Python Regular Expressions

Ekspresi reguler adalah urutan khusus karakter yang membantu Anda mencocokkan atau menemukan string atau rangkaian senar lainnya, menggunakan sintaks khusus yang dipegang dalam sebuah pola. Ekspresi reguler banyak digunakan di dunia UNIX.

Modul ini memberikan dukungan penuh untuk ekspresi reguler seperti Perl dengan Python. Modul re meningkatkan pengecualian re.error jika terjadi kesalahan saat mengkompilasi atau menggunakan ekspresi reguler.

Kami akan membahas dua fungsi penting, yang akan digunakan untuk menangani ekspresi reguler. Tapi ada hal kecil dulu: Ada berbagai karakter, yang tentunya memiliki arti khusus bila digunakan dalam ekspresi reguler. Untuk menghindari kebingungan saat berhadapan dengan ekspresi reguler, kita akan menggunakan Raw Strings sebagai r'expression '.
Fungsi Pertandingan

Fungsi ini mencoba mencocokkan pola RE dengan string dengan flag pilihan.

Berikut adalah sintaks untuk fungsi ini -

Parameter	**Description**
pattern	Ini adalah ekspresi reguler yang harus disesuaikan.
string	Ini adalah string, yang akan dicari agar sesuai dengan pola pada awal string.
flags	Anda dapat menentukan flag yang berbeda menggunakan bitwise OR (|). Ini adalah pengubah, yang tercantum dalam tabel di bawah ini.

Fungsi re.match mengembalikan objek yang cocok pada kesuksesan, None on failure. Kami mengelompokkan (num) atau kelompok () fungsi objek pencocokan untuk mendapatkan ekspresi yang sesuai.

Match Object Methods	**Description**
group(num=0)	Metode ini mengembalikan seluruh kecocokan (atau jumlah sub-kelompok tertentu)
groups()	Metode ini mengembalikan semua subkelompok yang cocok dalam tupel (kosong jika tidak ada)

#! / Usr / bin / python
Impor kembali

Line = "Kucing lebih pintar dari pada anjing"

MatchObj = re.match (r '(. *) Adalah (. *?). *', Line, re.M | re.I)

jika cocokObj:
  cetak "matchObj.group ():", matchObj.group ()

```
  cetak "matchObj.group (1):", matchObj.group (1)
  Cetak "matchObj.group (2):", matchObj.group (2)
lain:
  cetak "Tidak ada pertandingan !!"
```

Bila kode diatas dieksekusi, maka menghasilkan hasil sebagai berikut -

```
MatchObj.group (): Kucing lebih pintar dari pada anjing
MatchObj.group (1): Kucing
MatchObj.group (2): lebih pintar
```

Fungsi Pencarian

Fungsi ini mencari kejadian pertama dari pola RE dalam string dengan flag pilihan.

Berikut adalah sintaks untuk fungsi ini:

Re.search (pola, string, flag = 0)

Berikut adalah deskripsi parameternya:

Parameter	Description
pattern	Ini adalah ekspresi reguler yang harus disesuaikan.
string	Ini adalah string, yang akan dicari agar sesuai dengan pola di manapun dalam string.
flags	Anda dapat menentukan flag yang berbeda menggunakan bitwise OR (|). Ini adalah pengubah, yang tercantum dalam tabel di bawah ini.

Fungsi re.search mengembalikan objek yang cocok pada kesuksesan, tidak ada yang gagal. Kami menggunakan fungsi kelompok (num) atau kelompok () dari objek pertandingan untuk mendapatkan ekspresi yang sesuai.

