

PYTHON BASICS

格式化和正则化

2020-04-30

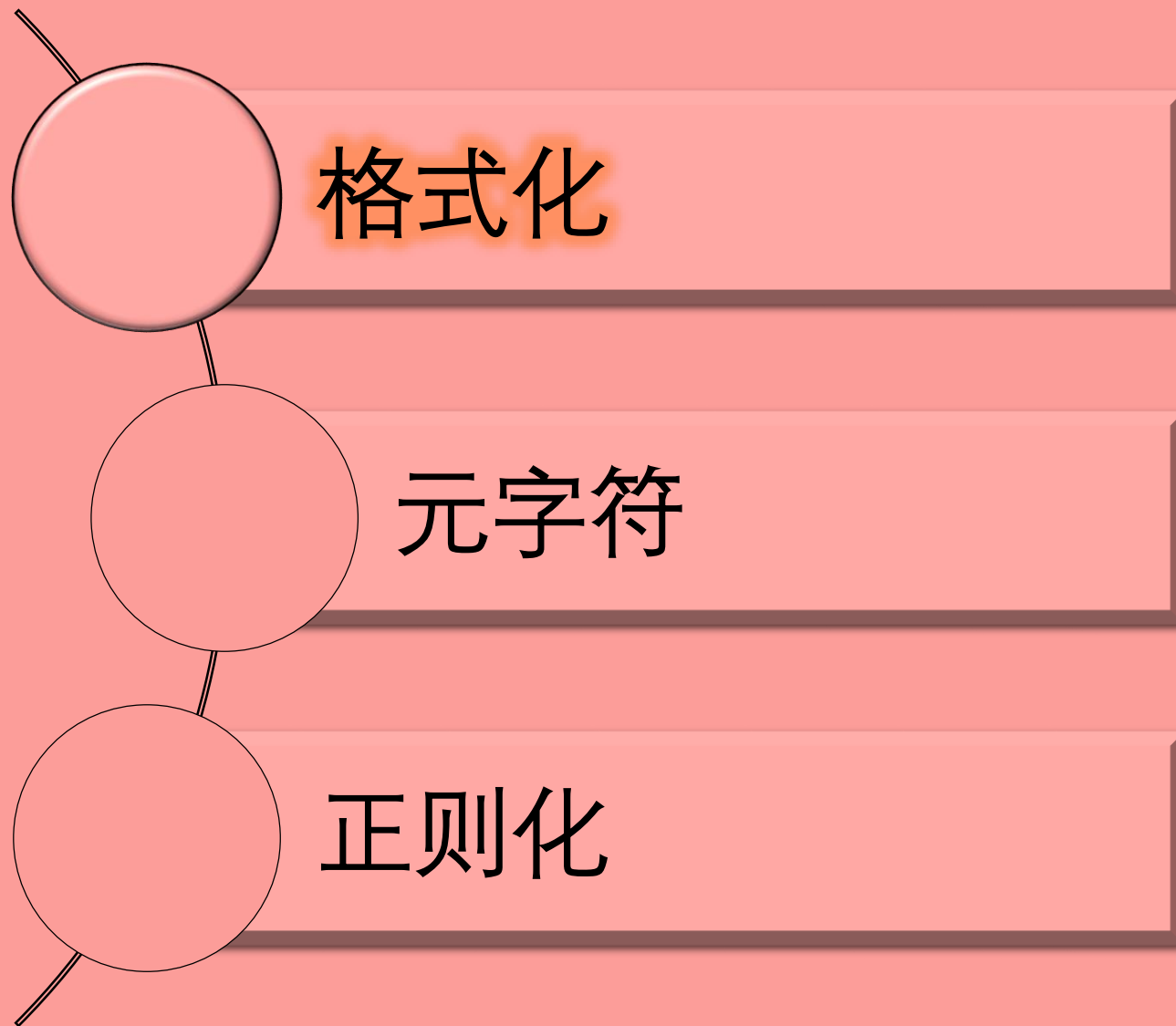
STEVEN WANG



上节总结

特点	含义	伪代码	用处
封装	将属性打包在一起	<code>o.fld, o.mtd()</code>	紧凑
继承	使用父类重复属性 (is-a)	<code>super.__init__()</code>	简洁
多态	将多对象的方法统一	<code>o1.mtd(), o2.mtd()</code>	一致
组合	对象中有其他对象 (has-a)	<code>self.__init__(o)</code>	实际

字符串专场



格式化字符串是以指定**输出参数格式**和**相对位置**来“美化”字符串。

- **输出参数格式**：数字的小数点位数、字符串大小写等。
- **相对位置**：标注出被格式化的字符串是在句中的位置。

```
(name, amount) = ('Steven', 123.456)
```



```
'It costs %s %.2f' %(name, amount)
```

```
'It costs {0:s} {1:.2f}'.format(name, amount)
```

```
f'It costs {name:s} {amount:.2f}'
```

