PYTHON-进阶-编码处理小结

整理下python编码相关的内容

注意: 以下讨论为Python2.x版本, Py3k的待尝试

开始

用python处理中文时，读取文件或消息，http参数等等

一运行，发现乱码(字符串处理，读写文件，print)

然后，大多数人的做法是，调用encode/decode进行调试，并没有明确思考为何出现乱码

所以调试时最常出现的错误

错误1

```
Traceback (most recent call last):
File "<stdin>", line 1, in <module>
UnicodeDecodeError: 'ascii' codec can't decode byte 0xe6 in position 0: ordinal not
in range(128)
```

错误2

```
Traceback (most recent call last):
File "<stdin>", line 1, in <module>
File
"/System/Library/Frameworks/Python.framework/Versions/2.7/lib/python2.7/encodings/utf_
 line 16, in decode
    return codecs.utf_8_decode(input, errors, True)
UnicodeEncodeError: 'ascii' codec can't encode characters in position 0-1: ordinal
not in range(128)
```

首先

必须有大体概念，了解下字符集，字符编码

ASCII | Unicode | UTF-8 | 等等

字符编码笔记：ASCII，Unicode和UTF-8

淘宝搜索技术博客-中文编码杂谈

str 和 unicode

> str和unicode都是basestring的子类

所以有判断是否是字符串的方法

```
def is_str(s):
    return isinstance(s, basestring)
```

> str和unicode 转换

decode 文档

encode 文档

```
str  -> decode('the_coding_of_str') -> unicode
unicode -> encode('the_coding_you_want') -> str
```

> 区别

str是字节串，由unicode经过编码(encode)后的字节组成的

声明方式

```
s = '中文'
s = u'中文'.encode('utf-8')

>>> type('中文')
<type 'str'>
```

求长度(返回字节数)

```
>>> u'中文'.encode('utf-8')
'\xe4\xb8\xad\xe6\x96\x87'
>>> len(u'中文'.encode('utf-8'))
6
```

unicode才是真正意义上的字符串，由字符组成

声明方式

```
s = u'中文'
s = '中文'.decode('utf-8')
s = unicode('中文', 'utf-8')

>>> type(u'中文')
<type 'unicode'>
```

求长度(返回字符数),在逻辑中真正想要用的

```
>>> u'中文'
u'\u4e2d\u6587'
>>> len(u'中文')
2
```

结论

搞明白要处理的是str还是unicode, 使用对的处理方法(str.decode/unicode.encode)

下面是判断是否为unicode/str的方法

```
>>> isinstance(u'中文', unicode)
True
>>> isinstance('中文', unicode)
```

```
False

>>> isinstance('中文', str)
True
>>> isinstance(u'中文', str)
False
```

简单原则：不要对str使用encode，不要对unicode使用decode (事实上可以对str进行encode的，具体见最后，为了保证简单，不建议)

```
>>> '中文'.encode('utf-8')
Traceback (most recent call last):
File "<stdin>", line 1, in <module>
UnicodeDecodeError: 'ascii' codec can't decode byte 0xe4 in position 0: ordinal not
in range(128)

>>> u'中文'.decode('utf-8')
Traceback (most recent call last):
File "<stdin>", line 1, in <module>
File
"/System/Library/Frameworks/Python.framework/Versions/2.7/lib/python2.7/encodings/utf_
 line 16, in decode
    return codecs.utf_8_decode(input, errors, True)
UnicodeEncodeError: 'ascii' codec can't encode characters in position 0-1: ordinal
not in range(128)
```

不同编码转换,使用unicode作为中间编码

```
#s是code_A的str
s.decode('code_A').encode('code_B')
```

文件处理,IDE和控制台

处理流程，可以这么使用，把python看做一个水池，一个入口，一个出口

入口处，全部转成unicode, 池里全部使用unicode处理，出口处，再转成目标编码(当然，有例外，处理逻辑中要用到具体编码的情况)

```
读文件

外部输入编码，decode转成unicode

处理(内部编码，统一unicode)

encode转成需要的目标编码
```

写到目标输出(文件或控制台)

IDE和控制台报错，原因是print时，编码和IDE自身编码不一致导致

输出时将编码转换成一致的就可以正常输出

```
>>> print u'中文'.encode('gbk')
????
>>> print u'中文'.encode('utf-8')
中文
```

