Regular Expression trong Python

Một Regular Expression là một dãy ký tự đặc biệt giúp bạn so khớp hoặc tìm các chuỗi khác hoặc tập hợp các chuỗi, bởi sử dụng một cú pháp riêng trong một pattern. Regular Expression được sử dụng phổ biến trong thế giới UNIX.

re Module cung cấp sự hỗ trợ đầy đủ các Perl-like Regular Expression trong Python. Module này tạo Exception là **re.error** nếu xảy ra một lỗi trong khi biên dịch hoặc khi sử dụng một Regular Expression.

Có hai hàm quan trọng sẽ được sử dụng để xử lý Regular Expression, đó là:

Hàm match trong Python

Hàm này cố gắng so khớp pattern với string với các **flag** tùy ý. Dưới đây là cú pháp cho hàm này.

```
re.match(pattern, string, flags=0)
```

Chi tiết về tham số:

- **pattern** : Đây là Regular Expression để được so khớp.
- **string** : Đây là chuỗi, mà sẽ được tìm kiếm để so khớp pattern tại phần đầu của chuỗi.
- **flags** : Bạn có thể xác định các flag khác nhau bởi sử dụng toán tử |. Các modifier này sẽ được liệt kê ở bảng bên dưới.

Hàm re.match trả về một đối tượng **match** nếu thành công và trả về **None** nếu thất bại. Chúng ta sử dụng hàm group(num) hoặc groups() của đối tượng match để lấy biểu thức đã được so khớp (kết nối).

- Phương thức group(num=0) trả về toàn bộ kết nối (hoặc num phân nhóm cụ thể).
- Phương thức groups() trả về tất cả các phân nhóm kết nối trong một Tuple (là trống nếu không có kết nối hay so khớp nào).

Ví dụ:

```
import re
```

```
line = "Hoc Python la de hon hoc Java?"

matchObj = re.match( r'(.*) la (.*?) .*', line, re.M|re.I)

if matchObj:
   print "matchObj.group() : ", matchObj.group()
   print "matchObj.group(1) : ", matchObj.group(1)
   print "matchObj.group(2) : ", matchObj.group(2)
else:
   print "Khong co ket noi!!"
```

Kết quả là:

```
matchObj.group() :  Hoc Python la de hon hoc Java?
matchObj.group(1) :  Hoc Python
matchObj.group(2) :  de
```

Hàm search trong Python

Hàm này tìm kiếm cho sự xuất hiện đầu tiên của pattern bên trong string với các flags tùy ý. Dưới đây là cú pháp cho hàm search:

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

Các tham số được giải thích như trong hàm match.

Hàm re.search trả về một đối tượng match nếu thành công và trả về None nếu thất bại. Chúng ta sử dụng hàm group(num) hoặc groups() của đối tượng match để lấy biểu thức đã được so khớp (kết nối). Các hàm này đã được trình bày ở trên.

Ví dụ

```
import re

line = "Hoc Python la de hon hoc Java?";

searchObj = re.search( r'(.*) la (.*?) .*', line, re.M|re.I)
```

```
if searchObj:
   print "searchObj.group() : ", searchObj.group()
   print "searchObj.group(1) : ", searchObj.group(1)
   print "searchObj.group(2) : ", searchObj.group(2)
else:
   print "Khong tim thay!!"
```

Kết quả là:

```
searchObj.group() :  Hoc Python la de hon hoc Java?
searchObj.group(1) :  Hoc Python
searchObj.group(2) :  de
```

Phân biệt match và search trong Python

Python cung cấp hai hoạt động cơ sở dựa trên Reguler Expression, đó là: **match** để kiểm tra chỉ một kết nối tại phần đầu của chuỗi, trong khi **search** tìm kiếm một kết nối ở bất cứ đâu trong chuỗi.

Ví dụ

```
import re

line = "Hoc Python la de hon hoc Java?";

matchObj = re.match( r'thon', line, re.M|re.I)
if matchObj:
   print "match --> matchObj.group() : ", matchObj.group()
else:
   print "Khong co ket noi!!"

searchObj = re.search( r'thon', line, re.M|re.I)
if searchObj:
   print "search --> searchObj.group() : ", searchObj.group()
else:
```

```
   print "Khong tim thay!!"
```

Kết quả là:

```
Khong co ket noi!!
search --> searchObj.group() :  thon
```

Tìm kiếm và thay thế trong Python

Một trong những phương thức quan trọng nhất mà sử dụng với Regular Expression là **sub**. Dưới đây là cú pháp:

```
re.sub(pattern, repl, string, max=0)
```

Phương thức này thay thế tất cả sự xuất hiện của pattern trong string với repl. Phương thức này sẽ thay thế tất cả sự xuất hiện trừ khi bạn cung cấp tham số max. Phương thức này trả về chuỗi đã được sửa đổi.

Ví dụ

```
import re

phone = "01633-810-628 # Day la so dien thoai"

# Xoa cac comment
num = re.sub(r'#.*$', "", phone)
print "So dien thoai : ", num

# Xoa cac ky tu khong phai ky so
num = re.sub(r'\D', "", phone)
print "So dien thoai : ", num
```

Kết quả là:

```
So dien thoai :  01633-810-628
So dien thoai :  01633810628
```

Danh sách modifier trong Python

Các modifier được liệt kê dưới đây có thể được sử dụng như là các flag tùy ý cho các hàm match và search. Bạn có thể cung cấp nhiều modifier bởi sử dụng toán tử |.

