Regular Expression

常用函式

函式	用途
search	從任意位置開始比配
match	從字串的頭開始比配
sub	尋代找到的pattern
findall	把所有的匹配都列出來

Group

確認找到以後, 我們可以使用group或者groups來拿取我們要的對象

你可以使用0來指定group

group: 0(full match) 1~最後(你特別指定的group)

groups: 會將1~最後組合一個元組給你

In [19]:
```
import re

target = "abaaabb"

result = re.search(r"ab", target)
print(result.group(0))

result = re.search(r"ba", target)
print(result.group(0))

result = re.match(r"ab", target)
print(result.group(0))

# 沒找到, result是None
result = re.match(r"ba", target)
print(result)

# 系統定義顯然要取出的group, +代表出現一次以上, 等等會詳細解釋
result = re.search(r"(b(a+)(b+)", target)
# full match
print(result.group(0))
# 第一個小括號(group)
print(result.group(1))
# groups
print(result.groups())
```

```
ab
ba
ab
None
baaabb
aaa
('aaa', 'bb')
```

位置

符號	用途
^	匹配字串開頭
$	匹配字串結尾

In [23]:
```
import re

target = "abaaabb"

result = re.search(r"^ab", target)
print(result)

result = re.search(r"^ba", target)
print(result)

result = re.search(r"abb$", target)
print(result)
```

```
<re.Match object; span=(0, 2), match='ab'>
None
None
```

任意字元

符號	用途
.	匹配任意字元(除了\n), 但如果指定DOTALL 就可以匹配所有字元
*	匹配字元0~任意次
+	匹配字元1~任意次
?	匹配字元0~1次
{m}	匹配m次
{m,n}	匹配m~n次

In [311]:
```
import re

target = "aba\ncdasdf5"

result = re.search(r".*5", target)
print(result)

result = re.search(r".*5", target, re.DOTALL)
print(result)

target = "5"
result = re.search(r".*5", target)
print(result)

# 用+的話至少要出現一次, 不匹配
target = "5"
result = re.search(r".+5", target)
print(result)

# 最後只會匹配一個, 所以會是c5
target = "abc5"
result = re.search(r".?5", target)
print(result)

target = "aaabbb"
result = re.search(r"a{2}b", target)
print(result)

target = "aaaaabbb"
result = re.search(r"a{2,4}b{2,5}", target)
print(result)
```

```
<re.Match object; span=(4, 11), match='o\ncdasdf5'>
<re.Match object; span=(0, 11), match='aba\ncdasdf5'>
<re.Match object; span=(0, 1), match='5'>
None
<re.Match object; span=(2, 4), match='c5'>
<re.Match object; span=(1, 4), match='abb'>
<re.Match object; span=(1, 8), match='aaaabbb'>
```

代稱符號

符號	用途
[]	集合, 可以配合-(範圍)和^(非)
|	or的符號
\b	匹配詞邊界(可以是空白或者標點)
\d	匹配數字(主標和下標數字皆是, 但re.ASCII被立起來的話只匹配正常0-9)
\D	^\d(\d的相反)
\s	匹配空白(blank, tab, 換行)
\w	匹配組成詞語的大部分字符, 如果re.ASCII被立起來的話等價於a-zA-Z0-9_]

In [351]:
```
target = "abbb222914"
result = re.search(r"[0-8]+", target)
print(result)

target = "23"
result = re.search(r"\d+", target)
print(result)

target = "23"
result = re.search(r"\d+", target, re.ASCII)
print(result)
```

```
<re.Match object; span=(4, 7), match='222'>
<re.Match object; span=(0, 2), match='23'>
<re.Match object; span=(0, 1), match='2'>
```

組合

我們可以使用()組成一個詞語, 但記得要加?來表示不用放入group

In [359]:
```
target = "Mr. Chou"
result = re.search(r"(Mr\.)\s+\w+", target)
print(result)
print(result.groups())

target = "Mr. Chou"
result = re.search(r"(?:Mr\.)\s+\w+", target)
print(result)
print(result.groups())

target = "Mrs. Chou"
result = re.search(r"((?:Mr\.)|(?:Mrs\.))\s+(\w+)", target)
print(result)
print(result.groups())

target = "Mrs.       Chou"
result = re.search(r"((?:Mr\.)|(?:Mrs\.))\s+(\w+)", target)
print(result)
print(result.groups())

target = "Mrsss.       Chou"
result = re.search(r"(Mr\.)|(Mrs\.))\s+\w+", target)
print(result)
```

```
<re.Match object; span=(0, 8), match='Mr. Chou'>
('Mr.',)
<re.Match object; span=(0, 8), match='Mr. Chou'>
(1
<re.Match object; span=(0, 9), match='Mrs. Chou'>
('Mrs.', 'Chou')
<re.Match object; span=(0, 13), match='Mrs.       Chou'>
('Mrs.', 'Chou')
None
```

Greedy v.s. Non-Greedy

regex匹配的方式是

也就是我們所說的greedy模式, 盡可能匹配多的字元, 如果是search的話, 第一個字無法匹配, 就會從第二個字開始再盡可能匹配

但有時候我們想要的是non-greedy的模式, 匹配盡可能少就好, 這時候只要數量符號後面加上?即可

In [398]:
```
# greedy
target = "abcasf!sdf!"
result = re.search(r".*!", target)
print(result)

# non-greedy
target = "abcasf!sdf!"
result = re.search(r".*?!", target)
print(result)
```

```
<re.Match object; span=(0, 11), match='abcasf!sdf!'>
<re.Match object; span=(0, 7), match='abcasf!'>
```

FLAG

符號	用途
re.DOTALL	.會吃\n
re.ASCII	只匹配ASCII字符
re.IGNORECASE	不區分大小寫匹配
re.MULTILINE	^會改成匹配每一行的頭, $會匹配每一行的尾

In [111]:
```
target = """1202asdfee\n2301absdf552\n19871222asdf"""
result = re.findall(r"\d+", target)
print(result)

result = re.findall(r"^\d+", target)
print(result)

result = re.findall(r"^\d+", target, re.MULTILINE)
print(result)

result = re.findall(r"[a-z]+$", target, re.IGNORECASE)
print(result)

result = re.findall(r"[a-z]+$", target, re.IGNORECASE | re.MULTILINE)
print(result)
```

```
['1202', '2301', '552', '19871222']
['1202']
['1202', '2301', '19871222']
['asdf']
['asdfee', 'asdf']
```

Verbose

建議使用Verbose配合註解, 讓你的regex更好讀

In [122]:
```
target = "Elwing Mr. Chou"
pattern = r"""
    ([a-z]+)                # match name
    \s+                     # match blanks
    (Mr|Mrs|Miss)\.?        # match middle
    \s+                     # match blanks
    ([a-z]+)                # match surname
"""
result = re.search(pattern, target, re.IGNORECASE | re.VERBOSE)
print(result)
print(result.groups())

target = "Elwing Mr. Chou\nAlice Mrs. Yu"
result = re.findall(pattern, target, re.IGNORECASE | re.VERBOSE | re.MULTILINE)
print(result)
```

```
<re.Match object; span=(0, 15), match='Elwing Mr. Chou'>
('Elwing', 'Mr', 'Chou')
[('Elwing', 'Mr', 'Chou'), ('Alice', 'Mrs', 'Yu')]
```

參考網站

寫不出來的時候可以來這網站拿考一下別人的regex

https://regexlib.com/Search.aspx?k=&c=-1&m=5&ps=20