17.正则表达式

一、正则表达式

正则表达式（RE）为高级文本模式匹配，以及搜索-替代等功能提供了基础。
正则表达式是一些由字符和特殊符号组成的字符串，他们描述了这些字符和字符的某种重复方式，因此能按某种模式匹配一个有相似特征的字符串集合，因此能按某种模式匹配一系列有相似特征的字符串。

二、正则表达式使用到的元字符（常用的）

https://www.runoob.com/regexp/regexp-metachar.html

符号	说明	用法示例
literal	匹配字符串的值(字面值)	foo
|	匹配前面或后面正则表达式	foo|bar
.	匹配任何一个字符（换行符除外）	b.b
^	匹配字符串的开始	^abc
$	匹配字符串的结尾	html$
*	匹配前面出现的正则表达式零次或多次， 贪婪模式（尽可能多的匹配）	a*
+	匹配前面出现的正则表达式一次或多次,贪婪模式	a+
?	匹配前面出现的正则表达式零次或一次，贪婪模式	abc?
{N}	匹配前面出现的正则表达式N次	a{3}
{M, N}	匹配重复出现M到N次的正则表达式	a{2, 5}
[....]	匹配字符组里面出现的任意一个字符	[abcde]
[x-y]	匹配从字符x到字符y中的任意一个字符	[0-9], [a-z]
[^...]	不匹配此字符集中出现的任何一个字符，包括某一范围的字符	[^abc], [^a-z]
*?, +?, ??	用于上面出现的任何非贪婪模式(尽可能少匹配)	.*?abc
(...)	匹配封闭括号中的正则表达式，并保存为子组	([0-9]{3})?, f(oo|u)bar
(?:...)	匹配括号中的正则表达式...但是不创建分组	(?:abc)
(?iLmsux)	iLmsux代表正则表达式的一个模式，详见下面的flag解析，只能用在正则表达式的开头	(?is).\w+?
特殊字符	**说明**	**用法示例**
\d	匹配任何数字，和[0-9]一样， \D则是匹配任何非数字	\d+?
\w	匹配任何数字和字母，和[A-Za-z0-9]相同， \W则相反	foo\w{3}?
\s	匹配任何空白字符，和[\n\t\r\v\f]相同， \S则相反	hello\s?world

符号	说明	用法示例
\b	匹配单词边界，\B则相反	\bthe\b
\nn	匹配已保存的子组，请参考上面的(...)	price:\16
\c	逐一匹配特殊字符c，即取消它的特殊意义，按字面匹配	., \?, \
\A(\Z)	匹配字符串的起始（结束）	\Apython

二、正则表达式使用案例

1. 关键词筛选
2. 字符替换

三、python中使用正则表达式常用函数和方法

在python中使用正则表达式需要导入正则表达式模块（re）这个是python内置的模块，因此不需要安装，但是需要注意的是我们给文件命名的时候不要使用这个名字，否则就会造成模块名称冲突导致用不了。

函数/方法	说明
模块的函数	
compile(pattern, flags=0)	对正则表达式模式pattern进行编译， flags是可选标识符，函数返回一个正则表达式对象（regex）
re模块的函数和对象方法（regex）	
match(pattern, string, flags=0)	尝试用正则表达式模式pattern匹配字符串string，flags是可选标识符，如果匹配成功，则返回一个匹配对象，否则返回None，注意match是从头开始匹配的，就是匹配开始的位置
search(pattern, string, flags=0)	在字符串string中搜索正则表达式模式pattern的第一次出现，flags是可选标识符，如果匹配成功，则返回一个匹配对象，否则返回None
findall(pattern, string, flags=0)	在字符串string中搜索正则表达式模式pattern的所有出现；返回一个匹配对象的列表，匹配不到返回空列表
finditer(pattern, string, flags=0)	在字符串string中搜索正则表达式模式pattern的所有出现；与findall不同的是该方法返回一个可迭代对象
split(pattern, string, maxsplit=0, flags=0)	根据正则表达式pattern中的分隔符把字符串string分割为一个列表，返回成功匹配的列表，最多分割maxsplit次（默认是分割所有匹配的地方）
sub(pattern, repl, string, count=0, flags=0)	把字符串string中所有匹配正则表达式pattern的地方替换成字符串repl， 如果count的值没有给出，则对所有匹配的地方进行替换，返回替换后的字符串
subn(pattern, repl, string, count=0, flags=0)	把字符串string中所有匹配正则表达式pattern的地方替换成字符串repl， 如果count的值没有给出，则对所有匹配的地方进行替换，返回的是一个双元素元组，元组的第一个元素是替换后的新字符串，第二个元素是替换的数量
匹配对象的方法	
group(num=0)	返回全部匹配对象（或指定编号是num的子组）
groups()	返回一个包含全部匹配子组的元组（如果没有成功匹配，就返回一个空元组）

re中的flag参数及其含义

1. 忽略大小写（常用）
   I = IGNORECASE = sre_compile.SRE_FLAG_IGNORECASE
2. \w, \W, \b, \B等是否生效取决于当前的系统环境（其实没啥用）
   L = LOCALE = sre_compile.SRE_FLAG_LOCALE
3. 匹配Unicode字符串，主要是对与非ASCII码字符串来讲的，因为python2默认的字符串都是ASCII编码的，所以模式\w+能匹配的都是ASCII字符，要想让\w+匹配Unicode字符，就可以设置这个flag
   U = UNICODE = sre_compile.SRE_FLAG_UNICODE
4. 多行匹配，主要就是当匹配行首(^)或行尾($)的时候，如果不使用多行匹配的话，对于多行的文本是不能匹配成功的
   M = MULTILINE = sre_compile.SRE_FLAG_MULTILINE
5. 让点号（.）也代表换行（常用）
   S = DOTALL = sre_compile.SRE_FLAG_DOTALL
6. 忽略表达式模式中的空白字符和注释
   X = VERBOSE = sre_compile.SRE_FLAG_VERBOSE

四、python正则表达式的使用案例

具体看视频，以实际提取网页数据为案例讲解

1. compile方法的使用
   compile方法是预先编译正则表达式的匹配模式，然后生成一个缓存，这样在之后的匹配中就可以直接使用这个缓存，而不需要每次匹配都重新编译，从而加快速度。一般情况下只有当一个正则表达式被重复使用多次的时候才需要事先使用compile方法编译，如果只是使用一次就不需要了。
2. match方法的使用
3. search方法的使用
4. findall方法的使用
5. finditer方法的使用
6. sub方法的使用
7. subn方法的使用

作业

1. 熟练使用正则表达式提取各种信息
2. 记住常用的正则表达式元字符，知道贪婪和非贪婪模式的区别
3. 使用python的正则表达式模块re结合前面的爬虫知识来提取一下站点的文章

> http://www.tvix.cn/

4. 熟练使用正则表达式模块re的各种常用方法，以及他们之间的区别和用途。