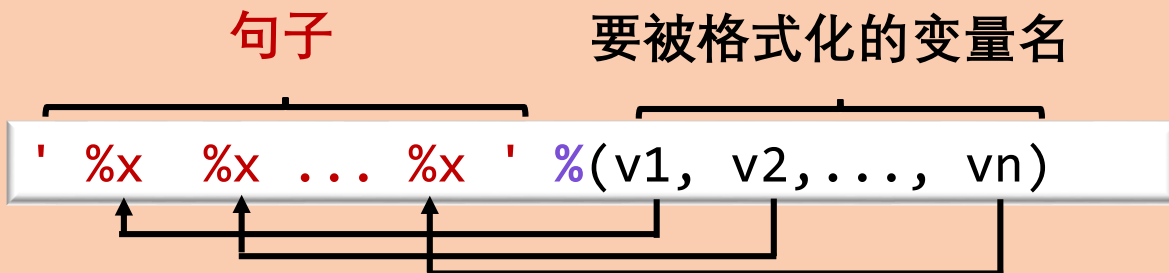


```
'It costs Steven 123.46'
```

% 字符

句子 要被格式化的变量名

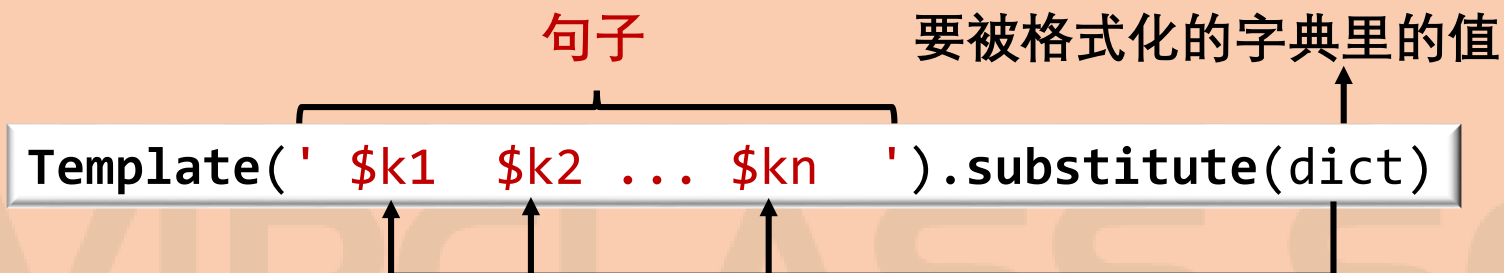
```
' %x %x ... %x ' % (v1, v2, ..., vn)
```



\$ 字符

句子 要被格式化的字典里的值

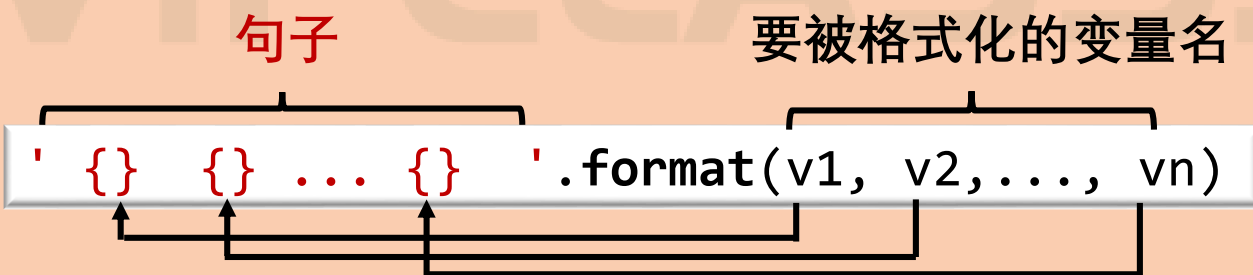
```
Template(' $k1 $k2 ... $kn ').substitute(dict)
```



format 函数

句子 要被格式化的变量名

```
' {} {} ... {} '.format(v1, v2, ..., vn)
```

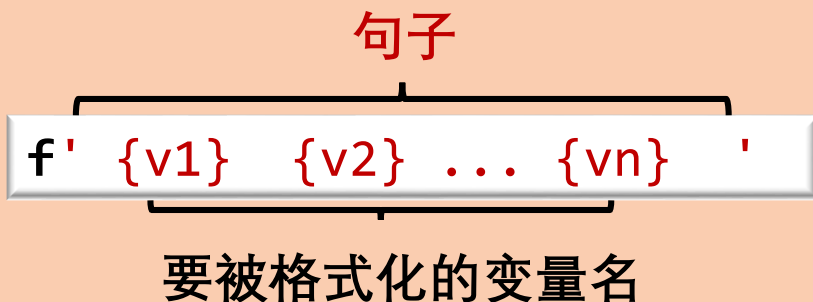


f-string 方法

句子

```
f' {v1} {v2} ... {vn} '
```

要被格式化的变量名



int
float
date

小数点位数
分隔符

{ 被格式化的元素 : 格式化的模式 }

dict.key
List[i]
object.field

个性化日期

范式

`f'{element:pattern}'``f'this is an integer {15:d}'``'this is an integer 15'``f'this is an integer {15:4d}'``'this is an integer 15'`