Match Object Methods	Description
group(num=0)	Metode ini mengembalikan seluruh kecocokan (atau jumlah subkelompok tertentu)
groups()	Metode ini mengembalikan semua subkelompok yang cocok dalam tupel (kosong jika tidak ada)

```
#!/usr/bin/python
import re

line = "Cats are smarter than dogs";

searchObj = re.search( r'(.*) are (.*?) .*', 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 "Nothing found!!"
```

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

Pencocokan Versus Searching

Python menawarkan dua operasi primitif yang berbeda berdasarkan ekspresi reguler: cek kecocokan untuk kecocokan hanya di awal string, sementara pencarian memeriksa kecocokan di manapun dalam string (inilah yang Perl lakukan secara default).
Contoh

```
#!/usr/bin/python
import re

line = "Cats are smarter than dogs";

matchObj = re.match( r'dogs', line, re.M |re.I)
if matchObj:
  print "match –> matchObj.group() : ", matchObj.group()
else:
  print "No match!!"

searchObj = re.search( r'dogs', line, re.M |re.I)
if searchObj:
  print "search –> searchObj.group() : ", searchObj.group()
else:
  print "Nothing found!!"
```

```
No match!!
search –> matchObj.group() : dogs
```

Cari dan Ganti

Salah satu metode re yang paling penting yang menggunakan ekspresi reguler adalah sub.
Sintaksis

```
Re.sub (pola, repl, string, max = 0)
```

Metode ini menggantikan semua kemunculan pola RE dalam string dengan repl, mengganti semua kejadian kecuali jika max diberikan. Metode ini mengembalikan string yang dimodifikasi.
Contoh

```
#!/usr/bin/python
import re
```

phone = "2004-959-559 # This is Phone Number"

# Delete Python-style comments
num = re.sub(r' #.* $', "", phone)
print "Phone Num : ", num

# Remove anything other than digits
num = re.sub(r' \D', "", phone)
print "Phone Num : ", num

Phone Num : 2004-959-559
Phone Num : 2004959559

Regular Expression Modifiers: Option Flags

Ekspresi reguler literal mungkin termasuk pengubah opsional untuk mengendalikan berbagai aspek pencocokan. Pengubah ditentukan sebagai bendera pilihan. Anda dapat memberikan beberapa pengubah menggunakan OR eksklusif (|), seperti yang ditunjukkan sebelumnya dan dapat ditunjukkan oleh salah satu dari ini -

Modifier	Description
re.I	Lakukan pencocokan case-insensitive.
re.L	Menafsirkan kata-kata sesuai dengan lokal saat ini. Interpretasi ini mempengaruhi kelompok abjad (\ w dan \ W), serta perilaku batas kata (\ b dan \ B).
re.M	Membuat $ cocok dengan akhir baris (bukan hanya akhir string) dan membuat ^ cocok dengan awal baris apapun (bukan hanya permulaan string).
re.S	Membuat sebuah periode (dot) cocok dengan karakter apapun, termasuk newline.
re.U	Menginterpretasikan huruf sesuai dengan karakter Unicode. Flag ini mempengaruhi perilaku \ w, \ W, \ b, \ B.
re.X	Memungkinkan sintaks ekspresi reguler "manis". Ini mengabaikan spasi (kecuali di dalam himpunan [] atau saat diloloskan oleh garis miring terbalik) dan memperlakukan unescaped # sebagai tanda komentar.

Pola Ekspresi Reguler

Kecuali karakter kontrol, (+?. ^ ^ $ () [] { } | \), Semua karakter cocok dengan karakter mereka sendiri. Anda bisa lolos dari karakter kontrol sebelum mendahului dengan garis miring terbalik.

Berikut daftar tabel sintaks ekspresi reguler yang tersedia dengan Python -
#!/usr/bin/python
import re

phone = "2004-959-559 # This is Phone Number"

```
# Delete Python-style comments
num = re.sub(r' #.* $', "", phone)
print "Phone Num : ", num

# Remove anything other than digits
num = re.sub(r' \D', "", phone)
print "Phone Num : ", num

Phone Num :  2004-959-559
Phone Num :  2004959559

<Directory "/var/www/cgi-bin">
  AllowOverride None
  Options ExecCGI
  Order allow,deny
  Allow from all
</Directory>

<Directory "/var/www/cgi-bin">
Options All
</Directory>

!/usr/bin/python

print "Content-type:text/html \r \n \r \n"
print '<html>'
print '<head>'
print '<title>Hello Word - First CGI Program</title>'
print '</head>'
print '<body>'
print '<h2>Hello Word! This is my first CGI program</h2>'
print '</body>'
print '</html>'
```

Halo kata! Ini adalah program CGI pertamaku

Script hello.py ini adalah skrip Python yang sederhana, yang menuliskan hasilnya pada file STDOUT, yaitu layar. Ada satu fitur penting dan tambahan yang tersedia yang merupakan baris pertama yang akan dicetak Content-type: text / html \ r \ n \ r \ n. Baris ini dikirim kembali ke browser dan ini menentukan jenis konten yang akan ditampilkan di layar browser.

