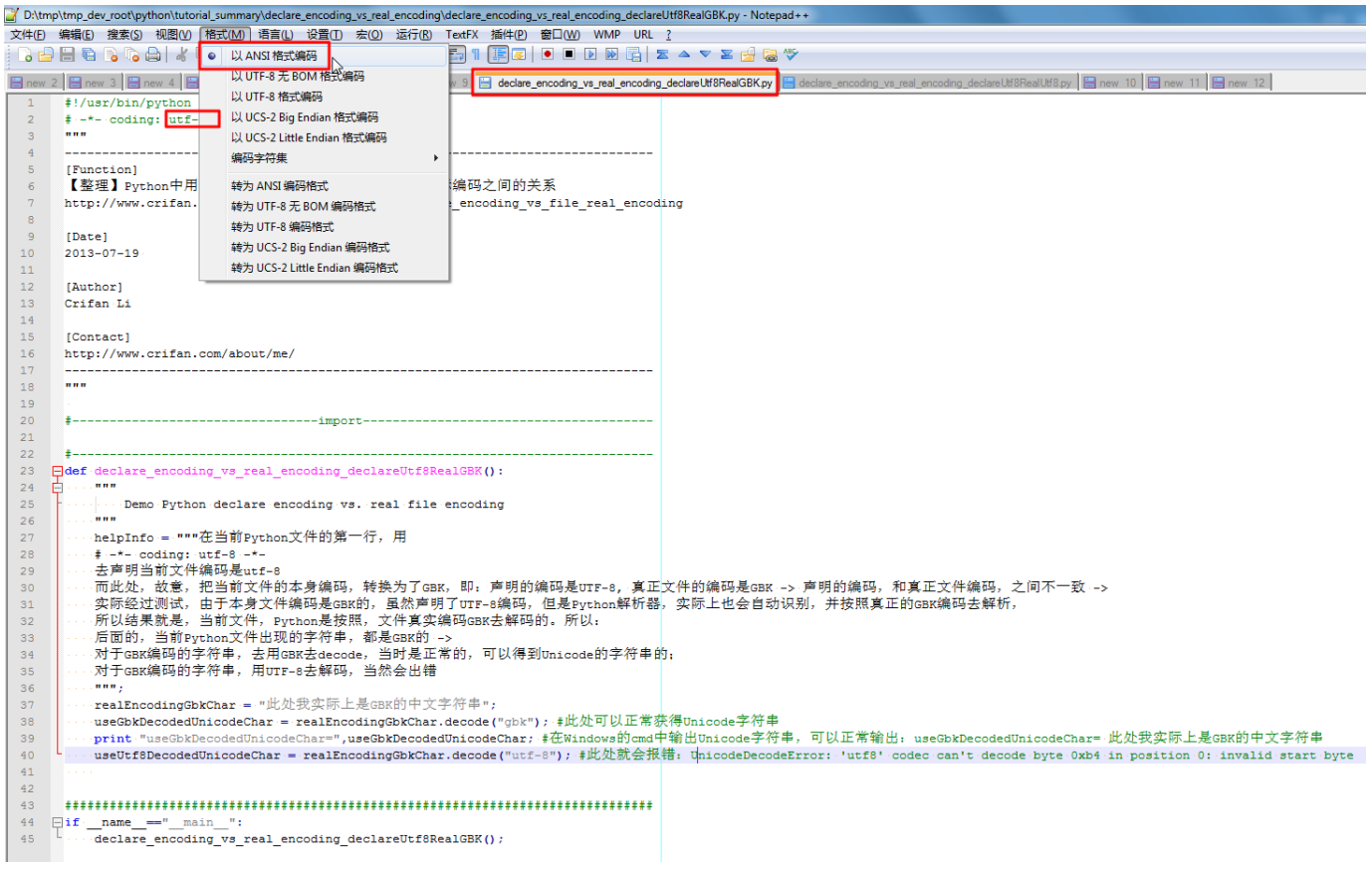
声明的编码和实际的编码不匹配的时候：声明为UTF-8，文件实际编码是（ANSI的）GBK

（1）示例代码：

```python
#!/usr/bin/python
# -*- coding: utf-8 -*-
"""
--------------------------------------------------------------------------
[Function]
【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
http://www.crifan.com/python_string_encoding_declare_encoding_vs_file

[Date]
2013-07-19

[Author]
Crifan Li

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

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

#--------------------------------------------------------------------------
def declare_encoding_vs_real_encoding_declareUtf8RealGBK():
    """
        Demo Python declare encoding vs. real file encoding
    """
    helpInfo = """在当前Python文件的第一行，用
    # -*- coding: utf-8 -*-
    去声明当前文件编码是utf-8
    而此处，故意，把当前文件的本身编码，转换为了GBK，即：声明的编码是UTF-8
    实际经过测试，由于本身文件编码是GBK的，虽然声明了UTF-8编码，但是Python
    所以结果就是，当前文件，Python是按照，文件真实编码GBK去解码的。所以：
    后面的，当前Python文件出现的字符串，都是GBK的 ->
    对于GBK编码的字符串，去用GBK去decode，当时是正常的，可以得到Unicode的
    对于GBK编码的字符串，用UTF-8去解码，当然会出错
    """;
    realEncodingGbkChar = "此处我实际上是GBK的中文字符串";
    useGbkDecodedUnicodeChar = realEncodingGbkChar.decode("gbk"); #此
    print "useGbkDecodedUnicodeChar=",useGbkDecodedUnicodeChar; #在Wi
    useUtf8DecodedUnicodeChar = realEncodingGbkChar.decode("utf-8");
```

```
43  ##############################################################################
44  if __name__=="__main__":
45      declare_encoding_vs_real_encoding_declareUtf8RealGBK();
```

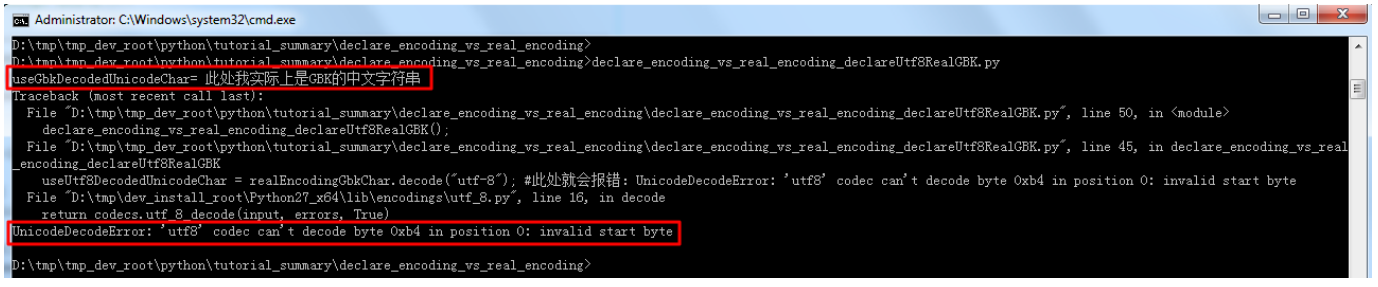
（2）在Notepad++中，可以看出，当前文件我是故意，已经转换为GBK了：



（3）代码下载（右键另存为）：

declare_encoding_vs_real_encoding_declareUtf8RealUtf8.py

（4）运行效果如下：



总结

可以看出：

虽然文件编码声明，即使和实际文件本身编码错误，Python解析器，也是识别真正的文件的编码的。

但是更明显，还是要尽量保持，python文件声明和实际上的文件本身的编码，一致，这样才不容易出问题。