正则表达式中不能随意加空格。

简单的数字匹配

例：regular\_test/regular\_test1.py

import re  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 regex = re.compile(r'\d\d\d-\d\d\d-\d\d\d\d')  
 text1 = 'My number is 415-555-4242'  
text2 = 'My number is 18811751156'  
text3 = 'My number is 415-5555-424'  
  
result = regex.search(text1)  
if result:  
print(result.group())  
else:  
print('Not match regular')  
  
 result = regex.search(text2)  
if result:  
print(result.group())  
else:  
print('Not match regular')  
  
 result = regex.search(text3)  
if result:  
print(result.group())  
else:  
print('Not match regular')

输出为：

415-555-4242

Not match regular

Not match regular

()分组

例：regular\_test/regular\_test2.py

import re  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 regex = re.compile(r'(\d\d\d)-(\d\d\d)-(\d\d\d\d)')  
 text1 = 'My number is 415-555-4242'  
result = regex.search(text1)  
print(result.group(0)) # 415-555-4242  
print(result.group(1)) # 415  
print(result.group(2)) # 555  
print(result.group(3)) # 4242  
print()  
  
# 文本中包含()  
text2 = 'My number is (415)-555-4242'  
regex1 = re.compile(r'(\(\d\d\d\))-(\d\d\d)-(\d\d\d\d)')  
 result = regex1.search(text2)  
print(result.group(1)) # (415)  
print(result.group(2)) # 555  
print(result.group(3)) # 4242

管道|匹配

例：regular\_test/regular\_test3.py

import re  
  
if \_\_name\_\_ == '\_\_main\_\_':  
# 匹配许多表达式中的1个  
regex = re.compile(r'Batman|Tina')  
 text = 'aaa Batman and Tina'  
result = regex.search(text)  
print(result.group()) # Batman  
  
text = 'Tina and Batman'  
result = regex.search(text)  
print(result.group()) # Tina  
  
regex = re.compile(r'Bat(man|mobile|copter|bat)')  
 text = 'aaaa Batmobile lost a wheel'  
result = regex.search(text)  
print(result.group()) # Batmobile  
  
text = 'aaaa Batman lost a wheel'  
result = regex.search(text)  
print(result.group()) # Batman  
  
text = 'aaaa Batcopter lost a wheel'  
result = regex.search(text)  
print(result.group()) # Batcopter

?实现可选匹配：?前面的分组为可选匹配（匹配0次或1次）

例：regular\_test/regular\_test4.py

import re  
  
if \_\_name\_\_ == '\_\_main\_\_':  
# ?表示wo为可选匹配  
regex = re.compile(r'Bat(wo)?man')  
 result = regex.search('The Adventures of Batman')  
print(result.group()) # Batman  
  
result = regex.search('The Adventures of Batwoman')  
print(result.group()) # Batwoman  
print()  
  
 regex = re.compile(r'(\d\d\d-)?\d\d\d-\d\d\d\d')  
 result = regex.search('My number is 415-555-4242')  
print(result.group(0)) # 415-555-4242  
print(result.group(1)) # 415-  
print()  
  
 result = regex.search('My number is 555-4242')  
print(result.group()) # 555-4242

\*匹配：\*前面的分组匹配0次或多次

例：regular\_test/regular\_test5.py

import re  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 # \*匹配0次或多次  
 regex = re.compile(r'Bat(wo)\*man')  
 result = regex.search('The Adventures of Batman')  
 print(result.group())  
  
 result = regex.search('The Adventures of Batwoman')  
 print(result.group())  
  
 result = regex.search('The Adventures of Batwowowoman')  
 print(result.group())

输出为：

Batman

Batwoman

Batwowowoman

+号匹配：+前面的分组匹配1次或多次

例：regular\_test/regular\_test6.py

import re  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 # +匹配1次或多次  
 regex = re.compile(r'Bat(wo)+man')  
 result = regex.search('The Adventures of Batman')  
 print(result)  
  
 result = regex.search('The Adventures of Batwoman')  
 print(result.group())  
  
 result = regex.search('The Adventures of Batwowowoman')  
 print(result.group())

None

Batwoman

Batwowowoman

{}匹配：匹配特定次数

例：regular\_test/regular\_test7.py

import re  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 # {}匹配特定次数  
 regex = re.compile(r'Bat(wo){3}man') # 匹配3次  
 result = regex.search('The Adventures of Batwowowoman')  
 print(result.group()) # Batwowowoman  
  
 result = regex.search('The Adventures of Batwowoman')  
 print(result) # None  
 print()  
  
 regex = re.compile(r'Bat(wo){3,5}man') # 匹配3-5次  
 result = regex.search('The Adventures of Batwowowoman') # 3次  
 print(result.group()) # Batwowowoman  
  
 result = regex.search('The Adventures of Batwowowowoman') # 4次  
 print(result.group()) # Batwowowowoman  
  
