# 正则匹配技巧

- 使用python包re

## 基本函数

- re.match(正则表达式, 原字符串) 从最开始匹配，但只匹配一次。注意，这个从开始匹配，是非常严格的，只要表达式不能在字符串的开始匹配到子串，则返回None
- re.findall(正则表达式, 原字符串) 匹配字符串中所有满足正则表达式的子串
- re.sub(正则表达式, 替换成什么, 原字符串) 将正则表达式匹配到的部分替换成任意其他

## 匹配用的特殊符号

### 1. . 匹配除 \n 之外的所有单字符

In [1]:
```
import re
txt = 'aaa\nasdd'
findall_result = re.findall(r'.', txt)
match_result = re.match(r'.', txt)
# findall返回一个list
# match返回一个对象，其内容为表达式在字符串开头能匹配到的部分
print(findall_result)
print(match_result)
```

```
['a', 'a', 'a', 'a', 's', 'd', 'd']
<_sre.SRE_Match object; span=(0, 1), match='a'>
```

### 2. * 匹配其前方字符的零次或n次重复

In [2]:
```
import re
txt = 'a00daaab00aa0acb'
findall_result = re.findall(r'a*', txt)
match_result = re.match(r'a*', txt)
# findall返回一个list
# match返回一个对象，其内容为表达式在字符串开头能匹配到的部分
print(findall_result)
print(match_result)
```

```
['a', '', '', '', 'aaa', '', '', '', 'aa', '', 'a', '', '', '']
<_sre.SRE_Match object; span=(0, 1), match='a'>
```

In [3]:
```
txt = 'a00daaab00aa0acb'
findall_result = re.findall(r'.*', txt)
match_result = re.match(r'.*', txt)
# findall返回一个list
# match返回一个对象，其内容为表达式在字符串开头能匹配到的部分
print(findall_result)
print(match_result)
```

```
['a00daaab00aa0acb', '']
<_sre.SRE_Match object; span=(0, 16), match='a00daaab00aa0acb'>
```

### 3. + 匹配其前方字符的一次或多次重复

In [4]:
```
import re
txt = 'a00daaab00aa0acb'
findall_result = re.findall(r'a+', txt)
match_result = re.match(r'a+', txt)
# findall返回一个list
# match返回一个对象，其内容为表达式在字符串开头能匹配到的部分
print(findall_result)
print(match_result)
```

```
['a', 'aaa', 'aa', 'a']
<_sre.SRE_Match object; span=(0, 1), match='a'>
```

### 4. [] 中括号内的内容被分成字符，任意匹配其中的每个字符，字符间无先后，是'或'的关系。

In [5]:
```
import re
txt = 'a00daaab00aa0acb'
findall_result = re.findall(r'[ab0]', txt)
match_result = re.match(r'[cba]', txt)
match_result2 = re.match(r'[cb]', txt)
# findall返回一个list
# match返回一个对象，其内容为表达式在字符串开头能匹配到的部分
print(findall_result)
print(match_result)
print(match_result2)
```

```
['a', '0', '0', 'a', 'a', 'a', 'b', '0', '0', 'a', 'a', '0', 'a', 'b']
<_sre.SRE_Match object; span=(0, 1), match='a'>
None
```

### 5. () 标识一个子表达式的范围，可用 (表达式1|表达式2) 同时进行多个表达式的匹配，匹配其中的任意一个。另外，注意表达式有 () 时，例如 (aa)bc，将只能匹配字符串中 caabcnaadd 中的 aa (即函数只返回 () 里的部分)。有多个 ()，则分别返回每个括号匹配到的值

In [6]:
```
import re
txt = 'a00daaab00aa0acb'
findall_result = re.findall(r'(da|a0)aa', txt)
findall_result2 = re.findall(r'(da)(aa)', txt)
findall_result3 = re.findall(r'(da)(ab)', txt)
match_result = re.match(r'(b0|a0)', txt)
# findall返回一个list
# match返回一个对象，其内容为表达式在字符串开头能匹配到的部分
print(findall_result)
print(findall_result2)
print(findall_result3)
print(match_result)
```

```
['da']
[('da', 'aa')]
[]
<_sre.SRE_Match object; span=(0, 2), match='a0'>
```

### 6. \ 用来匹配特殊字符（如换行符 \n），或用来匹配模式符号的原本含义:如，* 匹配 *

In [7]:
```
import re
txt = 'a00d*aa\nab00aa0acb\n'
findall_result = re.findall(r'\*', txt)
match_result = re.match(r'.*(\*)', txt)
match_result2 = re.match(r'\*', txt)
# findall返回一个list
# match返回一个对象，其内容为表达式在字符串开头能匹配到的部分
print(findall_result)
print(match_result)
print(match_result2)
```

```
['*']
<_sre.SRE_Match object; span=(0, 5), match='a00d*'>
None
```

