利用Python从文件中读取字符串（解决乱码问题）

2019年1月14日 星期一　　上午 0:46

首先声明这篇学习记录是**基于python3**的。

python3中，py文件中默认的文件编码就是unicode，不用像python2中那样加u，比如u'中文'。

不过在涉及路径时，比如C:\Users\Administrator\Desktop\StudyNote\Python，**还是要加r。**

eg:r'C:\Users\Administrator\Desktop\StudyNote\Python'。

因为\是转义符，想输出'\'得写成'\\'才可以。加了r就可以让python自动处理字符串,不让'\'进行转义，因此最终的字符串就是C:\Users\Administrator\Desktop\StudyNote\Python的意思。

OK，继续讲。

首先要明确一点，你的py文件用什么文件编码不重要，都可以顺利执行。不过要保证一点，**保存py文件时的编码必须与编码声明的一样**。假设你的py文件开头有以下编码声明:

eg: # -*- coding: utf-8 -*-

那么保存py文件时的编码也要为utf-8。

另外再普及一下**保存文件编码**时的知识，**用notepad++在选择编码模式的时候，有utf-8和utf-8 without BOM**，这个BOM实际上是在文件开头加注了三个字符，用以表明本文件的编码方式为utf-8，但这个是不需要的，往往会导致读取文件的时候出错（因为多了三个字符嘛，如果要去掉的话还得[3:]一下，挺麻烦的）。所以我们一般用**utf-8 without BOM**.

再言归正传。我们在处理文本文件的时候，**如果该文件是unicode编码，则不需要做任何的处理操作，直接用'r'参数读取直接可用**：

eg:

```
f=open('文件路径','r')
f_read=f.read()
print(f_read)
```

这样就能完整的输出文件里的字符串。

如果不是unicode编码，就不能直接用了,直接读会出现乱码。**只能先以'rb'参数读取二进制文件的方式读取进来，read之后再解码**。

eg:

```
f=open('文件路径','rb')
f_read=f.read()
f_read_decode=f_read.decode('该文件的编码方式')
print(f_read_decode)
```

这样才能完整显示，不然会有乱码出现。

问题又来了，往往我们并不知道该文件的**编码方式**，这该怎么办？

幸好python有个**强大的工具chardet**

eg:

```
import chardet
f=open('文件路径','rb')
f_read=f.read()
f_charInfo=chardet.detect(f_read)
f_charInfo的输出是这样的的一个字典{'confidence': 0.99, 'encoding': 'utf-8'}
```

前面‘confidence’是置信概率，后面是推断出的编码方式。以上的结果，意思为推断这段字符串的编码方式为'utf-8'的概率为99%.

经过我的测试，如果文件里的字符串比较少的话，chardet模块是比较难判断出正确的编码模式的，体现在置信概率比较小，字符串多的话，概率会大。我觉得至少大于90%才可信。

得到编码方式后，就可以用来解码了。

f_read_decode=f_read.decode(f_charInfo['encoding'])

print(f_read_decode)

最后补充一下chardet的安装方法：

pip install chardet

pip的安装方法请自行百度。

标签: python, 文件, 读取字符串

来自 <https://www.cnblogs.com/ArsenalfanInECNU/p/4811643.html>