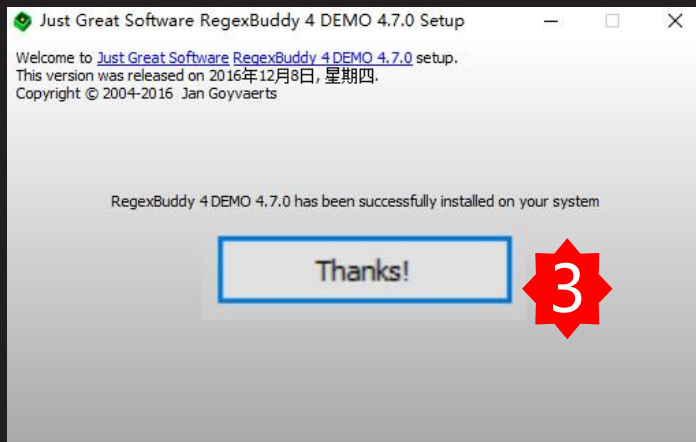
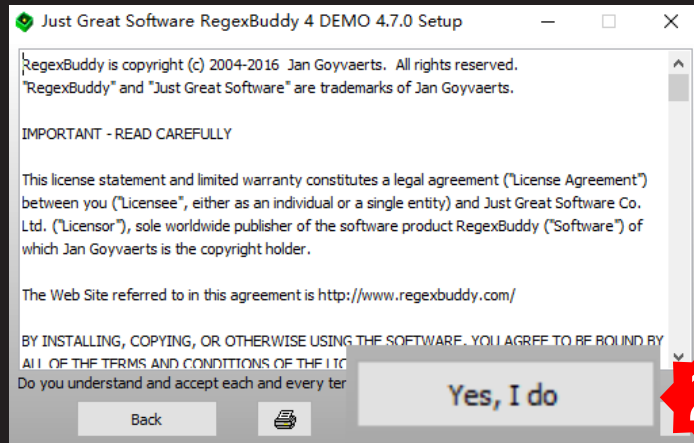
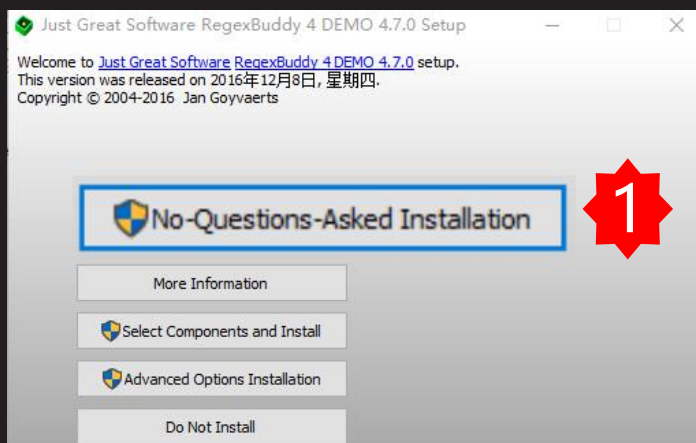