2 个空格

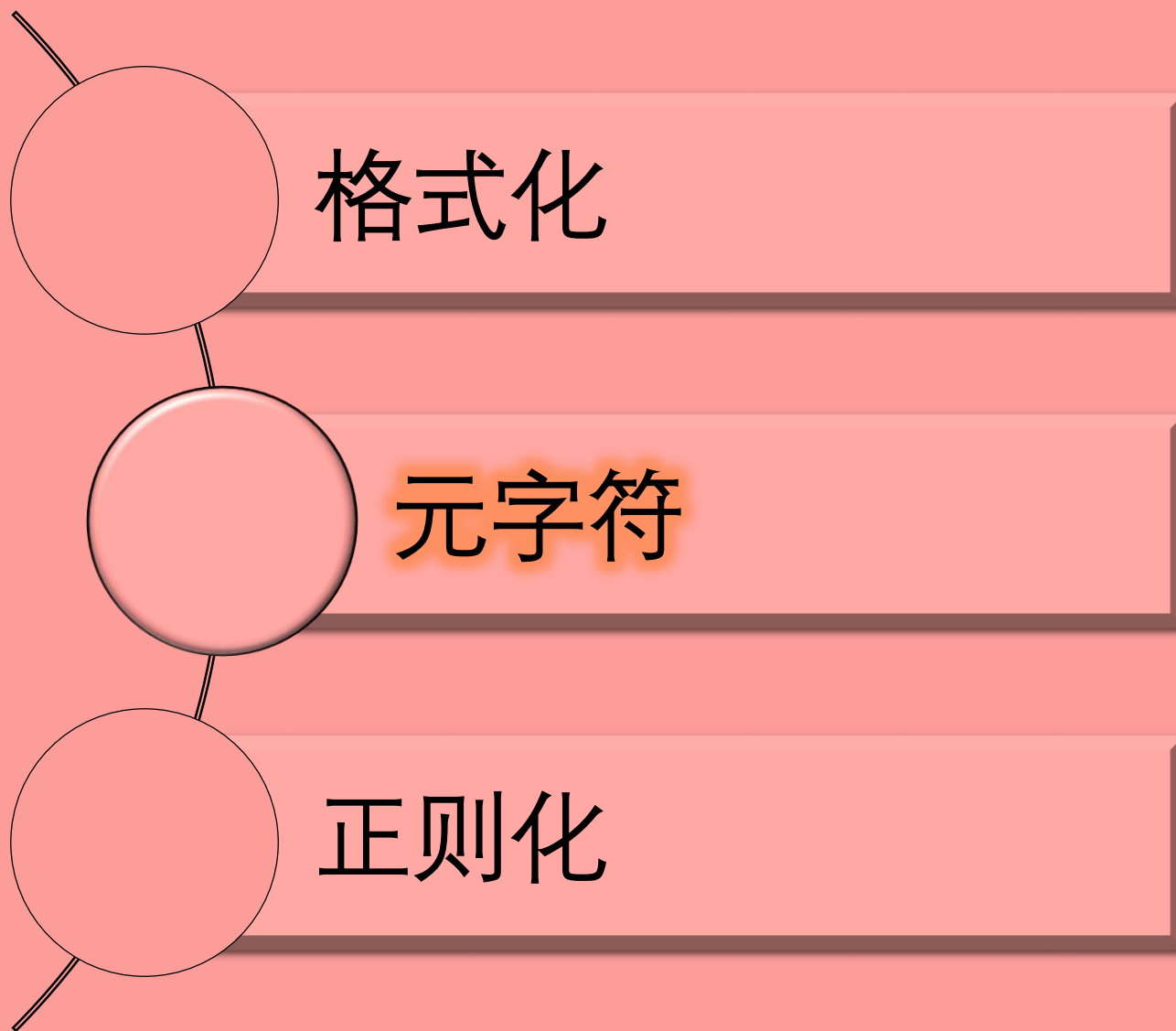
`f'this is an integer {15:04d}'``'this is an integer 0015'``f'this is a float {83.1031:.2f}'``'this is a float 83.10'``f'this is a float {83.1031:8f}'``'this is a float 83.103100'``f'this is a float {83.1031:8.2f}'``'this is a float 83.10'`

