




# 正则表达式

HOPE YOU HAVE FUN LEARNING REGULAR EXPRESSION



# 简介

正则表达式（**Regular Expression**）又称正规表达式、正规表示法、正规表达式等，是计算机科学的一个概念。

正则表达式使用单个字符串描述、匹配一系列符合某个语法规则的字符串。

在很多文本编辑器里，正则表达式通常被用来检索、替换那些匹配某个模式的文本。

# RE模块

Python通过re模块提供对正则表达式的支持。使用前需要使用import语句导入

```
import re
```

使用re的一般步骤是先用正则表达式的字符串形式编译为Pattern实例；然后使用pattern实例处理文本并获得匹配结果（一个match实例）；最后使用match实例获得信息，进行其他的操作。

# 常用操作符

. 表示任何单个字符

[ ] 字符集，对单个字符给出取值范围

[^] 字符非集，对单个字符给出排除范围

‘\*’ 前一个字符0或无限次扩展

‘+’ 前一个字符1或无限次扩展

? 前一个字符0或1次扩展

| 左右表达式任意一个

# 常用操作符

{m} 扩展前一个字符m次

{m,n} 扩展前一个字符m至n次

^ 开头

\$ 结尾

() 代表一个分组，内部可以用|

\d 数字，等价于[0-9]

\w 单词字符，等价于[0-9a-zA-Z]

# 常用方法

`re.compile` # 生成正则表达式

如：

`rule = re.compile(r'[0-9a-zA-Z]')` # 匹配所有数字和字母

`Rule = re.compile(r'[\u4E00-\u9FA5]')` # 匹配所有中文（要求编码为unicode，使用mac或者各种linux或者类unix系统的同学可以试一试）

# 常用方法

`re.search(pattern, string, flags=0)` # 查找字符串中的第一个匹配

```
In [1]: import re
```

```
In [2]: rule = re.compile(r'[1-9]\d{5}') # 匹配一个合法的六位数
```

```
In [3]: s = 'BIT 0123456789'
```

```
In [4]: match = re.search(rule, s)
```

```
In [5]: match.group()
```

```
Out[5]: '123456'
```

# 常用方法

`re.match(pattern, string, flags=0)` # 查找字符串中的第一个匹配(首字符)

```
In [6]: s0 = '0123456789'
```

```
In [7]: s1 = '1234567890'
```

```
In [8]: match0 = re.match(rule, s0)
```

```
In [9]: match1 = re.match(rule, s1)
```

```
In [10]: match0.group()
```

```
-----  
AttributeError                                Traceback (most recent call last)  
<ipython-input-10-b02855437741> in <module>()  
----> 1 match0.group()
```

```
AttributeError: 'NoneType' object has no attribute 'group'
```

```
In [11]: match1.group()
```

```
Out[11]: '123456'
```



# 常用方法

上例中匹配规则仍为`r'[1-9]\d{5}'`

但可见匹配`s0`时返回了一个`None`，及并未匹配到任何内容而匹配`s1`时则成功返回了一个`match`对象。

原因是`re.match`必须从字符串的第一个字符开始匹配，而显然`'0123456789'`首字符为`'0'`，并不符合匹配规则，故返回`None`

# 常用方法

`re.findall(pattern, string, flags=0)` # 查找字符串中所有匹配并以列表的形式返回

```
In [16]: s = '早上好おはよう'
```

```
In [17]: chinese_rule = re.compile(r'[\u4e00-\u9fa5]') # 匹配字符串中所有汉字
```

```
In [18]: result = re.findall(chinese_rule, s)
```

```
In [19]: result
```

```
Out[19]: ['早', '上', '好']
```

# 常用方法

`re.finditer(pattern, string, flags=0)` # 查找字符串中所有匹配并以`match`对象迭代器的形式返回

```
In [20]: s = '早上好おはよう'
```

```
In [21]: chinese_rule = re.compile(r'[\u4e00-\u9fa5]') # 匹配字符串中所有汉字
```

```
In [22]: result = re.finditer(chinese_rule, s)
```

```
In [23]: result
```

```
Out[23]: <callable_iterator at 0x104d63fd0>
```

```
In [24]: for i in result:
...:     print(i)
...:
```

```
<_sre.SRE_Match object; span=(0, 1), match='早 '>
```

```
<_sre.SRE_Match object; span=(1, 2), match='上 '>
```

```
<_sre.SRE_Match object; span=(2, 3), match='好 '>
```

# 常用方法

`re.split(pattern, string, flags=0, maxsplit=0)` # 将字符串按照正则表达式的规则进行切割并将剩余部分以list形式返回，`maxsplit`参数可选（表示匹配的最大个数），默认为0（表示有多少是多少）

```
In [25]: s = '早上好おはよう'
```

```
In [26]: chinese_rule = re.compile(r'[\u4e00-\u9fa5]') # 匹配字符串中所有汉字
```

```
In [27]: r1 = re.split(chinese_rule, s, maxsplit=0)
```

```
In [28]: r2 = re.split(chinese_rule, s, maxsplit=1)
```

```
In [29]: r1
```

```
Out[29]: ['', '', '', 'おはよう']
```

```
In [30]: r2
```

```
Out[30]: ['', '早上好おはよう']
```

```
In [31]: r2 = re.split(chinese_rule, s, maxsplit=4)
```

```
In [32]: r2
```

```
Out[32]: ['', '', '', 'おはよう']
```

# 常用方法

`re.sub(pattern, repl, string, count=0, flags=0)` # 查找字符串中所有匹配并用`repl`替换，然后返回新的字符串，`count`表示最大替换个数，默认为0（表示有多少就替换多少）

```
In [39]: s = '早上好おはよう'
```

```
In [40]: chinese_rule = re.compile(r'[\u4e00-\u9fa5]') # 匹配字符串中所有汉字
```

```
In [41]: new_s = re.sub(chinese_rule, '晚上好', s)
```

```
In [42]: new_s1 = re.sub(chinese_rule, '晚上好', s, count=1)
```

```
In [43]: new_s
```

```
Out[43]: '晚上好晚上好晚上好おはよう'
```

```
In [44]: new_s1
```

```
Out[44]: '晚上好上好おはよう'
```