# 正则表达式概述



# 正则表达式概述

- 准备：
  - 安装RegExpBuddy软件——专门测试正则表达式是否正确的软件

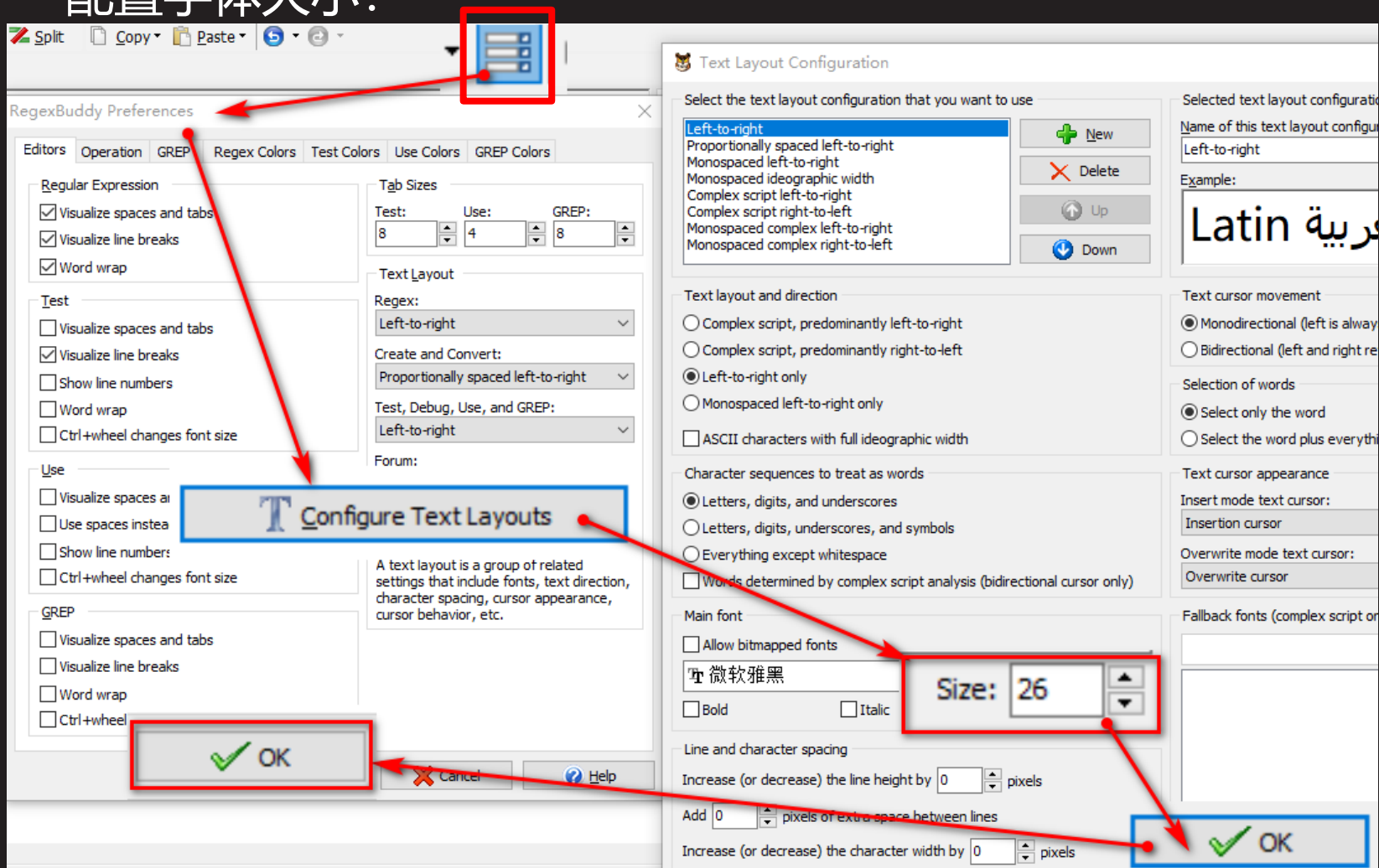


← 桌面上看到图标



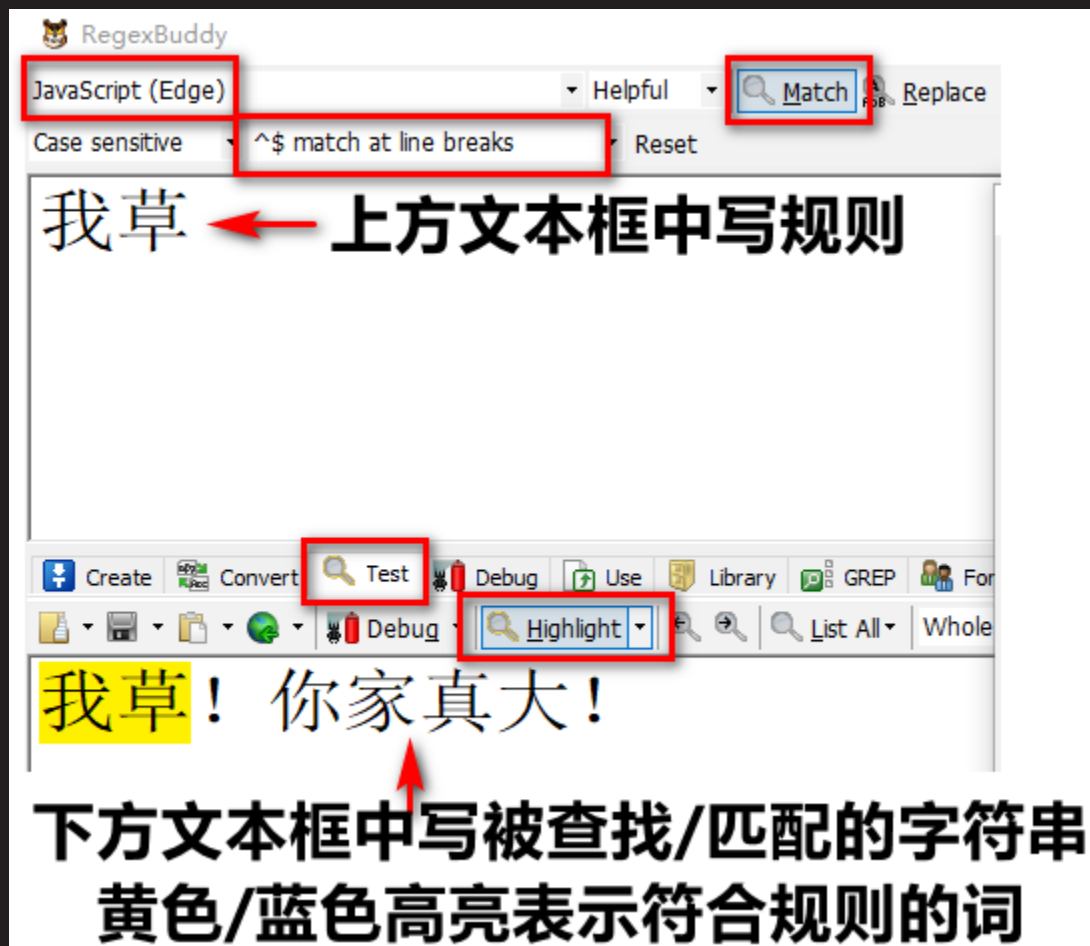
# 正则表达式概述

- 准备：
  - 配置字体大小：



# 正则表达式概述

- 准备：
  - 配置并测试规则匹配是否正常



# 正则表达式概述

- 回顾：注册用户时，要求填写手机号：
  - 如果填写错误的手机号，程序会提示错误！

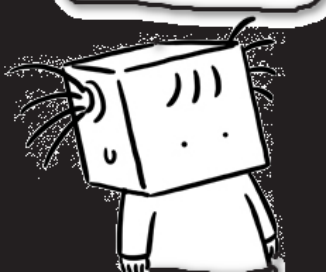
手机号：

- 如果填写正确的手机号，程序会提示正确

手机号：