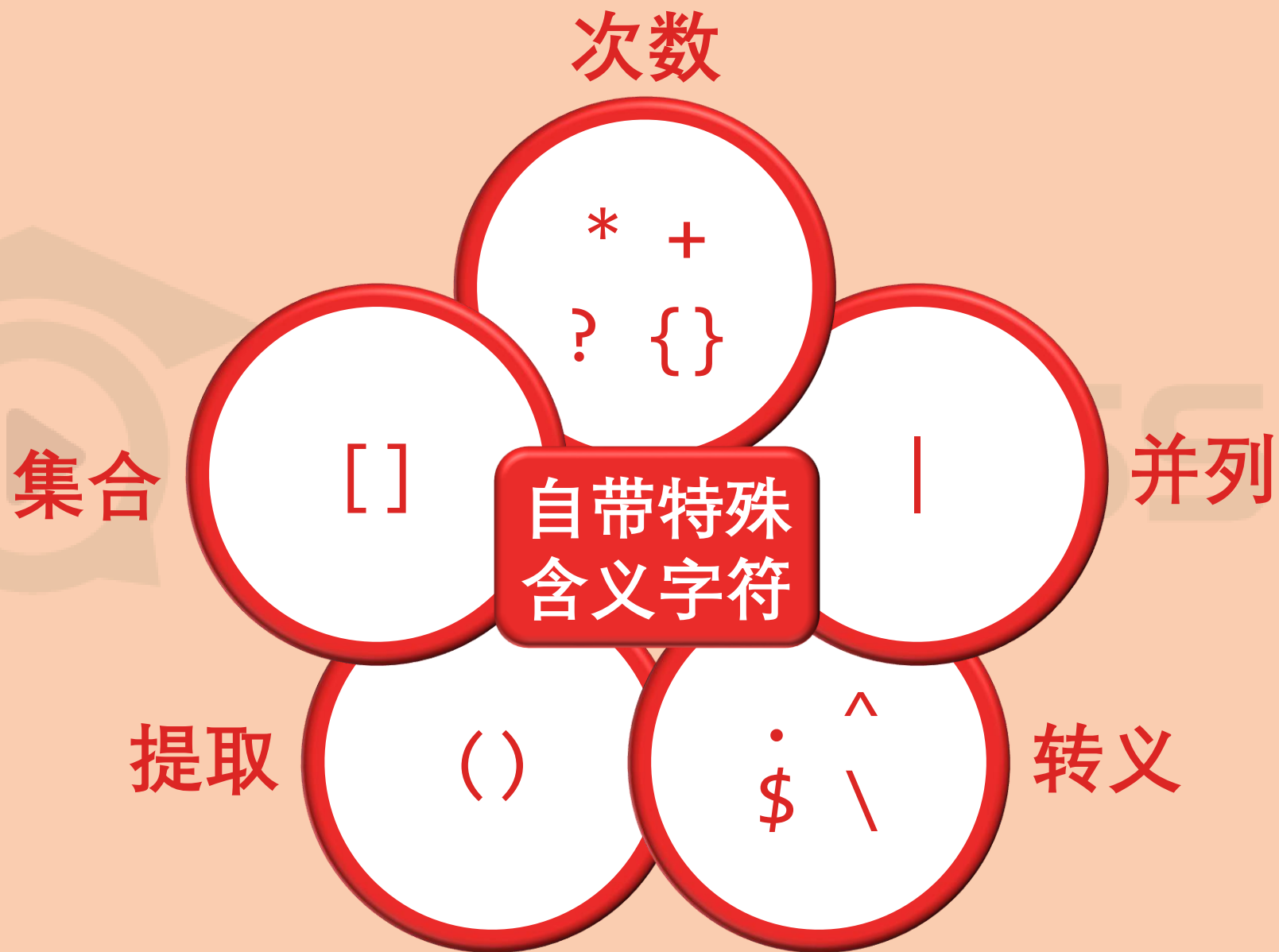
3 个空格

`'this is a string {'love':10s}'``'this is a string love'`

6 个空格

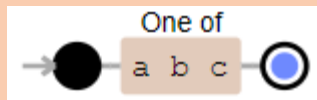
字符串专场





集合是用**中括号** **[]** 表示一个字符集，即创建的模式匹配中括号里指定字符集中的任意一个字符，有三种方式来表现：

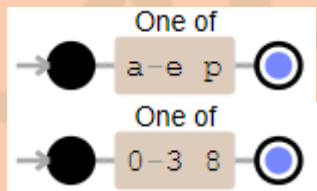
- **明确**字符：**[abc]** 匹配字符 a, b 或 c



One of

- **范围**字符：用 - 符号

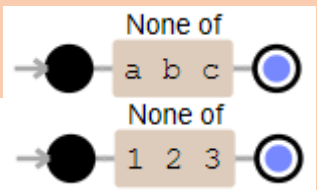
- **[a-e p]** 匹配字符 a 到 e 加上 p
- **[0-3 8]** 匹配字符 0 到 3 加上 8