Sekarang Anda pasti sudah mengerti konsep dasar CGI dan Anda bisa menulis banyak program CGI yang rumit dengan menggunakan Python. Script ini bisa berinteraksi dengan sistem eksternal lainnya juga untuk bertukar informasi seperti RDBMS.
Header HTTP

Baris Content-type: text / html \ r \ n \ r \ n adalah bagian dari header HTTP yang dikirim

ke browser untuk memahami isinya. Semua header HTTP akan berada dalam bentuk berikut -

```
#!/usr/bin/python

# Import modules for CGI handling
import cgi, cgitb

# Create instance of FieldStorage
form = cgi.FieldStorage()

# Get data from fields
first _name = form.getvalue('first _name')
last _name  = form.getvalue('last _name')

print "Content-type:text/html \r \n \r \n"
print "<html>"
print "<head>"
print "<title>Hello - Second CGI Program</title>"
print "</head>"
print "<body>"
print "<h2>Hello %s %s</h2>" % (first _name, last _name)
print "</body>"
print "</html>"
```

```
<form action="/cgi-bin/hello _get.py" method="get">
First Name: <input type="text" name="first _name">  <br />

Last Name: <input type="text" name="last _name" />
<input type="submit" value="Submit" />
</form>
```

```
#!/usr/bin/python

# Import modules for CGI handling
import cgi, cgitb

# Create instance of FieldStorage
form = cgi.FieldStorage()

# Get data from fields
first _name = form.getvalue('first _name')
last _name  = form.getvalue('last _name')

print "Content-type:text/html \r \n \r \n"
print "<html>"
print "<head>"
print "<title>Hello - Second CGI Program</title>"
print "</head>"
print "<body>"
```

```
print "<h2>Hello %s %s</h2>" % (first _name, last _name)
print "</body>"
print "</html>"

<form action="/cgi-bin/hello _get.py" method="post">
First Name: <input type="text" name="first _name"><br />
Last Name: <input type="text" name="last _name" />

<input type="submit" value="Submit" />
</form>
```

Mari kita ambil lagi contoh yang sama seperti di atas yang melewati dua nilai menggunakan HTML FORMULIR dan tombol kirim. Kami menggunakan skrip CGI yang sama hello _get.py untuk menangani masukan ini.

Melewati Data Kotak Centang ke Program CGI

Kotak centang digunakan bila lebih dari satu pilihan diperlukan untuk dipilih.

Berikut adalah contoh kode HTML untuk form dengan dua kotak centang -

Melewati Data Tombol Radio ke Program CGI

Tombol Radio digunakan bila hanya satu pilihan yang harus dipilih.

Berikut adalah contoh kode HTML untuk form dengan dua tombol radio

```
#!/usr/bin/python

# Import modules for CGI handling
import cgi, cgitb

# Create instance of FieldStorage
form = cgi.FieldStorage()

# Get data from fields
if form.getvalue('subject'):
   subject = form.getvalue('subject')
else:
   subject = "Not set"

print "Content-type:text/html \r \n \r \n"
print "<html>"
print "<head>"
print "<title>Radio - Fourth CGI Program</title>"
print "</head>"
print "<body>"
print "<h2> Selected Subject is %s</h2>" % subject
```

```
print "</body>"
print "</html>"

<form action="/cgi-bin/textarea.py" method="post" target=" _blank">
<textarea name="textcontent" cols="40" rows="4">
Type your text here...
</textarea>
<input type="submit" value="Submit" />
</form>

#!/usr/bin/python

# Import modules for CGI handling
import cgi, cgitb

# Create instance of FieldStorage
form = cgi.FieldStorage()

# Get data from fields
if form.getvalue('textcontent'):
   text _content = form.getvalue('textcontent')
else:
   text _content = "Not entered"

print "Content-type:text/html \r \n \r \n"
print "<html>"
print "<head>";
print "<title>Text Area - Fifth CGI Program</title>"
print "</head>"
print "<body>"
print "<h2> Entered Text Content is %s</h2>" % text _content
print "</body>"
```