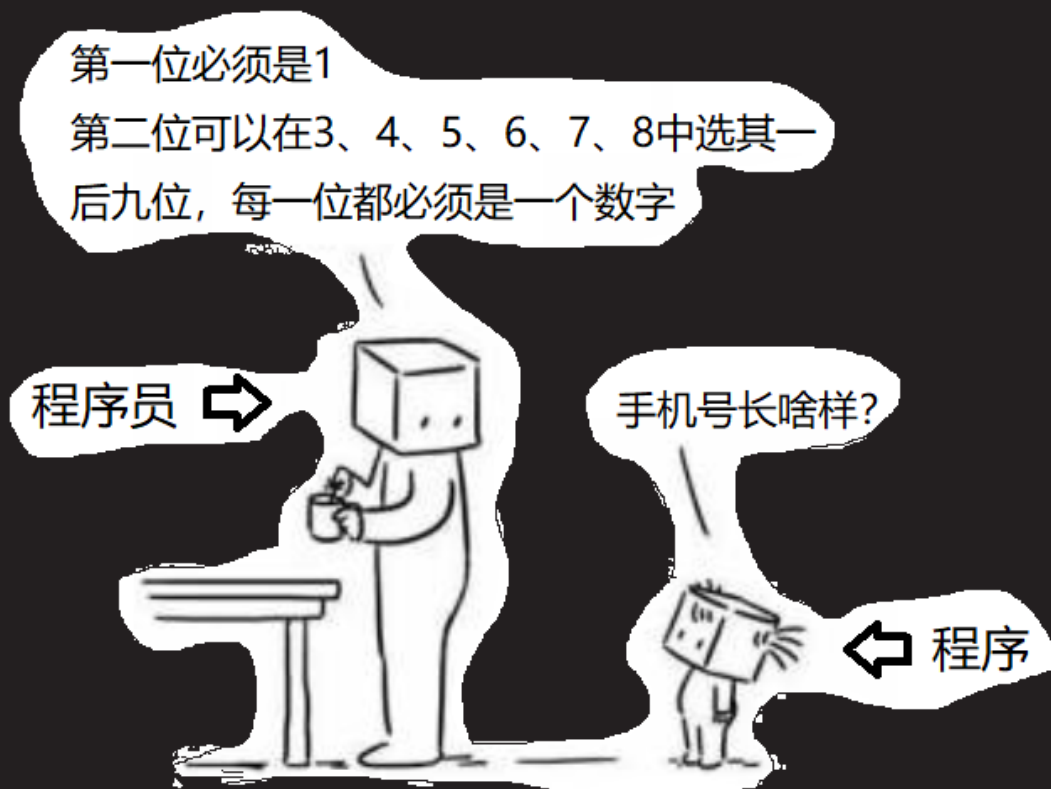
- 问题：程序真的认识手机号吗？
  - 其实，程序原本不认识手机号，是程序员教程序认识的

一脸懵逼...



# 正则表达式概述

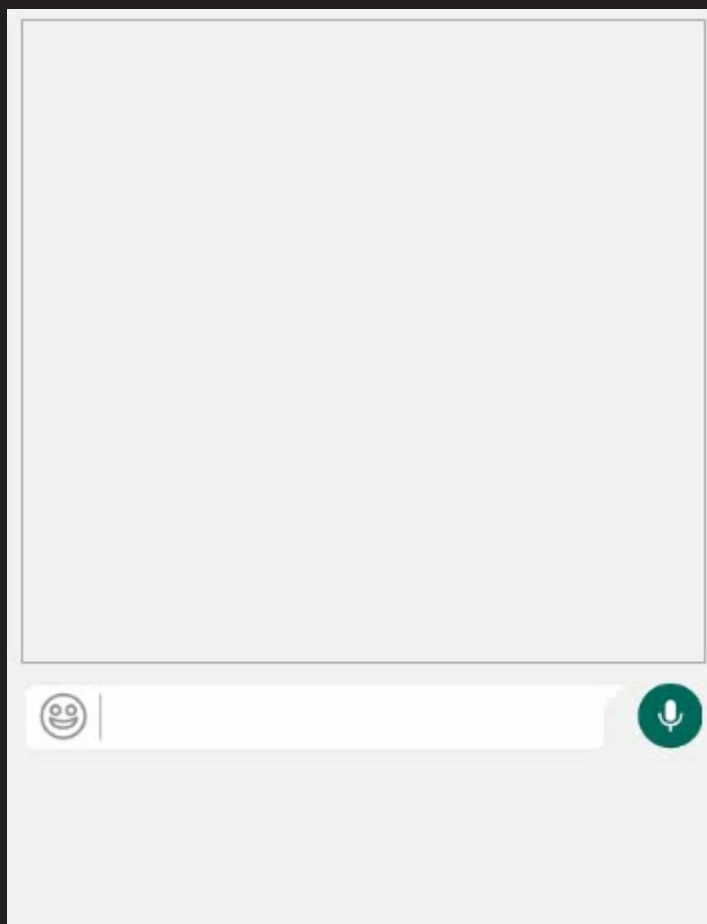
- 正则表达式(Regular Expression): 专门描述字符串中字符出现规则的表达式。
- 因为程序不认识人类语言中的词汇, 所以才需要程序员用正则表达式教程序认识人类语言中的词汇。



# 正则表达式初体验

- 正则表达式可用于：
  - 1. 验证字符串格式
  - 2. 查找敏感词（查水表）

知识讲解



# 定义正则表达式





# 普通字符

- 最简单的规则，就是一个关键词原文
  - 比如：“我草”