建议

规范编码

统一编码，防止由于某个环节产生的乱码

环境编码，IDE/文本编辑器, 文件编码，数据库数据表编码

保证代码源文件编码

这个很重要

py文件默认编码是ASCII, 在源代码文件中，如果用到非ASCII字符，需要在文件头部进行编码声明

文档

不声明的话，输入非ASCII会遇到的错误,必须放在文件第一行或第二行

```
File "XXX.py", line 3
SyntaxError: Non-ASCII character '\xd6' in file c.py on line 3, but no encoding
declared; see http://www.python.org/peps/pep-0263.html for details
```

声明方法

```
# -*- coding: utf-8 -*-
或者
#coding=utf-8
```

若头部声明coding=utf-8, a = '中文' 其编码为utf-8

若头部声明coding=gb2312, a = '中文' 其编码为gbk

so, 同一项目中所有源文件头部统一一个编码,并且声明的编码要和源文件保存的编码一致(编辑器相关)

在源代码用作处理的硬编码字符串，统一用unicode

将其类型和源文件本身的编码隔离开, 独立无依赖方便流程中各个位置处理

```
if s == u'中文':  #而不是 s == '中文'
    pass
#注意这里 s到这里时，确保转为unicode
```

以上几步搞定后，你只需要关注两个 unicode和 你设定的编码(一般使用utf-8)

处理顺序

1. Decode early
2. Unicode everywhere
3. Encode later

相关模块及一些方法

获得和设置系统默认编码

```
>>> import sys
>>> sys.getdefaultencoding()
'ascii'

>>> reload(sys)
<module 'sys' (built-in)>
>>> sys.setdefaultencoding('utf-8')
>>> sys.getdefaultencoding()
'utf-8'
```

str.encode('other_coding')

在python中，直接将某种编码的str进行encode成另一种编码str

```
#str_A为utf-8
str_A.encode('gbk')
```

执行的操作是

```
str_A.decode('sys_codec').encode('gbk')
这里sys_codec即为上一步 sys.getdefaultencoding() 的编码
```

'获得和设置系统默认编码'和这里的str.encode是相关的，但我一般很少这么用，主要是觉得复杂不可控,还是输入明确decode，输出明确encode来得简单些(个人观点)

chardet

文件编码检测，下载

```
>>> import chardet
>>> f = open('test.txt','r')
>>> result = chardet.detect(f.read())
>>> result
{'confidence': 0.99, 'encoding': 'utf-8'}
```

\u字符串转对应unicode字符串

```
>>> u'中'
u'\u4e2d'

>>> s = '\u4e2d'
>>> print s.decode('unicode_escape')
中

>>> a = '\\u4fee\\u6539\\u8282\\u70b9\\u72b6\\u6001\\u6210\\u529f'
>>> a.decode('unicode_escape')
u'\u4fee\u6539\u8282\u70b9\u72b6\u6001\u6210\u529f'
```

python unicode文档

入口

好了，暂时就这么多，希望讲清楚了

thx

wklken

2013-08-31 于深圳

上一篇: Python资源入口汇总

下一篇: [翻译]快速Python性能优化要点

Nelman Guo • 10个月前
感谢博主的全面介绍！
^ | v • 回复 • 分享 ›

Ggicci • 10个月前
总结得超棒！
^ | v • 回复 • 分享 ›

Will Liu • 1年前
对我有帮助，感谢博主~
^ | v • 回复 • 分享 ›

baocaixiong • 4年前
真的是非常感谢。
^ | v • 回复 • 分享 ›

Pom Yee • 2年前
感谢
^ | v • 回复 • 分享 ›

在 WKLKEN'S BLOG 上还有

vim插件: syntastic[语法检查]
1条评论 • 2年前•
anakin — nice

工作四周年小结
7条评论 • 2年前•
zhkzyth — 深圳做python的好像不多~~有机会面基下，哈哈哈=)

vim插件: delimitmate[符号自动补全]
2条评论 • 2年前•
freetg — http://www.untaken.org/vim-...set pastetoggle=<f3>

vim插件: vundle[管理插件]
2条评论 • 2年前•
wklken — 最新版本已经切换到vim-plug插件了

订阅 在您的网站上使用 Disqus添加 Disqus添加 隐私