2.9 将Unicode文本标准化¶

问题¶

你正在处理Unicode字符串，需要确保所有字符串在底层有相同的表示。

解决方案¶

在Unicode中，某些字符能够用多个合法的编码表示。为了说明，考虑下面的这个例子：

```
>>> s1 = 'Spicy Jalape\u00f1o'
>>> s2 = 'Spicy Jalapen\u0303o'
>>> s1
'Spicy Jalapeño'
>>> s2
'Spicy Jalapeño'
>>> s1 == s2
False
>>> len(s1)
14
>>> len(s2)
15
>>>
```

这里的文本”Spicy Jalapeño”使用了两种形式来表示。 第一种使用整体字符”ñ”(U+00F1)，第二种使用拉丁字母”n”后面跟一个”~”的组合字符(U+0303)。

在需要比较字符串的程序中使用字符的多种表示会产生问题。 为了修正这个问题，你可以使用unicodedata模块先将文本标准化：

```
>>> import unicodedata
>>> t1 = unicodedata.normalize('NFC', s1)
>>> t2 = unicodedata.normalize('NFC', s2)
>>> t1 == t2
True
>>> print(ascii(t1))
'Spicy Jalape\xf1o'
>>> t3 = unicodedata.normalize('NFD', s1)
>>> t4 = unicodedata.normalize('NFD', s2)
>>> t3 == t4
True
>>> print(ascii(t3))
'Spicy Jalapen\u0303o'
>>>
```

`normalize()` 第一个参数指定字符串标准化的方式。 NFC表示字符应该是整体组成(比如可能的话就使用单一编码)，而NFD表示字符应该分解为多个组合字符表示。

Python同样支持扩展的标准化形式NFKC和NFKD，它们在处理某些字符的时候增加了额外的兼容特性。比如：

```
>>> s = '\ufb01' # A single character
>>> s
'fi'
>>> unicodedata.normalize('NFD', s)
'fi'
# Notice how the combined letters are broken apart here
>>> unicodedata.normalize('NFKD', s)
'fi'
>>> unicodedata.normalize('NFKC', s)
'fi'
>>>
```

讨论¶

标准化对于任何需要以一致的方式处理Unicode文本的程序都是非常重要的。 当处理来自用户输入的字符串而你很难去控制编码的时候尤其如此。

在清理和过滤文本的时候字符的标准化也是很重要的。 比如，假设你想清除掉一些文本上面的变音符的时候(可能是为了搜索和匹配)：

```
>>> t1 = unicodedata.normalize('NFD', s1)
>>> ''.join(c for c in t1 if not unicodedata.combining(c))
'Spicy Jalapeno'
>>>
```

最后一个例子展示了 `unicodedata` 模块的另一个重要方面，也就是测试字符类的工具函数。 `combining()` 函数可以测试一个字符是否为和音字符。 在这个模块中还有其他函数用于查找字符类别，测试是否为数字字符等等。

Unicode显然是一个很大的主题。如果想更深入的了解关于标准化方面的信息， 请看考 Unicode官网中关于这部分的说明 Ned Batchelder在 他的网站 上对Python的Unicode处理问题也有一个很好的介绍。