第一个字



我

第二个字



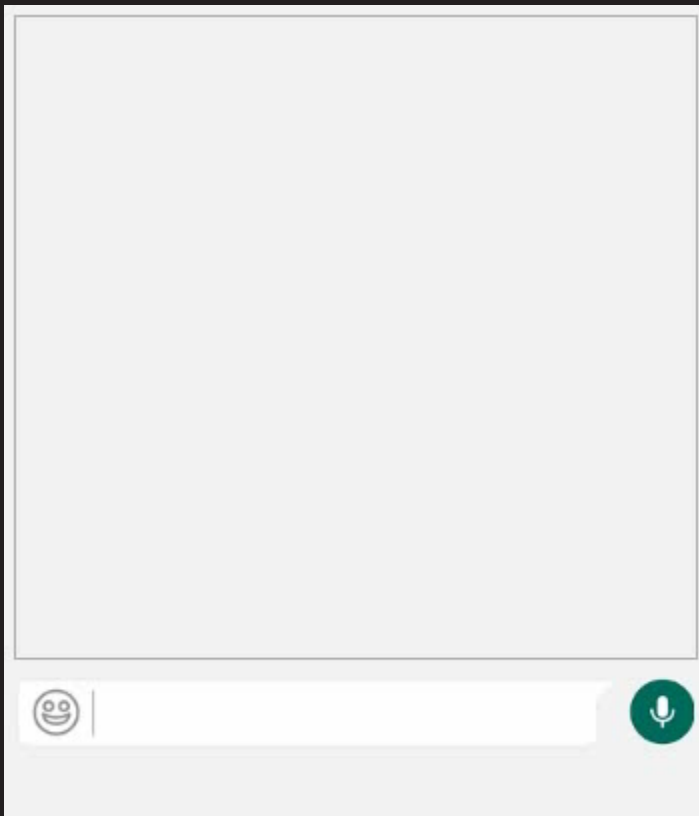
草

# 最简单的正则表达式

- 在RegExp Buddy中
  - 查找一句话中是否包含敏感词 “我草”
  - 如果将一句话中的 “我草” , 换成 “我++” , 还能查询出来吗?

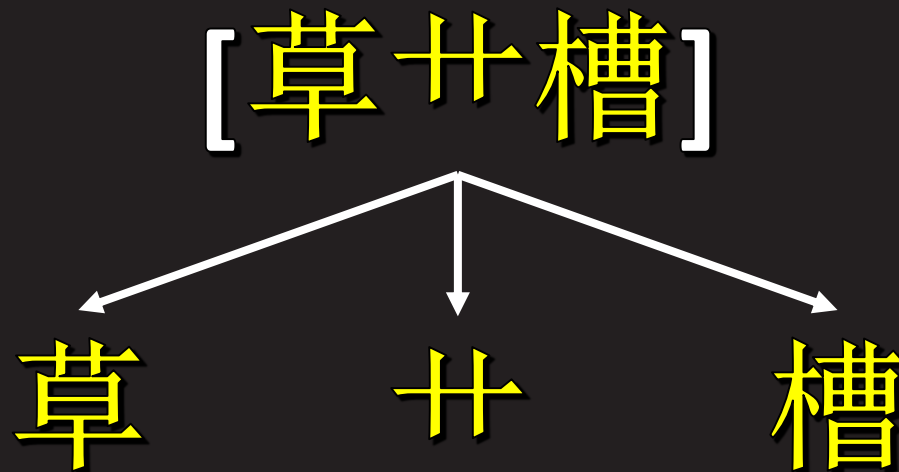
课堂练习

我都听不懂...



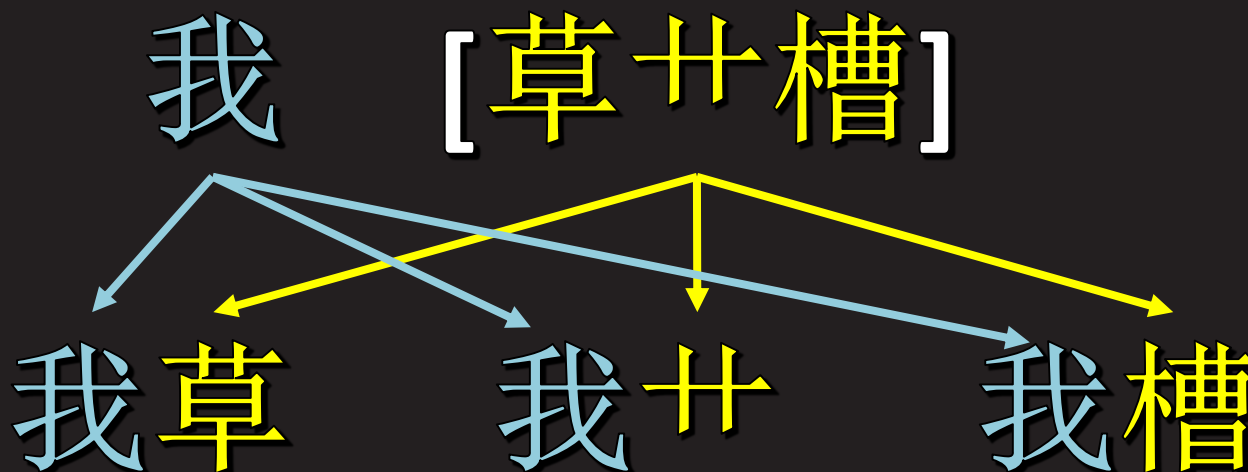
# 字符集

- 问题：第二个字符换成另一个同音字，就匹配不到了
- 字符集是规定一位字符上多种备选字的列表
- 只要规则中某一位字符上有多种备选字时，就用字符集
- 匹配时，只要与[]中任意一个字符匹配，就算满足规则
- 如何: [备选字列表]
  - 比如:



