python re

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

re.match(pattern, string, flag=0)

匹配**开头**是否满足正则

In [2]:
```
m = re.match('www', 'www.baidu.com')
print(m)
m = re.match('com', 'www.baidu.com')
print(m)
```

```
<re.Match object; span=(0, 3), match='www'>
None
```

可以使用 `group(num)` 或 `groups()` 匹配对象函数， 来获取匹配表达式

In [3]:
```
s = "Cats are smarter than dogs"
m = re.match('(.*) are (.*?) .*', s, re.M | re.I)
if m:
    print('m.group() : ' + m.group())
    print('m.group(0) : ' + m.group(0))
    print('m.group(1) : ' + m.group(1))
    print('m.group(2) : ' + m.group(2))
    # print('m.group(3) : ' + m.group(3)) 没有则会报错
```

```
m.group() : Cats are smarter than dogs
m.group(0) : Cats are smarter than dogs
m.group(1) : Cats
m.group(2) : smarter
```

Note that even in MULTILINE mode, `re.match()` will only match at the beginning of the string and not at the beginning of each line.

re.search(pattern, string, flags=0)

在一个字符串中匹配正则表达式的**第一个位置**，返回match对象

In [4]:
```
m = re.search('www', 'www.baidu.com')
print(m)
m = re.search('com', 'www.baidu.com')
print(m)
m = re.search('com', 'www.baidu.com.aaaa.com')
print(m)
```

```
<re.Match object; span=(0, 3), match='www'>
<re.Match object; span=(10, 13), match='com'>
<re.Match object; span=(10, 13), match='com'>
```

re.sub(pattern, repl, string, count=0, flags=0)

在一个字符串中替换所有匹配正则表达式的子串，返回替换后的字符串 repl：替换的字符串，也可为一个函数 count：模式匹配后替换的最大次数，默认 0 表示替换所有的匹配

正则表达式和替换的表达式一定都要用 `'r'` 开头，表示为 raw string, 否则会引发转义，无法正常替换。 可以使用 `\num` 形式表示之前匹配到的 `group` 对应序号的匹配项。

In [5]:
```
m = re.sub(r'def\s+([a-zA-Z_][a-zA-Z_0-9]*)\s*\(\s*\):', r'static PyObjec
print(m)
```

```
static PyObject*
py_myfunc(void)
{
```

re.findall(pattern, string, flags=0)

或 re.findall(string[, pos[, endpos]]) 搜索字符串，返回全部匹配的子串，返回的是列表类型，如果没有找到匹配的，就返回一个空列表 pos：可选参数，指定字符串的起始位置，默认为 0。 endpos：可选参数，指定字符串的结束位置，默认为字符串的长度

In [6]:
```
m = re.findall(r'\bf[a-z]*', 'which foot or hand fell fastest')
print(m)
m = re.findall(r'(\w+)=(\d+)', 'set width=20 and height=10')
print(m)
```

```
['foot', 'fell', 'fastest']
[('width', '20'), ('height', '10')]
```

Match.group()

如果输入参数为空，默认返回 `group(0)`, 即返回匹配到的整个结果 如果输入参数是一个，返回一个string； 如果输入参数是多个，返回一个对应的元组；

In [7]:
```
m = re.match(r"(\w+) (\w+)", "Isaac Newton, physicist")
print(m.group(0))  # The entire match
print(m.group(1))  # The first parenthesized subgroup.
print(m.group(2))  # The second parenthesized subgroup.
print(m.group(1, 2))  # Multiple arguments give us a tuple.
```

```
Isaac Newton
Isaac
Newton
('Isaac', 'Newton')
```

正则表达式中如果有带参数的形式 `(?P<name>...)`， 可以使用 `group('name')` 返回结果

In [8]:
```
m = re.match(r"(?P<first_name>\w+) (?P<last_name>\w+)", "Malcolm Reynolds
print(m.group('first_name'))
print(m.group(1))
print(m.group('last_name'))
print(m.group(2))
```

```
Malcolm
Malcolm
Reynolds
Reynolds
```

如果一个组匹配多次，则只能访问最后一个匹配

In [9]:
```
m = re.match(r"(..)+", "a1b2c3")  # Matches 3 times.
print(m.group(1))
```

```
c3
```