Expressions régulières

Chapitre 11

Python for Informatics: Exploring Information
www.pythonlearn.com

Expressions régulières

En informatique, une expression régulière, aussi dénommé “regex” ou “regexp”, fournit un moyen concis et flexible pour la correspondance de chaînes de texte, tels que des caractères particuliers, mots ou motifs de caractères. Une expression régulière est écrite dans un langage formel qui peut être interprété par un processeur d'expression régulière.

http://fr.wikipedia.org/wiki/Expression_rationnelle

Expressions régulières

Ce sont des expressions “Joker“ très intelligentes
pour rechercher et analyser des chaînes de texte

http://fr.wikipedia.org/wiki/Expression_rationnelle

Vraiment intelligent "Trouver" ou "Rechercher"

Comprendre les expressions régulières

- Très puissant et très énigmatique
- Amusant une fois que vous les comprenez
- Les expressions régulières sont un langage en soi
- Un langage de « caractères marqueurs » - programmation avec symboles
- C’est une sorte de langage “vieille école” - compacte

http://xkcd.com/208/

Guide rapide des expressions régulières

`^`	Correspond au début d'une ligne
`$`	Correspond à la fin d'une ligne
`.`	Correspond à n'importe quel caractère
`\s`	Correspond à un espace
`\S`	Correspond à n'importe quel caractère sans espace
`*`	Répète un caractère zéro ou plusieurs fois
`*?`	Répète un caractère zéro ou plusieurs fois (non vorace)
`+`	Répète un caractère une ou plusieurs fois
`+?`	Répète un caractère une ou plusieurs fois (non vorace)
`[aeiou]`	Correspond à un caractère unique dans une liste
`[^XYZ]`	Correspond à un caractère unique qui n'est pas dans une liste
`[a-z0-9]`	Le jeu de caractères peut inclure un éventail
`(`	Indique où l'extraction de chaîne doit commencer
`)`	Indique où l'extraction de chaîne doit se terminer

Le Module d'expressions régulières

- Avant que vous puissiez utiliser des expressions régulières dans votre programme, vous devez importer la bibliothèque à l'aide de ''import re''
- Vous pouvez utiliser re.search() pour voir si une chaîne correspond à une expression régulière, semblable à l'utilisation de la méthode find() pour les chaînes
- Vous pouvez utiliser re.findall() pour extraire des fragments d'une chaîne correspondant à votre expression régulière semblable à une combinaison de find() et coupe: var[5:10]

Utiliser re.search() comme find()

```
hand = open('mbox-short.txt')
for line in hand:
    line = line.rstrip()
    if line.find('From:') >= 0:
        print line
```

```
import re

hand = open('mbox-short.txt')
for line in hand:
    line = line.rstrip()
    if re.search('From:', line) :
        print line
```

Utiliser re.search() comme startswith()

```
hand = open('mbox-short.txt')
for line in hand:
    line = line.rstrip()
    if line.startswith('From:') :
        print line
```

```
import re

hand = open('mbox-short.txt')
for line in hand:
    line = line.rstrip()
    if re.search('^From:', line) :
        print line
```

Nous avons perfectionné la recherche en ajoutant des caractères spéciaux à la chaîne

Caractères Joker

- Le point correspond à tout caractère
- Si vous ajoutez le symbole astérisque, il sera répété « n'importe quel nombre de fois »