# 字符集

- 比如：连上前边写死的“我”字规则，可匹配三种词



## 使用字符集

- 在RegExp Buddy中
  - 定义一个规则同时匹配 “我草” , “我艹” , “我槽”  
三种敏感词  
答案: 我[草艹槽]
  - 扩展: 修改规则, 使其进一步匹配 “卧槽”  
答案: [我卧][草艹槽]



# 字符集

- 比如，手机号规则：
  - 第一位: 1
  - 第二位: 3、4、5、6、7、8中选其一
  - 后9位，每一位：一位数字即可
- 结果：
  - 1 [345678] [0123456789] [0123456789] [0123456789]  
     [0123456789] [0123456789] [0123456789]  
     [0123456789] [0123456789] [0123456789]



# 字符集

- 简写：如果[]中部分备选字符连续，可用-省略中间字符
- 比如，手机号规则中：

—[345678] 可简写为 [3-8]

—[0123456789] 可简写为 [0-9]

- 所以，手机号规则可简写为：
  - 1[3-8][0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9]



# 使用字符集简写

- 在RegExp Buddy中
    - 利用字符集简写定义手机号规则，验证手机号
- 答案: 1[3-8][0-9][0-9][0-9][0-9][0-9][0-9][0-9] [0-9] [0-9]





# 字符集

- 其它简写：
  - 要匹配一位小写字母: [a-z]
  - 要匹配一位大写字母: [A-Z]
  - 要匹配一位字母: [A-Za-z]
  - 要匹配一位字母或数字: [0-9A-Za-z]
  - 要匹配一位汉字: [\u4e00-\u9fa5]



## 使用字符集简写

- 在RegExp Buddy中
    - 利用字符集简写定义车牌号规则:
    - 第一位: 1位汉字
    - 第二位: 1位大写字母
    - 第三位: .
    - 后五位, 每一位: 都是一位大写字母或数字
- 答案: `[\u4e00-\u9fa5][A-Z].[0-9A-Z][0-9A-Z][0-9A-Z][0-9A-Z]`



# 预定义字符集

- 正则表达式语法为四种最常用的字符集定义了最简化写法，称为预定义字符集。
- 包括：
  - 要匹配一位数字： `\d` 等效于 `[0-9]`
  - 要匹配一位字母、数字或下划线： `\w` 等效于 `[0-9A-Za-z_]`
  - 要匹配一位空字符： `\s` 可匹配 空格、制表符Tab 等空白
  - 要匹配所有文字（通配符）： `.`
- 所以，手机号规则可进一步简写为：
  - `1[3-8]\d\d\d\d\d\d\d\d`



## 使用预定义字符集

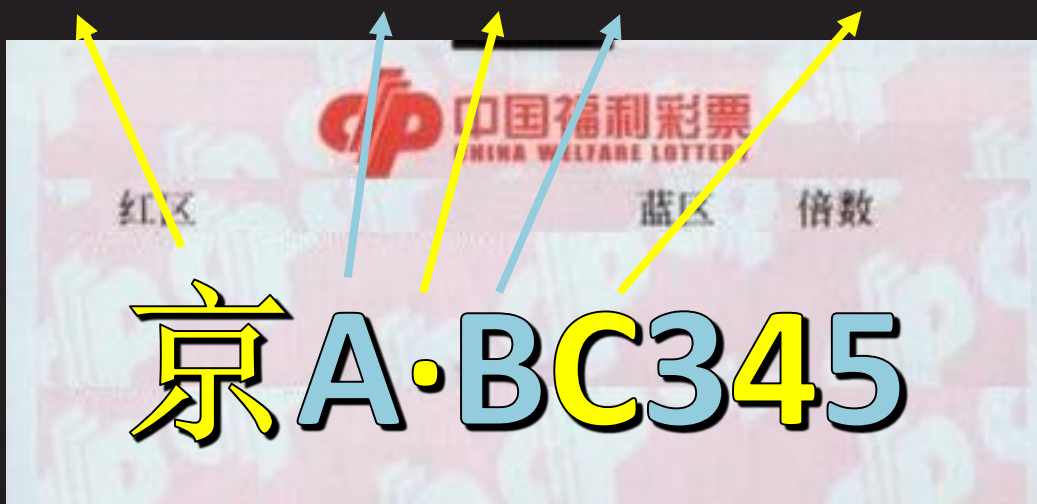
- 在RegExp Buddy中
  - 利用预定义字符集简写手机号规则  
答案: `1[3-8]\d\d\d\d\d\d\d\d`
  - 利用`\s`匹配一句英文中的每个空格  
答案: `\s`



# 数量词

- 问题：手机号规则中\d写了9遍，车牌号规则中[0-9A-Z]也写了五遍！
- 原因：一个字符集 (\d或[0-9]) 只能匹配一位字符，要匹配9位字符，就必须重复写9遍
- 程序用规则匹配字符串，就像彩票兑奖一样，是逐字逐个规则匹配。不但内容要符合规则，位数首先要一致

**`[\u4e00-\u9fa5][A-Z]·[A-Z0-9][A-Z0-9]...[A-Z0-9]`**



# 数量词

- 数量词，是专门规定一个字符集出现次数的规则
- 今后，只要一个字符集在规则中可能连续反复出现多次，就要用数量词以简写方式定义出现次数。
- 如何：数量词紧跟在其修饰的字符集之后，默认修饰相邻的前一个字符集
- 比如：手机号中连续的9个数字\ d，可进一步简写为：

