2.10 在正则式中使用Unicode¶

问题¶

你正在使用正则表达式处理文本，但是关注的是Unicode字符处理。

解决方案¶

默认情况下 `re` 模块已经对一些Unicode字符类有了基本的支持。 比如， `\\d` 已经匹配任意的unicode数字字符了：

```
>>> import re
>>> num = re.compile('\d+')
>>> # ASCII digits
>>> num.match('123')
<_sre.SRE_Match object at 0x1007d9ed0>
>>> # Arabic digits
>>> num.match('\u0661\u0662\u0663')
<_sre.SRE_Match object at 0x101234030>
>>>
```

如果你想在模式中包含指定的Unicode字符，你可以使用Unicode字符对应的转义序列(比如 `\uFFF` 或者 `\UFFFFFFF`)。 比如，下面是一个匹配几个不同阿拉伯编码页面中所有字符的正则表达式：

```
>>> arabic = re.compile('[\u0600-\u06ff\u0750-\u077f\u08a0-\u08ff]+')
>>>
```

当执行匹配和搜索操作的时候，最好是先标准化并且清理所有文本为标准化格式(参考2.9小节)。 但是同样也应该注意一些特殊情况，比如在忽略大小写匹配和大小写转换时的行为。

```
>>> pat = re.compile('stra\u00dfe', re.IGNORECASE)
>>> s = 'straße'
>>> pat.match(s) # Matches
<_sre.SRE_Match object at 0x10069d370>
>>> pat.match(s.upper()) # Doesn't match
>>> s.upper() # Case folds
'STRASSE'
>>>
```

讨论¶

混合使用Unicode和正则表达式通常会让你抓狂。 如果你真的打算这样做的话，最好考虑下安装第三方正则式库， 它们会为Unicode的大小写转换和其他大量有趣特性提供全面的支持，包括模糊匹配。