PyQt 中文编码问题

1.python 2.x ， 采用 bytes 格式的字符串， 而 python3.0 采用 unicode 编码

2.QT 所有都采用 unicode 编码

3.python -> QT : str = unicode(str,’utf-8′)　　#unicode 中文显示 统一化可以先用 str(string)转化一下

QT -> python : b = s.toLocal8Bit()

先看一个正确的例子：

```
1 from PyQt4 import QtGui, QtCore

2 s = QtCore.QString(u’中文’)
3 b = s.toLocal8Bit()
4 u = unicode(b,’gbk’,'ignore’)

5 print b
6 print u
```

这段程序的输出是：
> 中文
> 中文

来解释一下上面这段程序。

1. QString 必须接受 unicode 编码的字符串

print type(s) 可以得到如下输出
> <class ‘PyQt4.QtCore.QString’>
s 是一个 QString 对象

但是如果将第 2 行改成
s = QtCore.QString(‘中文’)
将会出现乱码，这是因为在 Qt 中，QString 只接受以 unicode 编码的字符串。而在 Python 中，’中文’是 bytes 格式的字符串，u’中文’才是 unicode 的字符串。但在 python 3.0 中，字符串默认是 unicode 的，而 bytes 格式的字符串必须加 b 前缀。

2. Qt 中的 QByteArray 等同于 Python 中的 bytes 格式字符串

print type(b) 得到
> <class ‘PyQt4.QtCore.QByteArray’>
b 是一个 QByteArray 对象

在 PyQt 中，QString.toLocal8Bit() 返回一个 QByteArray 对象。而 print b 能够正确显示，实际上 str(b) 就可以将 b 转换成 Python 字符串。QString 对象在 Python 中是不能直接显示的，print s 会导致错误。

3. Python 中的字符串格式转换

print type(u) 得到
> <type ‘unicode’>
u 是一个 python 下的 unicode 字符串

既然 QByteArray 等同于 bytes 格式的字符串，那么第 4 行代码就顺理成章了，这完全是 Python 内部的事情。这行代码所做的只不过是在 bytes 字符串和 unicode 字符串之间转换，对于中文字符串，需要显式的指定编码。