1 [3-8] \ d { 9 }

||

\ d x 9

## 使用数量词

- 在RegEx Buddy中
  - 利用数量词进一步简写手机号规则  
答案: `1[3-8]\d{9}`
  - 利用数量词进一步简写车牌号规则  
答案: `[\u4e00-\u9fa5][A-Z]·[0-9A-Z]{5}`



# 数量词

- 问题：短信验证码的数字可能是4位或6位，不确定位数
- 数量词包括两大类：

- 1. 有明确数量边界的数量词

字符集{n} 表示字符集必须重复n次，不能多也不能少

字符集{n,m} 表示字符集至少重复n次，最多重复m次，

比如：\d{4,6} 表示4到6位数字

字符集{n,} 表示字符集匹配的内容至少重复n次，多了不限，比如：\d{6,} 表示6位以上数字

- 2. 没有明确数量边界的数量词

- \* 可有可无，多了不限

- ? 可有可无，最多一次

- + 至少一次，多了不限



## 使用不确定数量的数量词

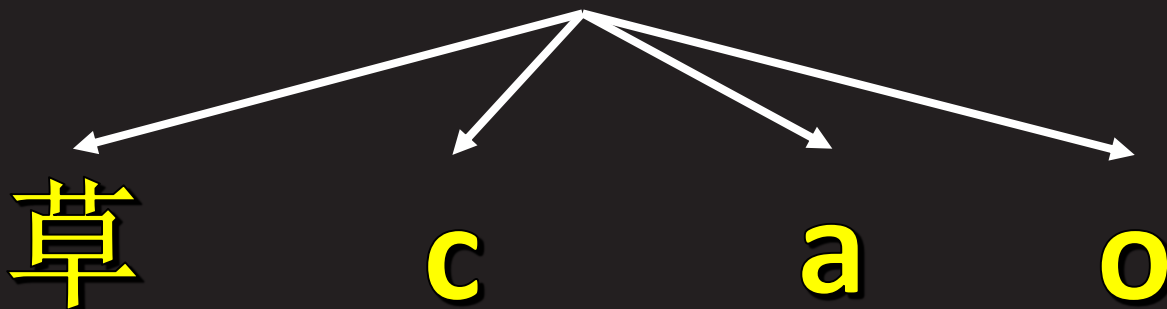
- 在RegExp Buddy中
  - 匹配手机短信中的验证码：连续的4位~6位数字  
答案: `\d{4,6}`
  - 匹配字符串中的一组连续空字符  
答案: `\s+`



# 选择和分组

- 问题：屏蔽敏感词时，屌丝把字换成拼音就查不出来了
- 错误的做法：字符集只认识单个字，不认识cao是整体

[草cao]



- 希望：草 或 cao

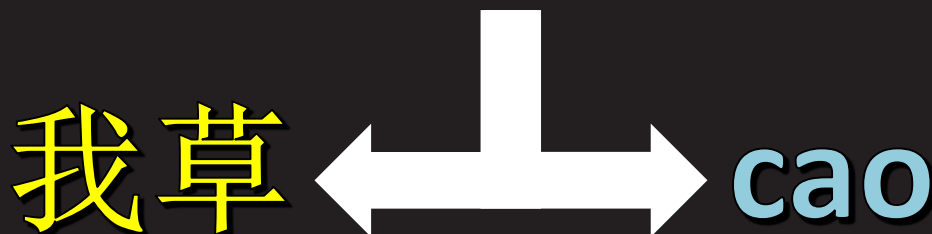
# 选择和分组

- 正确做法：选择
- 选择，是指在多个子规则中选其一匹配
- 今后，只要在多个子规则中选其一匹配时，就用选择
- 如何：子规则1|子规则2
- “|” 选择符只分左右，不考虑单个字符
- 比如：