One of

- **补集**字符：用 ^ 符号

- **[^abc]** 匹配除 a, b, c 以外的字符
- **[^123]** 匹配除 1, 2, 3 以外的字符

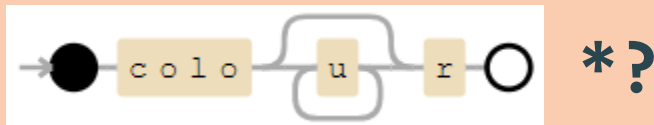


None of

只能匹配一个字符

贪心模式

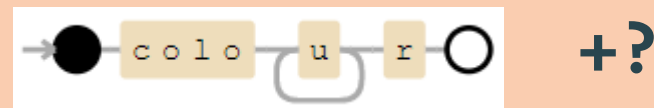
$*$ = 零个或多个字符



$*?$

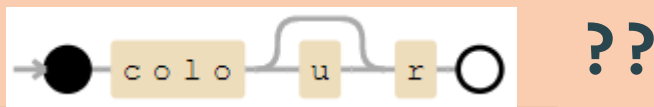
非贪心模式

$+$ = 一个或多个字符



$+?$

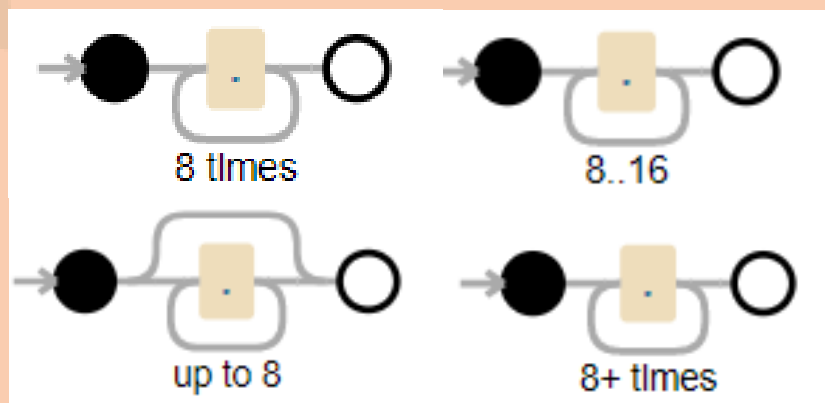
$?$ = 零或一个字符



$??$

$\{n\}$

出现 n 次



$\{n,m\}$

出现在 n 次和 m 次之间

$\{,n\}$

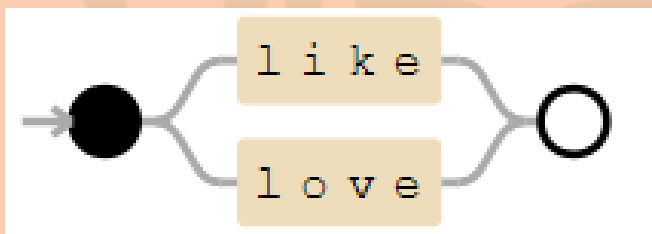
出现小于等于 n 次

$\{n,\}$

出现大于等于 n 次

