字符集合（字符组），各个字符之间是逻辑或（多选一）。[]还会将特殊字符变为普通字符，转义字符（如\n）和预定义字符集(如\d)除外。
* -在中括号内表示字符区间（前后必须有字符，码值小在前，码值大在后）。可以同时指定多个字符区间，例如[0-24-5]。注意：-不需要被转义。
* \表示对元字符进行转义，按文本形式处理字符，忽略其特殊意义。方括号[]和花括号{}只需转义开括号，但圆括号()两个都要转义。
* 字符集是一种简写形式，d代表数字（digit），w代表单词（word），s代表空白字符（space）。POSIX派系有单独的字符集，而且必须出现在方括号内，所以必须写成[[:digit:]]。
![image.png](https://p3-juejin.byteimg.com/tos-cn-i-k3u1fbpfcp/d2c554ee9abb4625903bfa57b2874d25~tplv-k3u1fbpfcp-jj-mark:3024:0:0:0:q75.awebp#?w=699&h=520&s=48321&e=png&b=f4f7eb)
* 限定符也称为量词，表示字符或子表达式的重复次数，默认为贪婪模式（greedy，最长匹配，优先匹配，越多越好，尽可能多，多多益善）。限定符后紧跟?表示非贪婪模式，也称为懒惰模式（lazy，最短匹配，优先忽略，越少越好，尽可能少，适可而止）。
* 定位符也称为锚点（anchor）。\b用来匹配一个单词的开头或结尾（boundary），单词边界匹配的是一个位置（一边是单词字符，另一边不是单词字符），而不是任何实际的字符。正则可以匹配文本和位置。
* |表示选择（逻辑或），具有最低优先级。从左往右测试每个条件，如果满足某个条件，就不会管其它条件（优先左边）。注意：[]只表示单字符，|可表示子表达式。a|b|c和[abc]表示一样，ab|bc表示ab或bc。
* ()表示使用子表达式（subexpression）进行分组（将子表达式看成一个整体），每个分组会自动分配一个组号，规则是：从左到右按左括号顺序编号，1为第一个分组的匹配结果，依次类推。反向引用用于引用子表达式（类似变量），\num或$num。(exp)编号捕获、(?exp)和(?Pexp)命名捕获。括号的另一种用途是通过(?#comment)表示注释。
* 断言（assertion）是一个结果为真或假的逻辑判断式。正则断言只用于匹配位置，而不是文本内容本身。常见的断言有三种：单词边界、行的开始/结束、环视。
* 环视（look-around，停在原地四处张望）就是要 求匹配部分的前面或后面要满足（或不满足）某种规则。环视本身不匹配任何字符（不消耗字符），因此也称零宽断言。肯定顺序环视（positive look-ahead）记法为(?=exp)，表示匹配exp前面的位置。 肯定逆序环视（positive look-behind）记法为(?<=exp)，表示匹配exp后面的位置。

> 记忆：=肯定、?否定、<逆序

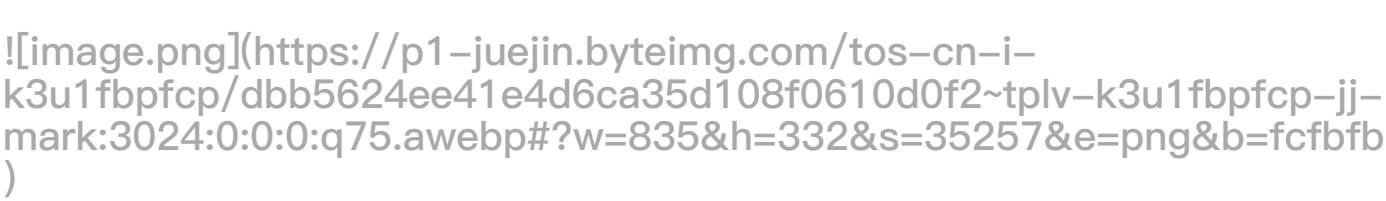
```
('/(?=\d{3})/g', '12345')       肯定顺序环视：右侧必须出现3个数字的位置
```

（0-0、1-1、2-2）
('/(?!\d{3})/g', '12345')　　否定顺序环视：右侧不能出现3个数字的位置
（3-3、4-4、5-5）
('/(?<=\d{3})/g', '12345')　　肯定逆序环视：左侧必须出现3个数字的位置
（3-3、4-4、5-5）
('/(?<!\d{3})/g', '12345')　　否定逆序环视：左侧不能出现3个数字的位置（
* 对于多选分支提取公共部分，并且出现概率大的放左边
* 只在必要时使用分组，尽量使用非捕获分组(?:)。警惕嵌套的分组重复，例如`(.*)*`

正则表达式和通配符
==========

* 正则表达式用于匹配字符串，属于包含匹配，常用命令有grep，awk，sed等。
* 通配符用于匹配文件名或目录名，属于完全匹配，常用命令有find，ls，cp等。

> 通配符和正则表达式看起来有点像，但不能混淆，正则表达式比通配符复杂多。通配符中*可以匹配0个或多个字符，而在正则表达式中*表示重复之前的一个或者多个字符，不能独立使用的。注意：+不是通配符。



参考资料
====

《精通正则表达式》

《正则表达式必知必会》

《正则指引》

《正则表达式入门课》

[正则表达式 - 维基百科](http://cxyroad.com/
"https://zh.wikipedia.org/wiki/%E6%AD%A3%E5%88%99%E8%A1%A
8%E8%BE%BE%E5%BC%8F")

[Regex cheatsheet](http://cxyroad.com/
"https://remram44.github.io/regex-cheatsheet/regex.html")

[正则表达式30分钟入门教程](http://cxyroad.com/
"https://deerchao.cn/tutorials/regex/regex.htm")

[正则表达式 - POSIX & PCRE](http://cxyroad.com/
"https://zhuanlan.zhihu.com/p/435815082")

[觅迹寻踪之正则表达式](http://cxyroad.com/
"https://zhuanlan.zhihu.com/p/434673514")

浅谈正则表达式——从原理到实战

[正则表达式入门课](http://cxyroad.com/
"https://cloud.tencent.com/developer/article/2316659?areaId=106001")

[Linux/Unix 工具与正则表达式的 POSIX 规范](http://cxyroad.com/
"https://www.infoq.cn/article/2011/07/regular-expressions-6-POSIX/")

[刨根究底正则表达式之一：正则表达式概述](http://cxyroad.com/
"https://www.infoq.cn/article/regular-expressions-introduction-
part01")

[命令行通配符教程](http://cxyroad.com/
"http://www.ruanyifeng.com/blog/2018/09/bash-wildcards.html")

原文链接: https://juejin.cn/post/7372393680596647962