Modifier	Miêu tả
re.I	Thực hiện việc kết nối hoặc so khớp không phân biệt kiểu chữ
re.L	Thông dịch các từ theo Locale hiện tại
re.M	Làm cho $ kết nối với phần cuối của một dòng (mà không chỉ kết nối phần cuối của chuỗi) và làm cho ^ kết nối với phần đầu của bất cứ dòng nào (mà không chỉ kết nối phần đầu của chuỗi)
re.S	Làm cho dot (dấu chấm) kết nối với bất kỳ ký tự nào, bao gồm một newline
re.U	Thông dịch các chữ cái theo bộ ký tự Unicode. Flag này ảnh hưởng tới hành vi của \w, \W, \b, \B
re.X	Cho phép cú pháp "cuter" regular expression. Nó bỏ qua các khoảng trống trắng whitespace (ngoại trừ bên trong một [] hoặc khi được tránh bởi \) và coi # dưới dạng một comment

Các Pattern trong Python

Ngoại trừ các ký tự điều khiển, **(+ ? . * ^ $ ( ) [ ] { } | \)**, thì tất cả ký tự còn lại sẽ kết nối với chính chúng. Bạn có thể tránh một ký tự điều khiển bằng cách đặt trước nó một dấu \.

Dưới đây là danh sách các pattern có sẵn trong Python:

Pattern	Miêu tả
^	Dưới đây là danh sách các pattern có sẵn trong Python:

$	Kết nối với phần cuối của dòng
.	Kết nối bất kỳ ký tự đơn nào ngoại trừ newline. Sử dụng tùy chọn m cho phép nó kết nối với newline
[...]	Kết nối với bất kỳ ký tự đơn nào trong []
[^...]	Kết nối với bất kỳ ký tự đơn nào không ở trong []
re*	Kết nối với 0 hoặc nhiều sự xuất hiện của biểu thức đặt trước
re+	Kết nối với 1 hoặc nhiều sự xuất hiện của biểu thức đặt trước
re?	Kết nối với 0 hoặc 1 sự xuất hiện của biểu thức đặt trước
re{ n}	Kết nối với n sự xuất hiện của biểu thức đặt trước
re{ n,}	Kết nối với n hoặc nhiều hơn sự xuất hiện của biểu thức đặt trước
re{ n, m}	Kết nối với ít nhất n và nhiều nhất m sự xuất hiện của biểu thức đặt trước
a| b	Kết nối với a hoặc b
(re)	Nhóm các Regular Expression và ghi nhớ text đã kết nối
(?imx)	Bật toggle tạm tời các tùy chọn i, m hoặc x bên trong một Regular Expression. Nếu trong cặp dấu ngoặc đơn thì chỉ khu vực đó bị ảnh hưởng
(?-imx)	Tắt toggle tạm tời các tùy chọn i, m hoặc x bên trong một Regular Expression. Nếu trong cặp dấu ngoặc đơn thì chỉ khu vực đó bị ảnh

	hưởng
(?: re)	Nhóm các Regular Expression mà không ghi nhớ text đã kết nối
(?imx: re)	Bật toggle tạm thời các tùy chọn i, m, hoặc x bên trong cặp dấu ngoặc đơn
(?-imx: re)	Tắt toggle tạm thời các tùy chọn i, m, hoặc x bên trong cặp dấu ngoặc đơn
(?#...)	Comment
(?= re)	Xác định vị trí bởi sử dụng một pattern. Không có một dãy giá trị
(?! re)	Xác định vị trí bởi sử dụng sự phủ định pattern. Không có một dãy giá trị
(?> re)	Kết nối pattern độc lập mà không backtrack
\w	Kết nối các ký tự từ
\W	Kết nối các ký tự không phải là từ
\s	Kết nối với whitespace. Tương đương với [\t\n\r\f]
\S	Kết nối với các ký tự không là whitespace
\d	Kết nối với các ký số. Tương đương với [0-9]
\D	Kết nối với các ký tự không là ký số

\A	Kết nối với phần đầu của chuỗi
\Z	Kết nối với phần cuối của chuỗi. Nếu một newline tồn tại, nó kết nối với phần ở trước newline
\z	Kết nối với phần cuối của chuỗi
\G	Kết nối với điểm mà tại đó kết nối cuối cùng được tìm thấy
\b	Kết nối với các giới hạn từ khi bên ngoài các dấu []. Kết nối với backspace (mã là 0x08) khi bên trong các dấu []
\B	Kết nối với các giới hạn không phải là từ
\n, \t, etc.	Kết nối với newline, carriage return, tab, ...
\1...\9	Kết nối với biểu thức con được nhóm thứ n
\10	Kết nối biểu thức con được nhóm thứ n nếu nó đã kết nối. Nếu không, tham chiếu tới biểu diễn bát phân của một mã ký tự

Ví dụ lớp ký tự trong Python

Ví dụ	Miêu tả
[Pp]ython	Kết nối với "Python" hoặc "python"
rub[ye]	Kết nối với "ruby" hoặc "rube"
[aeiou]	Kết nối với bất kỳ một nguyên âm nào

[0-9]	Kết nối với bất kỳ ký số nào, giống dạng [0123456789]
[a-z]	Kết nối với bất kỳ chữ cái thường ASCII nào
[A-Z]	Kết nối với bất kỳ chữ cái hoa ASCII nào
[a-zA-Z0-9]	Kết nối với bất kỳ cái nào ở trên
[^aeiou]	Kết nối với bất kỳ cái nào ở trên ngoại trừ nguyên âm
[^0-9]	Kết nối với bất kỳ cái nào ở trên ngoại trừ một ký số