不能并行匹配

并列用垂线 | 字符，举例 $A|B$ ，如果 A 匹配了，则不再查找 B，反之亦然。

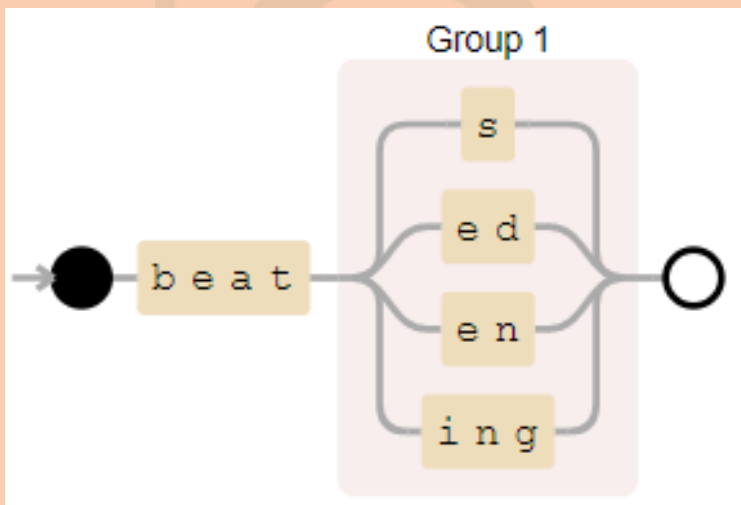


并行
通路

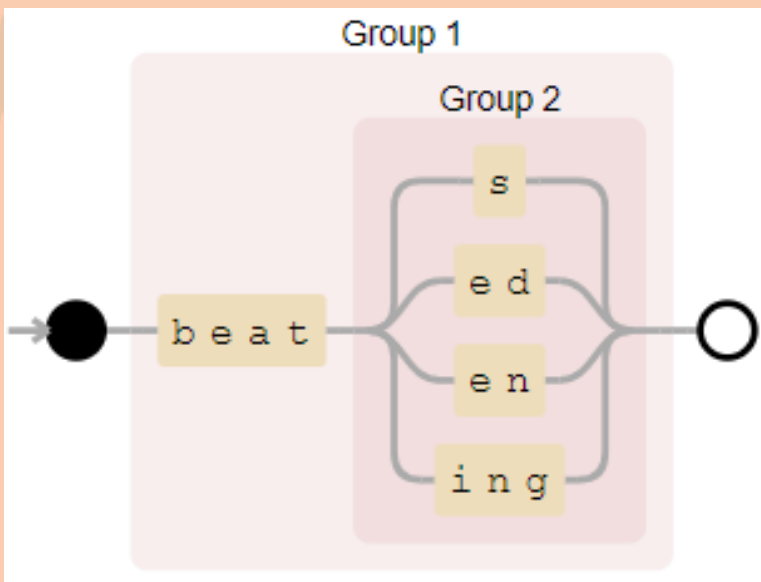
匹配后不能提取

用小括号 () 可提取已匹配的内容，在小括号里可设计任意正则表达式。

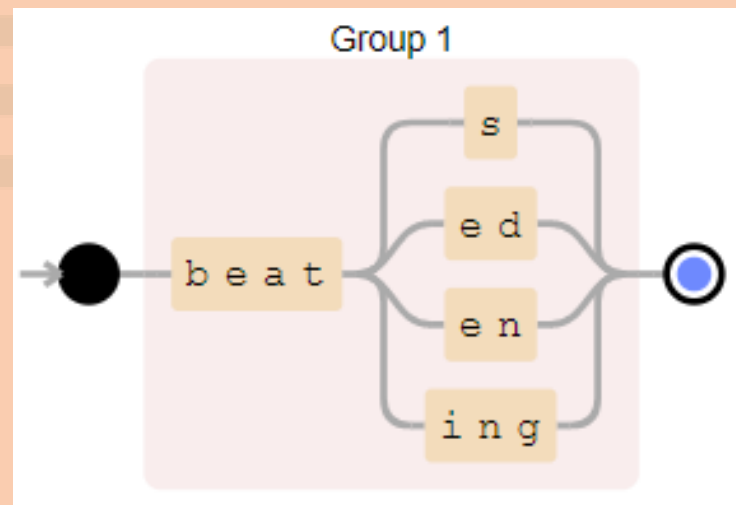
`beat(s|ed|en|ing)`



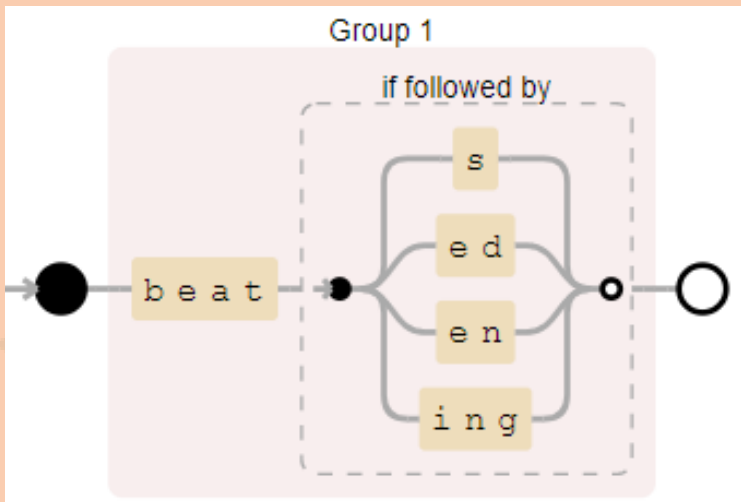
`(beat(s|ed|en|ing))`



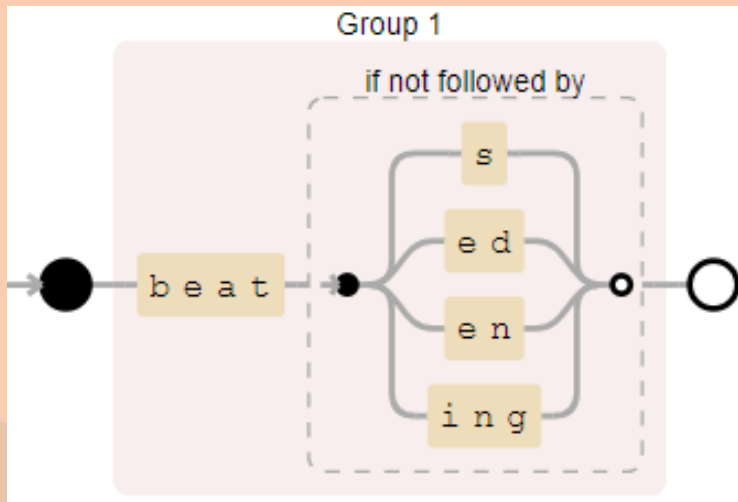
只匹配不获取
↑
`(beat(?:s|ed|en|ing))`



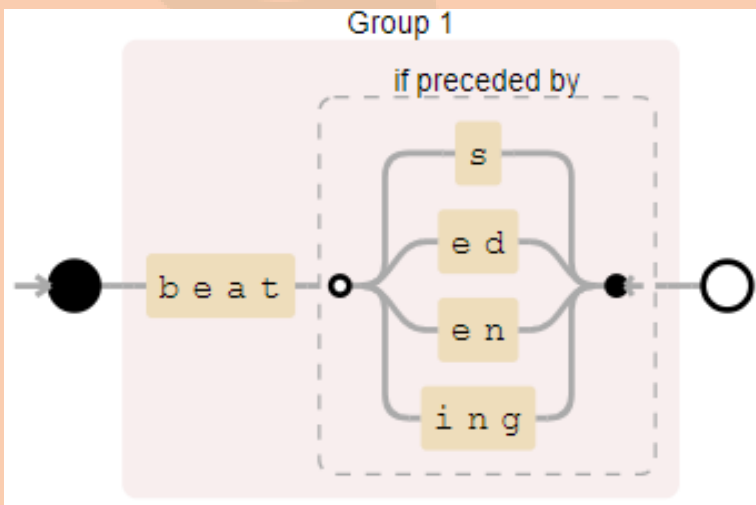
`(beat(?=s|ed|en|ing))`



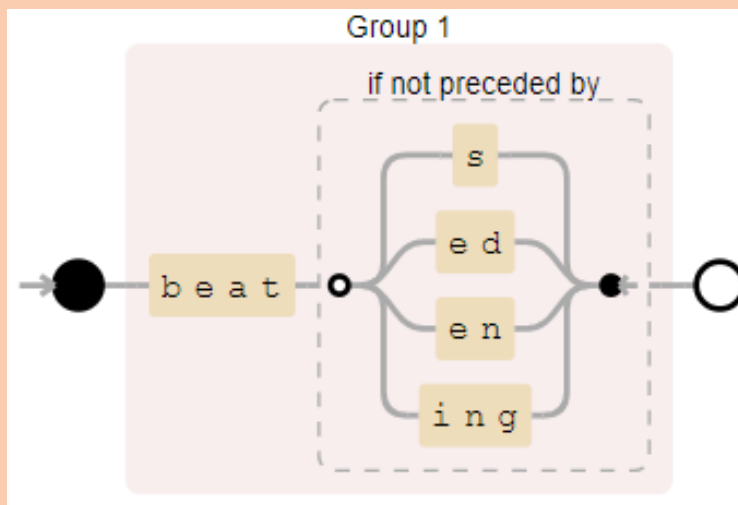