Menggunakan Cookies di CGI

Protokol HTTP adalah protokol tanpa kewarganegaraan. Untuk situs komersial, diperlukan informasi sesi di antara halaman yang berbeda. Misalnya, satu pendaftaran pengguna berakhir setelah menyelesaikan banyak halaman. Bagaimana cara mempertahankan informasi sesi pengguna di semua halaman web?

Dalam banyak situasi, menggunakan cookies adalah metode yang paling efisien untuk mengingat dan melacak preferensi, pembelian, komisi, dan informasi lainnya yang diperlukan untuk pengalaman pengunjung atau statistik situs yang lebih baik.

Contoh dasar
Joke: apa yang kamu sebut babi dengan tiga mata? Piiig!

Aturan dasar pencarian ekspresi reguler untuk sebuah pola dalam sebuah string adalah:

Hasil pencarian melalui string dari awal sampai akhir, berhenti pada pertandingan pertama yang ditemukan

Semua pola harus dicocokkan, tapi tidak semua senar

Jika cocok = re.search (tepuk, str) berhasil, kecocokan tidak ada dan khususnya match.group () adalah teks yang cocok

Search for pattern 'iii' in string 'piiig'. # # All of the pattern must match, but it may appear anywhere.

On success, match.group() is matched text. match = re.search(r'iii', 'piiig') => found, match.group() == "iii" match = re.search(r'igs', 'piiig') => not found, match == None # # . = any char but \n match = re.search(r'..g', 'piiig') => found, match.group() == "iig" # # \d = digit char, \w = word char match = re.search(r' \d \d \d', 'p123g') => found, match.group() == "123" match = re.search(r' \w \w \w', '@@abcd!!') => found, match.group() == "abc"

Pengulangan

Hal menjadi lebih menarik saat Anda menggunakan + dan * untuk menentukan pengulangan dalam polanya

+ - 1 atau lebih kemunculan pola ke kiri, mis. 'I +' = satu atau lebih i's

* - 0 atau lebih kemunculan pola ke kiri

? - cocokkan 0 atau 1 kemunculan pola ke kiri

Paling kiri & terbesar

Pertama, pencarian menemukan kecocokan paling kiri untuk pola tersebut, dan kedua mencoba menggunakan sebanyak mungkin string - yaitu + dan * sejauh mungkin (huruf + dan * dikatakan "serakah").

Contoh pengulangan

i+ = one or more i's, as many as possible. match = re.search(r'pi+', 'piiig') => found, match.group() == "piii" # # Finds the first/leftmost solution, and within it drives the + # # as far as possible (aka 'leftmost and largest'). # # In this example, note that it does not get to the second set of i's. match = re.search(r'i+', 'piigiiii') => found, match.group() == "ii" # # \s* = zero or more whitespace chars # # Here look for 3

digits, possibly separated by whitespace. match = re.search(r' \d \s* \d \s* \d', 'xx1 2 3xx') => found, match.group() == "1 2 3" match = re.search(r' \d \s* \d \s* \d', 'xx12 3xx') => found, match.group() == "12 3" match = re.search(r' \d \s* \d \s* \d', 'xx123xx') => found, match.group() == "123" # # ^ = matches the start of string, so this fails: match = re.search(r' ^b \w+', 'foobar') => not found, match == None # # but without the ^ it succeeds: match = re.search(r'b \w+', 'foobar') => found, match.group() == "bar"

Contoh email

Misalkan Anda ingin mencari alamat email di dalam string 'xyz alice-b@google.com ungu monyet'. Kami akan menggunakan ini sebagai contoh yang berjalan untuk menunjukkan fitur ekspresi reguler. Berikut adalah upaya menggunakan pola r ' \ w + @ \ w +':

Str = 'ungu alice-b@google.com monyet pencuci piring'

Match = re.search (r ' \ w + @ \ w +', str)
Jika cocok:
print match.group () # # 'b @ google'

Pencarian tidak mendapatkan keseluruhan alamat email dalam kasus ini karena \ w tidak cocok dengan '-' atau '.' Di alamat Kami akan memperbaikinya menggunakan fitur ekspresi reguler di bawah ini.