# 选择和分组

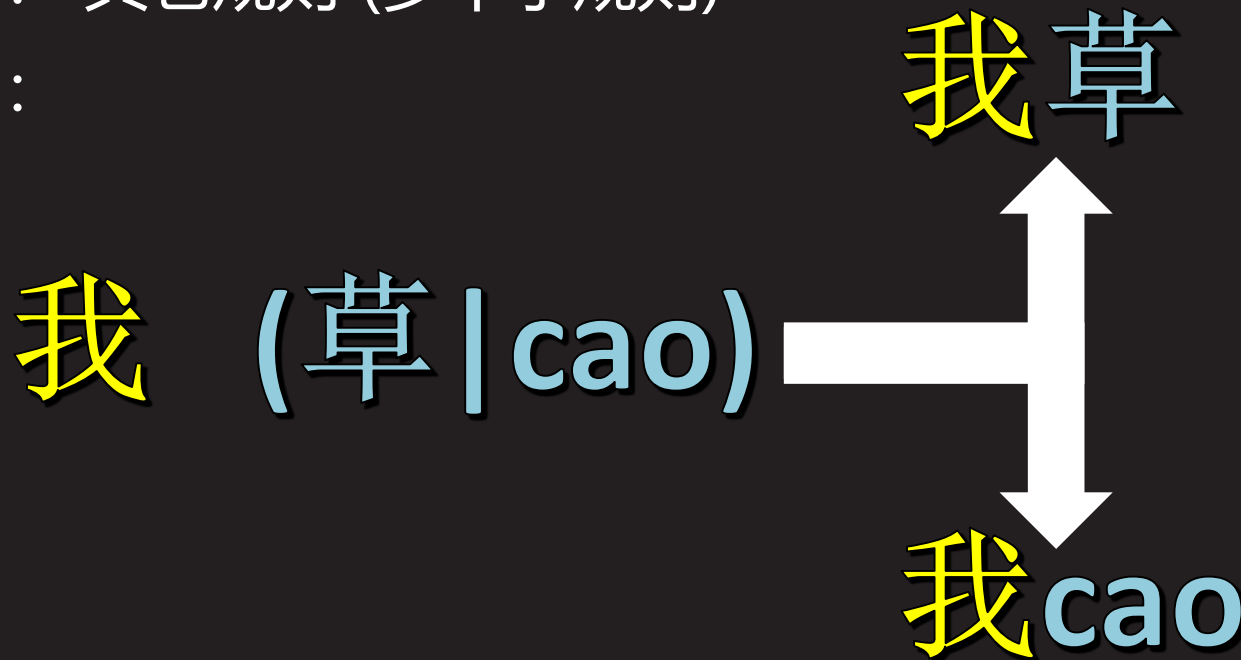
- 问题：如果规则写成 “我草|cao” 呢？
- 希望是：我草 或 我cao

- 实际却是：我草 | cao
- 

- 因为 “|” 选择符只分左右，不考虑单个字符

# 选择和分组

- 分组，将多个子规则视为一组，再和分组外的规则匹配
- 只要希望将多个子规则视为一个整体，再和其它规则匹配时，就用分组
- 如何： 其它规则 (多个子规则)
- 比如：



## 使用选择和分组

- 在RegExp Buddy中定义规则
  - 匹配一个“草”字或“cao”这个拼音  
答案: 草|cao
  - 匹配“我草”或“我cao”  
答案: 我(草|cao)



# 选择和分组

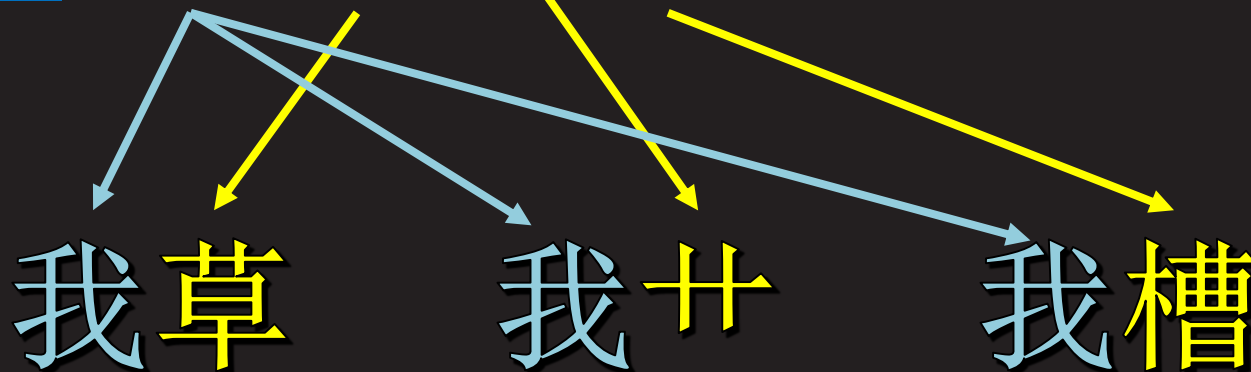
- 比如：同时验证同音字和拼音：

我 ([草 艹 槽] | cao)

我[草 艹 槽]



我cao



# 选择和分组

- 比如：“我”字也可能是“卧”或“wo”

**([我卧]|wo)([草艹槽]|cao)**

- 在比如：可能在中间加不确定个数的空字符：

**([我卧]|wo)\s\*([草艹槽]|cao)**



## 使用选择和分组

- 在RegExp Buddy中定义规则
  - 匹配“我草”，“卧槽”，“wocao”，“我 草”等敏感词



## 使用选择和分组

- 在RegExp Buddy中定义规则

- 定义完整手机号规则：

- +86或0086

- (\+86|0086)

- 至少一个空字符：\s+

- 之前所有，整体可有可无，最多一次：()

- 1

- 3~8 任选其一

- 9位数字

答案: `((\+86|0086)\s+)?1[3-8]\d{9}`



## 使用选择和分组

- 在RegExp Buddy中定义规则
  - 定义完整身份证号规则:
    - 15位数字: `\d{15}`
    - 2位数字: `\d\d`
    - 最后一位: 1位数字或x: `[0-9x]`
    - 最后三位 可有可无, 最多一次 (最后三位) ?