`(beat(?!s|ed|en|ing))`



`(beat(?<=s|ed|en|ing))`



`(beat(?<!=s|ed|en|ing))`



还能再强大些吗？

特殊字符转义成普通字符
普通字符转义成特殊字符

特殊字符

^ 匹配文本行的开头
\$ 匹配文本行的结尾
. 匹配任何除换行符的字符

用 \ 转义

\^ 就代表的乘方符号
\\$ 就代表的是美元符号
\. 就代表的是点

普通字符

普通字符

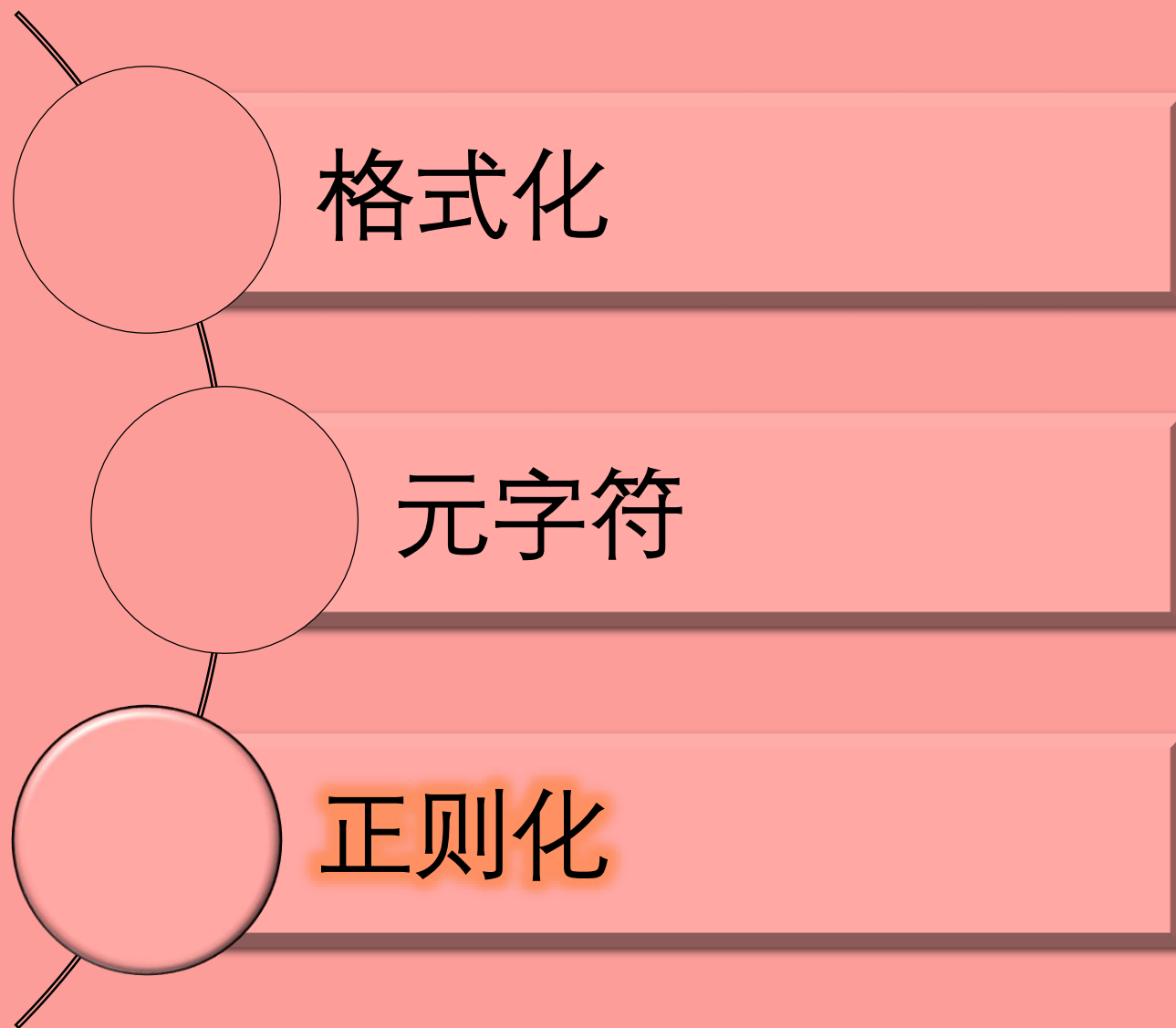
b B d D s S w W A Z t r n 代表英文字母

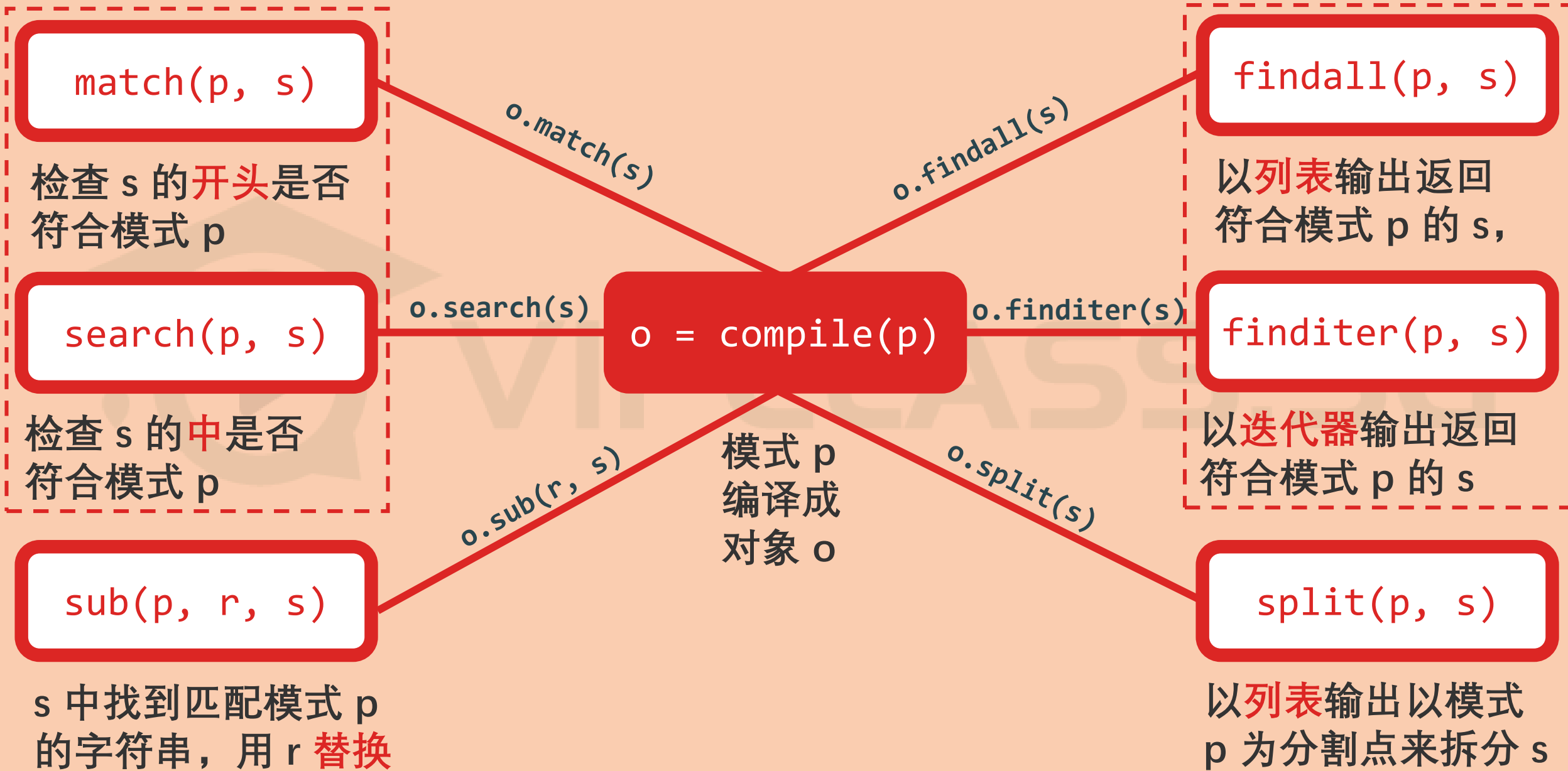
用 \ 转义

\b 匹配空字符串，但仅适用于单词的“首尾”
\B 匹配空字符串，但仅适用于单词的“非首尾”
\d 匹配任何“数字”字符，等价于 [0-9]
\D 匹配任何“非数字”字符，等价于 [^0-9]
\s 匹配任何“空白”字符，等价于 [\t\n\r]
\S 匹配任何“非空白”字符，等价于 [^\t\n\r]
\w 匹配任何“字母数字下划线”字符，等价于 [a-zA-Z0-9_]
\W 匹配任何“非字母数字下划线”字符，等价于 [^a-zA-Z0-9_]
\A 匹配句子的“开头”字符，等价于 ^
\Z 匹配句子的“结尾”字符，等价于 \$
\t 匹配句子的“制表键 (tab)”字符
\r 匹配句子的“回车键 (return)”字符
\n 匹配句子的“换行键 (newline)”字符

特殊字符

字符串专场





总结

内容	核心知识点	用处
格式化	% 字符, \$ 字符, format 函数, f-string	按 模式 美化
元字符	集合 [] 次数 * + ? {} 并列 提取 () 转义 . ^ \$ \	创建 模式
正则化	match, search, findall, finditer, split, sub, compile	按 模式 获取

<https://www.debuggex.com/>

下节预告：解析表达式

终身学习 快乐学习

王圣元 Steven Wang
微信公众号：王的机器