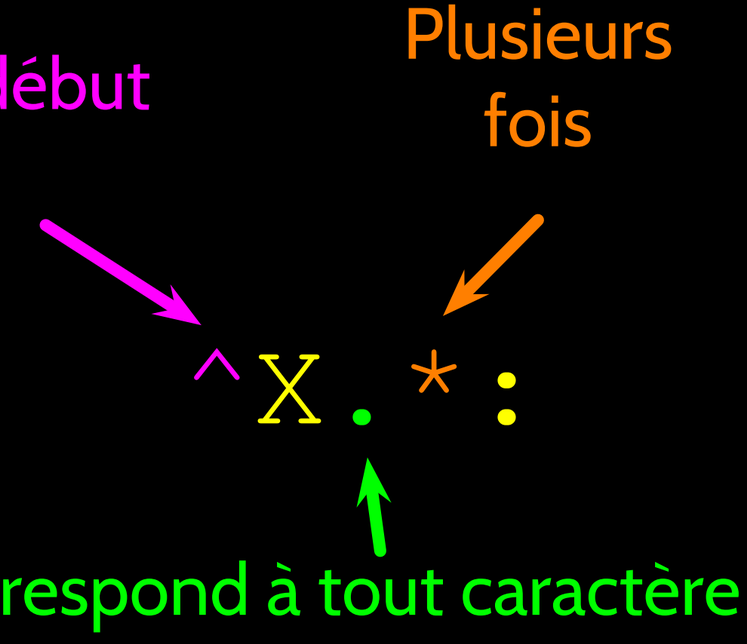
```
X-Sieve: CMU Sieve 2.3
X-DSPAM-Result: Innocent
X-DSPAM-Confidence: 0.8475
X-Content-Type-Message-Body: text/plain
```

```
^X.*:
```

Caractères Joker

- Le point correspond à tout caractère
- Si vous ajoutez le symbole astérisque, il sera répété « n'importe quel nombre de fois »

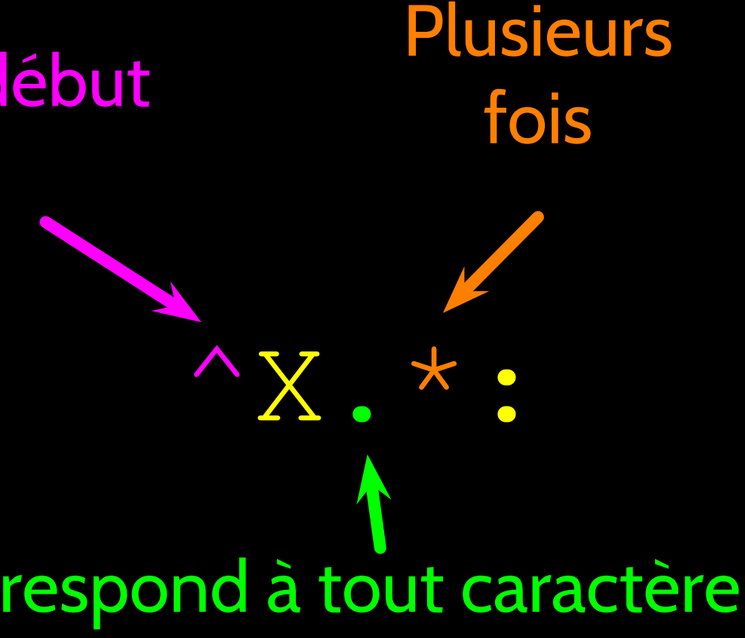
```
X-Sieve: CMU Sieve 2.3
X-DSPAM-Result: Innocent
X-DSPAM-Confidence: 0.8475
X-Content-Type-Message-Body: text/plain
```



Ajustez votre recherche

- Selon le degré de «propreté» de vos données et l'objet de votre demande, vous pouvez raccourcir légèrement votre recherche