答案: `\d{15}(\d\d[0-9x])?`



## 使用选择和分组

- 在RegExp Buddy中定义规则
  - 匹配“微信”，“weixin”，“wx”等情况，并防止中间加空格——作业

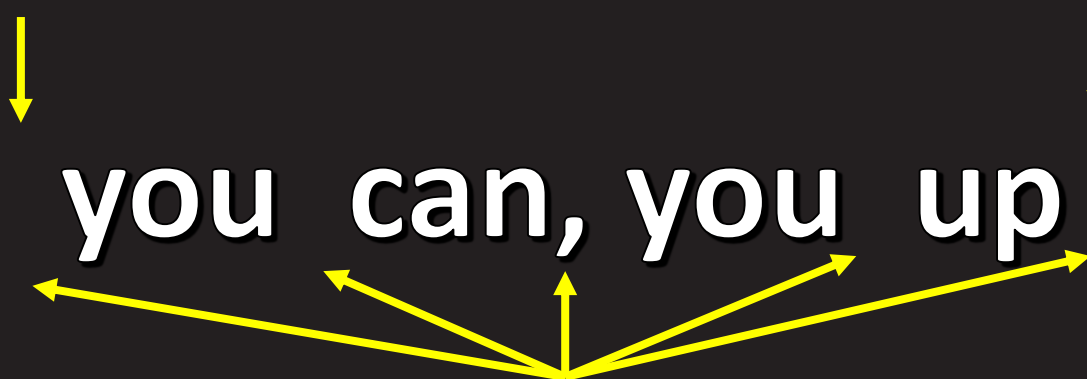


# 指定匹配位置

- 一个字符串中三个位置比较特殊：
  - 1. 字符串开头
  - 2. 字符串结尾
  - 3. 英文句子中的每个单词中间的空白位置

字符串开头

字符串结尾



单词边界

# 指定匹配位置

- 如果只希望匹配特殊位置上的关键词时，就可用特殊符号表示特殊位置。
- 包括：
  - 1. ^ 表示字符串开头
  - 2. \$ 表示字符串结尾
  - 3. \b 表示单词边界，可匹配：空格，标点符号，字符串开头和结尾等可将一个单词与其它单词分割开的符号。

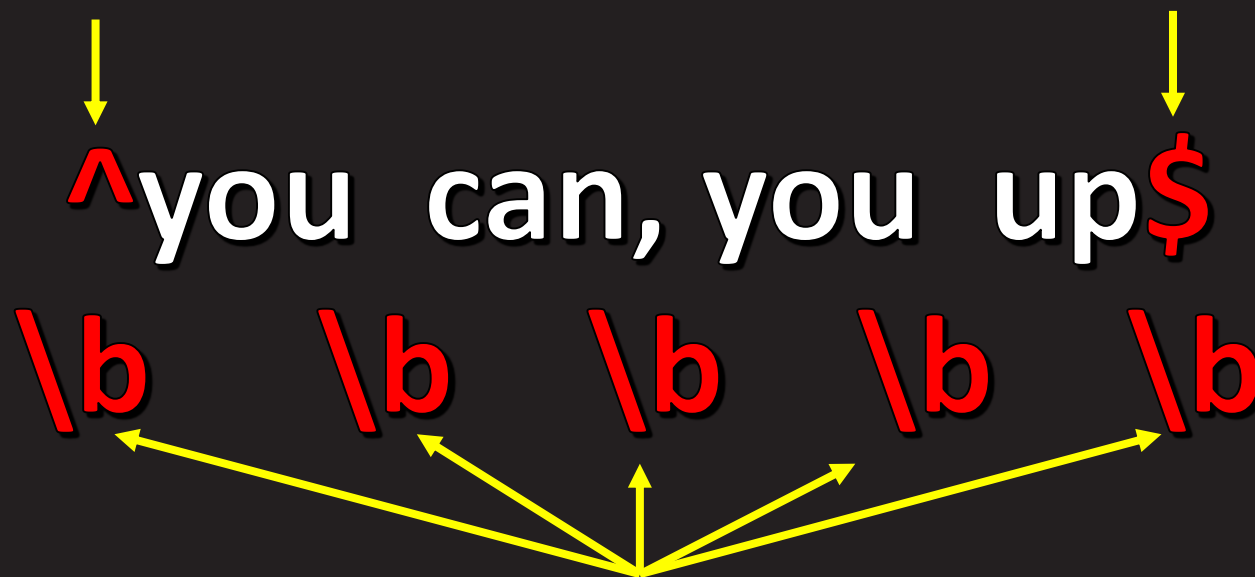


# 指定匹配位置

- 比如：

字符串开头

字符串结尾



单词边界

# 指定匹配位置

- 比如: 匹配一组连续的空字符
  - 1. 匹配任意一组连续的空字符

**\s+**

no zuo no die

- 2. 仅匹配开头的空字符

**^\s+**

no zuo no die

- 3. 仅匹配结尾的空字符

**\s+\$**

no zuo no die



# 指定匹配位置

- 4. 同时匹配开头和结尾的空字符:

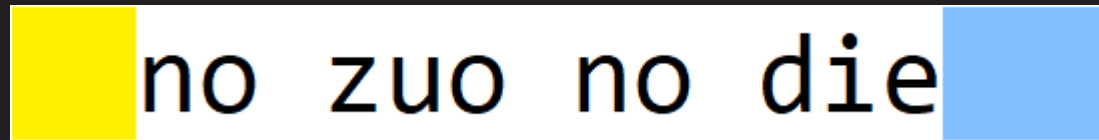
错误的做法:  $\wedge \backslash s+ \$$  表示从开头到结尾之间只能是空字符!

正确做法: 用 “|” 选择符, 将整个规则强行一分为二

$\wedge \backslash s+ | \backslash s+ \$$

$\wedge \backslash s+$

$\backslash s+ \$$

no zuo no die

# 指定匹配位置

- 再比如: 找到每个单词首字母  
——前边紧挨着单词边界的字母

**y**ou    **c**an    **y**ou    **u**p

\b    \b    \b    \b    \b

- 所以: **\b[a-z]**    you can you up

## 使用特殊位置

- 在RegExp Buddy中定义规则
  - 仅匹配字符串开头的空字符
  - 仅匹配字符串结尾的空字符
  - 同时匹配字符串开头和结尾的空字符
  - 匹配每个单词首字母



# 总结和答疑