Kurung persegi

Tanda kurung siku dapat digunakan untuk menunjukkan sekumpulan karakter, jadi [abc] cocok dengan 'a' atau 'b' atau 'c'. Kode \ w, \ s dll bekerja di dalam kurung siku juga dengan satu pengecualian bahwa titik (.) Hanya berarti titik literal. Untuk masalah email, tanda kurung siku adalah cara mudah untuk menambahkan '.' Dan '-' ke kumpulan karakter yang dapat muncul di sekitar @ dengan pola r '[\ w .-] + @ [\ w .-] +' untuk mendapatkan keseluruhan alamat email:

Match = re.search (r '[\ w .-] + @ [\ w .-] +', str)

Jika cocok:
print match.group () # # 'alice-b@google.com'

(Lebih banyak fitur kotak-braket) Anda juga dapat menggunakan tanda hubung untuk menunjukkan jangkauan, jadi [a-z] cocok dengan semua huruf kecil. Untuk menggunakan tanda hubung tanpa menunjukkan jangkauan, pasang tanda hubung terakhir, mis. [abc-]. Top-up (^) pada awal set persegi-braket telah membalikkannya, jadi [^ ab] berarti karakter apapun kecuali 'a' atau 'b'.

Ekstraksi Grup

Fitur "kelompok" dari ekspresi reguler memungkinkan Anda untuk memilih bagian dari teks yang sesuai. Misalkan untuk masalah email yang ingin kita ekstrak username dan host secara terpisah. Untuk melakukan ini, tambahkan kurung () di sekitar nama pengguna dan

host dalam pola, seperti ini: r '([\ w .-] +) @ ([\ w .-] +)'. Dalam kasus ini, tanda kurung tidak mengubah pola yang akan cocok, sebaliknya mereka membentuk "kelompok" logis dalam teks pertandingan. Pada pencarian yang sukses, match.group (1) adalah teks kecocokan yang sesuai dengan tanda kurung kiri ke 1, dan match.group (2) adalah teks yang sesuai dengan kurung kiri ke-2. Match.group polos () masih merupakan keseluruhan teks pertandingan seperti biasa.

```
Str = 'ungu alice-b@google.com monyet pencuci piring'

Match = re.search ('([ \ w .-] +) @ ([ \ w .-] +)', str)
Jika cocok:
  Print match.group () # # 'alice-b@google.com' (keseluruhan pertandingan)
  Print match.group (1) # # 'alice-b' (nama pengguna, grup 1)
  Print match.group (2) # # 'google.com' (host, grup 2)
```

Alur kerja umum dengan ekspresi reguler adalah Anda menulis sebuah pola untuk hal yang

Anda cari, menambahkan kelompok tanda kurung untuk mengekstrak bagian yang Anda inginkan.

Temukan semua

Findall () mungkin adalah fungsi tunggal yang paling kuat dalam modul re. Di atas kami menggunakan re.search () untuk menemukan kecocokan pertama untuk sebuah pola. Findall () menemukan * semua * kecocokan dan mengembalikannya sebagai daftar string, dengan masing-masing string mewakili satu kecocokan.

```
  # # Misalkan kita memiliki teks dengan banyak alamat email

Str = 'ungu alice@google.com, bla monyet bob@abc.com blah pencuci piring'
  # # disini re.findall () mengembalikan daftar semua string email yang ditemukan

Email = re.findall (r '[ \ w \ .-] + @ [ \ w \ .-] +', str) # # ['alice@google.com', 'bob@abc.com']
Untuk email di email:
  # Lakukan sesuatu dengan setiap string email yang ditemukan
  Cetak email
```