### 7. ？出现在*、+后时，标记前面的子表达式是非贪婪的（意思是最小匹配，原字符串中连续重复多次，但只会匹配其中第一个）

In [8]:
```
import re
txt = 'a00da*aab00aa0acb'
findall_result = re.findall(r'a.*?b', txt)
findall_result2 = re.findall(r'a(.*?)b', txt)
match_result = re.match(r'a.*?b', txt)
match_result2 = re.match(r'.*', txt)
# findall返回一个list
# match返回一个对象，其内容为表达式在字符串开头能匹配到的部分
print(findall_result)
print(findall_result2)
print(match_result)
print(match_result2)
```

```
['a00da*aab', 'aa0acb']
['00da*aa', 'a0ac']
<_sre.SRE_Match object; span=(0, 9), match='a00da*aab'>
<_sre.SRE_Match object; span=(0, 17), match='a00da*aab00aa0acb'>
```

### 8. {n}、{n, n+}表示匹配的次数的限制

In [9]:
```
import re
txt = 'a00da*aab00aa0acb'
findall_result = re.findall(r'a0{2}', txt)
findall_result2 = re.findall(r'a0{1}', txt)
match_result = re.match(r'a{2}', txt)
match_result2 = re.match(r'a{1}', txt)
# findall返回一个list
# match返回一个对象，其内容为表达式在字符串开头能匹配到的部分
print(findall_result)
print(findall_result2)
print(match_result)
print(match_result2)
```

```
['a00']
['a0', 'a0']
None
<_sre.SRE_Match object; span=(0, 1), match='a'>
```

### 9. ^用于限制正则匹配只能匹配字符串的开始位置的相应字符串

In [10]:
```
import re
txt = 'a00da*aab00aa0acb'
findall_result = re.findall(r'^a0', txt)
findall_result2 = re.findall(r'a0', txt)

# findall返回一个list
# match返回一个对象，其内容为表达式在字符串开头能匹配到的部分
print(findall_result)
print(findall_result2)
```

```
['a0']
['a0', 'a0']
```

**10. ^ 在表达式 [^a-z] 中，表示一个取反的正则表达式，匹配a-z以外的所有字符，其中 [] 的用法，参照第3条**

In [11]:
```
import re
txt = 'a00da*aab00aa0acb'
findall_result = re.findall(r'[^0]', txt)
findall_result2 = re.findall(r'[0]', txt)

# findall返回一个list
# match返回一个对象，其内容为表达式在字符串开头能匹配到的部分
print(findall_result)
print(findall_result2)
```

```
['a', 'd', 'a', '*', 'a', 'a', 'b', 'a', 'a', 'a', 'c', 'b']
['0', '0', '0', '0', '0']
```

**11. txt='adb00000000acb', ab=re.findall( r'a(.*?)b', txt)，返回字符串中匹配了表达式 () 中的部分的子串**

In [12]:
```
import re
txt = 'a00da*aab00aa0acb'
findall_result = re.findall(r'a(.*?)b', txt)

# findall返回一个list
# match返回一个对象，其内容为表达式在字符串开头能匹配到的部分
print(findall_result)
```

```
['00da*aa', 'a0ac']
```

**12. **高难度用法**：（1号表达式)(2号表达式)(3号表达式)，这样用 () 括号，将不同正则表达式括起来后，对字符串进行匹配后，可以用 re.group(1或2或3) 单独获取到三个表达式对应匹配到的字符串。( re.group(0) 或者不使用 group 则获取整个全局的正则匹配)**

In [13]:
```
import re
txt = 'a00da*aab00aa0acb'
findall_result = re.search(r'(a.*a)(b.*a)(0.*b)', txt)

# findall返回一个list
# match返回一个对象，其内容为表达式在字符串开头能匹配到的部分
print(findall_result.group(1))
print(findall_result.group(2))
print(findall_result.group(3))
```

```
a00da*aa
b00aa
0acb
```

**13. **高难度用法**：python版命名组方法 (?P<name>正则表达式) 使用 re.group('name')，可以用指定的name匹配某个字符串**

In [14]:
```
import re
txt = 'a00da*aab00aa0acb'
findall_result = re.search(r'(?P<ZhangZhe>d.*)00', txt)

# findall返回一个list
# match返回一个对象，其内容为表达式在字符串开头能匹配到的部分
print(findall_result.group('ZhangZhe'))
```

```
da*aab
```

**注意：网上一些代码类似，** `reg = /a-zA-z/` **其中 / 符号是某些编程语言中，指示一个正则表达式起止的记号**

## 获得英文标点符号

In [15]:
```
from string import punctuation as en_punc

print(en_punc)
```

```
!"#$%&'()*+,-./:;<=>?@[\]^_`{|}~
```