 result = regex.search('The Adventures of Batwowowowowowoman')  
 print(result) # None  
 print()  
  
 regex = re.compile(r'Bat(wo){3,}man') # 匹配3次及以上  
 result = regex.search('The Adventures of Batwowowoman') # 3次  
 print(result.group()) # Batwowowoman  
  
 result = regex.search('The Adventures of Batwowowowoman') # 4次  
 print(result.group()) # Batwowowowoman  
  
 result = regex.search('The Adventures of Batwowowowowowoman') # 6次  
 print(result.group()) # Batwowowowowowoman  
 print()  
  
 regex = re.compile(r'Bat(wo){,5}man') # 匹配0-5次  
 result = regex.search('The Adventures of Batwowowoman') # 3次  
 print(result.group()) # Batwowowoman  
  
 result = regex.search('The Adventures of Batwowowowoman') # 4次  
 print(result.group()) # Batwowowowoman  
  
 result = regex.search('The Adventures of Batman') # 0次  
 print(result.group()) # Batman

输出为：

Batwowowoman

None

Batwowowoman

Batwowowowoman

None

Batwowowoman

Batwowowowoman

Batwowowowowowoman

Batwowowoman

Batwowowowoman

Batman

findall：

例：regular\_test/regular\_test8.py

import re  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 # 无分组，返回字符串列表  
 regex = re.compile(r'\d\d\d-\d\d\d-\d\d\d\d')  
 print(regex.findall('Cell: 415-555-9999 Word: 212-555-0000'))  
  
 # 有分组  
 regex = re.compile(r'(\d\d\d)-(\d\d\d)-(\d\d\d\d)')  
 print(regex.findall('Cell: 415-555-9999 Word: 212-555-0000'))

输出为：

['415-555-9999', '212-555-0000']

[('415', '555', '9999'), ('212', '555', '0000')]

常用字符分类：

|  |  |
| --- | --- |
| 缩写字符分类 | 表示 |
| \d | 0-9任何数字 |
| \D | 除0-9以外的任何字符 |
| \w | 任何字母、数字或下划线字符 |
| \W | 除字母、数字和下划线以外的字符 |
| \s | 空格、制表符或换行符 |
| \S | 除空格、制表符和换行符以外的字符 |

起始字符(^)和结束字符($)

^:匹配必须发生在被查找文本起始处

$:匹配必须以$之前的正则表达式结束

例：regular\_test/regular\_test9.py

import re  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 regex = re.compile(r'^Hello')  
 result = regex.search('Hello, world!')  
 print(result.group()) # Hello  
  
 result = regex.search('aHello, world!')  
 print(result) # None  
 print()  
  
 regex = re.compile(r'\d$')  
 result = regex.search('Number is 42')  
 print(result.group()) # 2  
  
 result = regex.search('Number is two')  
 print(result) # None

输出为：

Hello

None

2

None

句点通配符：.

例：regular\_test/regular\_test10.py

import re  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 # .匹配除换行之外的任意字符  
 regex = re.compile(r'.at') # 除换行符外，任意字符后跟at都匹配  
 result = regex.findall('The cat in the hat sat on the flat matlab')  
 # .通配符只匹配1个字符  
 print(result) # ['cat', 'hat', 'sat', 'lat', 'mat']

贪心匹配和非贪心匹配：通过?来区别

例：regular\_test/regular\_test11.py

import re  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 # .\*贪心匹配：匹配尽可能多的字符  
 regex = re.compile(r'<.\*>')  
 result = regex.search('<To sever man> fro dinner>')  
 print(result.group()) # <To sever man> fro dinner>  
  
 # .\*非贪心匹配：匹配尽可能少的字符  
 regex = re.compile(r'<.\*?>')  
 result = regex.search('<To sever man> fro dinner>')  
 print(result.group()) # <To sever man>

强口令检测正则：

例：regular\_test/regular\_test12.py

import re  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 text = 'aaL12222222'  
 if len(text) >= 8:  
 have\_number = False  
 regex = re.compile(r'\d+')  
 result = regex.search(text)  
 if result:  
 regex = re.compile(r'[a-z]+')  
 result = regex.search(text)  
 if result:  
 regex = re.compile(r'[A-Z]+')  
 result = regex.search(text)  
 if result:  
 print('Ok')  
 else:  
 print('Not have upper char')  
 else:  
 print('Not have lower char')  
 else:  
 print('Not have number')  
 else:  
 print('length must >= 8')

正则实现strip()

例：regular\_test/regular\_test13.py

import re  
  
  
def re\_strip(text, del\_char=r'\s'):  
 regex\_format = r'^{0}\*|{1}\*$'.format(del\_char, del\_char)  
 regex = re.compile(regex\_format)  
 text = regex.sub('', text)  
 return text  
  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 print(re\_strip('aadasdfaaa', 'a')) # dasdf  
 print(re\_strip(' dafas ')) # dafas