Python 的编码（encode）与解码（decode）

由于，P3 的 string 均为 unicode 编码，因此在做 encode/decode 转换时，会以 unicode 作为中间编码，即：先将其他编码的字符串解码（decode）成 unicode，再从 unicode 编码（encode）成另一种编码。

编码（encode）：将 unicode str 转换为特定编码格式的 bytecode 并存储，例如：将 unicode str1 转换成 gb2312 bytecode。

解码（decode）：将特定编码格式的 bytecode 转换为 unicode str 的过程，例如：将 gb2312 bytecode 换成 unicode str2。

举例来说：

当我们用 VIM 编辑器打开一个 .py 文件，输入代码 a = 123，那么这个 a = 123 就是一个 unicode str。当我们保存文件后，这个 str 就会根据 VIM 的设置被转换为对应的编码格式（e.g. utf8）的 bytecode 保存到系统的硬盘，这是一个 encode 过程；

然后，当 Python 解释器执行 .py 文件时，先将 bytecode 按照指定的编码格式 decode 为 unicode str，然后运行程序，这是一个 decode 过程。

>>> '美丽人生'.encode('gbk')

b'\xc3\xc0\xc0\xf6\xc8\xcb\xc9\xfa'

>>> b'\xc3\xc0\xc0\xf6\xc8\xcb\xc9\xfa'.decode('gbk')

'美丽人生'

>>> '美丽人生'.encode('utf-8')

b'\xe7\xbe\x8e\xe4\xb8\xbd\xe4\xba\xba\xe7\x94\x9f'

>>> b'\xe7\xbe\x8e\xe4\xb8\xbd\xe4\xba\xba\xe7\x94\x9f'.decode('utf-8')

'美丽人生'

>>> b'\xc3\xc0\xc0\xf6\xc8\xcb\xc9\xfa'.decode('gbk').encode('utf-8')

b'\xe7\xbe\x8e\xe4\xb8\xbd\xe4\xba\xba\xe7\x94\x9f'

上述的 b’str’ 即为 bytecode，一个斜杠就是一个 byte。可见，一个常用汉字用 GBK 格式编码后占 2byte，用 UTF-8 格式编码后占 3byte。

在某些 Terminal 或 Console 中，String 的输出总是出现乱码，甚至错误，其实是由于 Terminal 或 Console 自身不能 decode 该 encode 类型的 string。

[**Python：编码与解码和转义字符**](https://www.cnblogs.com/qi-yuan-008/p/12933939.html)

Python里面的字符一般默认为Unicode字符，属于str类型；而编码则是将字符转换成字节流的过程，反之解码是将字节流解析为字符的过程。

所谓的字节流，在pyhton里面，属于bytes类型。

1. python有两种方式实现编码和解码：

* encode 和 decode 方法：encode()函数根据括号内的编码方式，把str类型的字符串转换为bytes字符串，decode反之。

[复制代码](javascript:void(0);)

b = '龙'.encode('utf-8') # '龙'：unicode字符，encode进行 utf-8 编码

print('b:',b) # 开头的 b 字符表示bytes类型

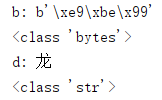
print(type(b)) #类型是'bytes'

d = b.decode('utf-8') #解码

print('d:',d)

print(type(d))

[复制代码](javascript:void(0);)



* str 和 bytes 方法

[复制代码](javascript:void(0);)

ss = bytes('龙', encoding='utf-8') # 编码

print('ss:',ss)

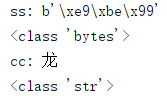
print(type(ss))

cc = str(ss, encoding='utf-8') # 解码

print('cc:',cc)

print(type(cc))

[复制代码](javascript:void(0);)



求编码的字节长度，可以通过 len() 函数 对bytes类型的编码求长度得到：

print(len(b)) #编码的字节数，3个字节编码，因为b就是字节类型 bytes，此时为b：b'\xe9\xbe\x99'

其中，[Unicode与UTF-8的关系可见](https://www.cnblogs.com/qi-yuan-008/p/12926547.html)。

2. bytes函数：class bytes([source[, encoding[, errors]]]) —— 返回一个新的 bytes 对象，该对象是一个 0 <= x < 256 区间内的整数不可变序列。

[**参数**](https://www.runoob.com/python3/python3-func-bytes.html)

* 如果 source 为整数，则返回一个长度为 source 的初始化数组；
* 如果 source 为字符串，则按照指定的 encoding 将字符串转换为字节序列；
* 如果 source 为可迭代类型，则元素必须为[0 ,255] 中的整数；
* 如果 source 为与 buffer 接口一致的对象，则此对象也可以被用于初始化 bytearray。
* 如果没有输入任何参数，默认就是初始化数组为0个元素。

[复制代码](javascript:void(0);)

ss = bytes('龙', encoding='utf-8') # 字符串

print('ss:',ss)

print(type(ss))

s2 = bytes(12) # 整数

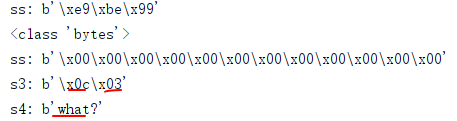
print('ss:',s2)

s3 = bytes([12,3]) # 可迭代类型，元素必须为[0 ,255] 中的整数；

print('s3:',s3)

s4 = bytes('what?', encoding='utf-8') # 字符串，英文  
print('s4:',s4)

[复制代码](javascript:void(0);)



3. 乱码：因为字符串等都是经过某一编码方式编码的，所以如果解码的时候并不是对应的编码方式，则很容易出现乱码；此时就需要使用对应的编码方式解码。

（1）如果结果是一堆bytes类型的：'\x..\x..\x..'，则可以直接decode，通过相应的编码方式，例如decode('utf-8')

（2）如果是由于解码和编码的方法不一致，则反向先编码再解码即可，例如：

[复制代码](javascript:void(0);)

nn = '龙在天下'.encode('gbk')

mm = nn.decode('ISO-8859-1')

print('mm:',mm) #乱码

kk = mm.encode('ISO-8859-1').decode('gbk') # 对乱码反向解码

print('kk:',kk)

[复制代码](javascript:void(0);)

https://img2020.cnblogs.com/blog/1468269/202005/1468269-20200521220744272-1175105762.png

不过一般情况下，会直接报错，例如gbk和utf-8混用时会出现：UnicodeDecodeError: 'utf-8' codec can't decode byte 0xc1 in position 0: invalid start byte。

最重要的是：编码和解码的方法最好一致。

4. 转义字符：遇到特殊字符的时候需要用到转义字符，就像不能直接用class来做变量一样，python中的转义字符主要如下：

| **转义字符** | **描述** |
| --- | --- |
| \(在行尾时) | 续行符 |
| \\ | 反斜杠符号 |
| \' | 单引号 |
| \" | 双引号 |
| \a | 响铃 |
| \b | 退格(Backspace) |
| \e | 转义 |
| \000 | 空 |
| \n | 换行 |
| \v | 纵向制表符 |
| \t | 横向制表符 |
| \r | 回车 |
| \f | 换页 |
| \oyy | 八进制数，yy代表的字符，例如：\o12代表换行 |
| \xyy | 十六进制数，yy代表的字符，例如：\x0a代表换行 |
| \other | 其它的字符以普通格式输出 |

所以上面经常遇到类似：b'\xe9\xbe\x99'，其中 b 表示bytes类型，\x 其实表示十六进制值。