**Python3正则表达式(一)基本语法规则**

正则表达式是一个很强大的字符串处理工具，它能帮助我们方便的检查一个字符串是否与某种模式匹配。

**一、python中正则表达式的基本语法规则**

| **序号** | **语法** | **解释** | **表达式** | **成功匹配对象** |
| --- | --- | --- | --- | --- |
| **1** | 一般字符 | 匹配自身相对应的字符 | abc | abc |
| **2** | . | 匹配除换行符(\n)以外的任意字符 | a.c | abc |
| **3** | \ | 转义字符，可以改变原字符的意思 | a\.c | a.c |
| **4** | \d | 匹配数字:0~9 | \dabc | 1abc |
| **5** | \w | 匹配单词字符,a~z;A~Z;0~9 | \w\w\w | oX2 |
| **6** | \s | 匹配空格字符(\t,\n,\r,\f,\v) | a\sc | a c |
| **7** | \D | 匹配非数字字符 | \Dabc | aabc |
| **8** | \W | 匹配非单词字符 | a\Wc | a c |
| **9** | \S | 匹配非空格字符 | \S\Sc | 1bc |
| **10** | [] | 字符集，对应位置上可以是字符集里的任意字符 | a[def]c | aec |
| **11** | [^] | 对字符集当中的内容进行取反 | a[^def]c | a2c |
| **12** | [a-z] | 指定一个范围字符集 | a[A-Z]c | aBc |
| **13** | \* | 允许前一个字符可以出现0次或者无限次 | a\*b | aaab或b |
| **14** | + | 前一个字符至少出现1次 | a+b | aaab或ab |
| **15** | ? | 前一个字符只能出现一次或者不出现 | a?b | ab或b |
| **16** | {m} | 允许前一个字符只能出现m次 | a{3}b | aaab |
| **17** | {m,n} | 允许前一个字符至少出现m次,最多出现n次(如果不写n，则代表至少出现m次) | a{3,5}b和a{3,} | aaaab和aaaaaab |
| **18** | ^ | 匹配字符串的开始，多行内容时匹配每一行的开始 | ^abc | abc |
| **19** | $ | 匹配字符串的结尾，多行内容时匹配每一行的结尾 | abc& | abc |
| **20** | \A | 匹配字符串开始位置，忽略多行模式 | \Aabc | abc |
| **21** | \Z | 匹配字符串结束位置，忽略多行模式 | abc\Z | abc |
| **22** | \b | 匹配位于单词开始或结束位置的空字符串 | hello \bworld | hello world |
| **23** | \B | 匹配不位于单词开始或结束位置的空字符串 | he\Bllo | hello |
| **24** | | | 表示左右表达式任意满足一种即可 | abc|cba | abc或cba |
| **25** | (…) | 将被括起来的表达式作为一个分组，可以使用索引单独取出 | (abc)d | abcd |
| **26** | (?P<name>…) | 为该分组起一个名字，可以用索引或名字去除该分组 | (?P<id>abc)d | abcd |
| **27** | \number | 引用索引为number中的内容 | (abc)d\1 | abcdabc |
| **28** | (?P=name) | 引用该name分组中的内容 | (?P<id>abc)d(?P=id) | abcdabc |
| **29** | (?:…) | 分组的不捕获模式，计算索引时会跳过这个分组 | (?:a)b(c)d\1 | abcdc |
| **30** | (?iLmsux) | 分组中可以设置模式，iLmsux之中的每个字符代表一个模式,单独介绍 | (?i)abc | Abc |
| **31** | (?#…) | 注释，#后面的内容会被忽略 | ab(?#注释)123 | ab123 |
| **32** | (?=…) | 顺序肯定环视，表示所在位置右侧能够匹配括号内正则 | a(?=\d) | a1最后的结果得到a |
| **33** | (?!…) | 顺序否定环视，表示所在位置右侧不能匹配括号内正则 | a(?!\w) | a c最后的结果得到a |
| **34** | (?<=…) | 逆序肯定环视，表示所在位置左侧能够匹配括号内正则 | 1(?<=\w)a | 1a |
| **35** | (?<!…) | 逆序否定环视，表示所在位置左侧不能匹配括号内正则 | 1 (?<!\w)a | 1 a |
| **36** | (?(id/name)yes|no) | 如果前面的索引为id或者名字为name的分组匹配成功则匹配yes区域的表达式，否则匹配no区域的表达式，no可以省略 | (\d)(?(1)\d|a) | 32 |

**二、(?iLmsux)为分组设置模式**

这里的”i”, “L”, “m”, “s”, “u”, “x”，它们不匹配任何字串，而是表示对应python中re模块当中的(re.I, re.L, re.M, re.S, re.U, re.X)的6种选项。   
可以在python源码中看到：

I = IGNORECASE # 忽略大小写

L = LOCALE # 字符集本地化,为了支持多语言版本的字符集使用环境

U = UNICODE # 使用\w,\W,\b,\B这些元字符时将按照UNICODE定义的属性

M = MULTILINE # 多行模式,改变 ^ 和 $ 的行为

S = DOTALL # '.' 的匹配不受限制,包括换行符

X = VERBOSE # 冗余模式,可以忽略正则表达式中的空白和#号的注释

* 1
* 2
* 3
* 4
* 5
* 6

六种模式在正则表达式中可以同时使用多个的，在 python 里面使用按位或运算符 | 同时添加多个模式   
如：re.compile(”, re.I|re.M|re.S)

**三 、反斜杠的使用**

在一般的编程语言当中，反斜杠“\”代表反转义字符，在反斜杠后面加一个字符可以表示一种特定的意思，接下来列举几个常见的转义字符：

| **序号** | **转义字符** | **解释** |
| --- | --- | --- |
| 1 | \f | 换页,将当前位置移到下页开头 |
| 2 | \n | 换行,将当前位置移到下一行开头 |
| 3 | \r | 回车,将当前位置移到本行开头 |
| 4 | \t | 水平制表,跳到下一个TAB位置 |
| 5 | \v | 垂直制表(暂时还没用过这个) |
| 6 | \\ | 代表一个反斜杠字符 ‘\’ |

因为在正则表达式的规则当中，‘\’就是转义字符的意思，前面基本语法规则里也有说到，但是在一般的变成语言中，’\’也有转义字符的意思，所以，如果我们要是写代码的时候用到正则表达式中的’\’的时候，就需要写四个’\’，才可以代表一个真正的反斜杠字符，如:“\\\\”。

“\\\\”：这里的第一个和第三个是在编程语言中起转义的作用，结果：“\\”，然后在正则表达式中它在进行一次反转义就代表着一个真正的反斜杠字符了。

但是在python语言中，我们可以不考虑这个问题。只需要在写好的字符串前面加一个’r’， 告诉编译器这个字符串是个原生字符串，不要转义’\’ 。

例如，上个例子中的正则表达式可以使用r”\”表示。因为加上这个’r’，只是告诉python语言不转义这个字符串，但是在正则表达式中还是要符合正则表达式的规则。同样，匹配一个数字的”\d”就可以写成r”\d”。所以，在python中写正则表达式时，要养成一个前面写’r’的习惯。

**四、总结**

这些就是正则表达式的基本语法了，和学习编程一样，活学活用，不懂就查。接下来将写一下在python3中使用正则表达式即re模块的学习笔记。

大家可以在写的时候参考一下:[最全的常用正则表达式大全](http://blog.csdn.net/carechere/article/details/52315728)