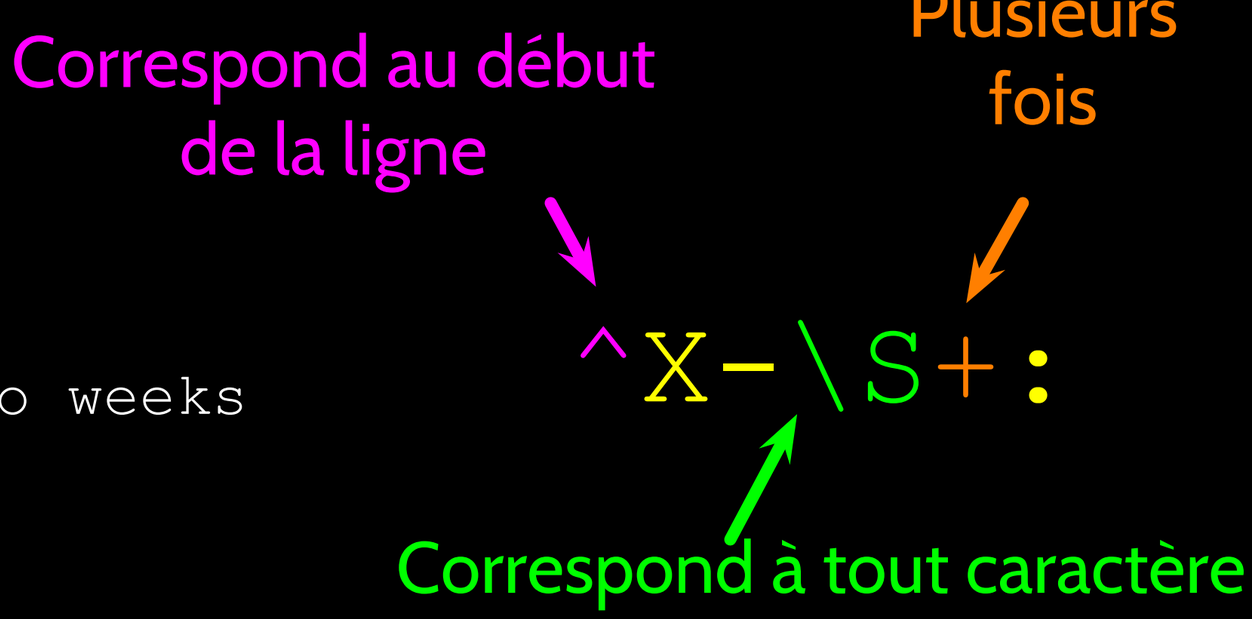
```
X-Sieve: CMU Sieve 2.3
X-DSPAM-Result: Innocent
X-Plane is behind schedule: two weeks
```



Ajustez votre recherche

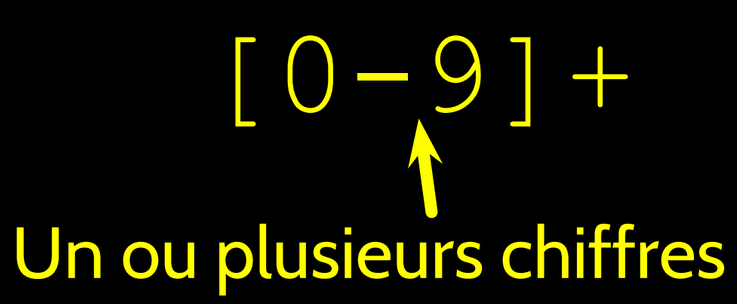
- Selon le degré de «propreté» de vos données et l'objet de votre demande, vous pouvez raccourcir légèrement votre recherche

```
X-Sieve: CMU Sieve 2.3
X-DSPAM-Result: Innocent
X-Plane is behind schedule: two weeks
```



Rechercher et exporter des données

- Le re.search() renvoie un Vrai/Faux selon si la chaîne correspond à l'expression régulière
- Si nous voulons réellement extraire les chaînes correspondantes, nous utilisons re.findall()




```
>>> import re
>>> x = 'My 2 favorite numbers are 19 and 42'
>>> y = re.findall('[0-9]+',x)
>>> print y
['2', '19', '42']
```

Rechercher et exporter des données

- Lorsque nous utilisons re.findall(), il retourne une liste de zéro ou plusieurs chaînes secondaires qui correspondent à l'expression régulière

```
>>> import re
>>> x = 'My 2 favorite numbers are 19 and 42'
>>> y = re.findall('[0-9]+',x)
>>> print y
['2', '19', '42']
>>> y = re.findall('[AEIOU]+',x)
>>> print y
[]
```

Attention: recherche vorace

- Les caractères (* et +) répétés déplacent vers l'extérieur dans les deux sens (vorace) afin de correspondre à la plus grande chaîne possible

```
>>> import re
>>> x = 'From: Using the : character'
>>> y = re.findall('^F.+:', x)
>>> print y
['From: Using the :']
```

Pourquoi pas 'From:' ?

Un ou plusieurs caractères

^F.+:

Premier caractère, correspondant à un F

Dernier caractère correspondant à un :

Recherche Non-Vorace

- Les expressions régulières qui répètent du code ne sont pas toutes voraces! Si vous ajoutez un caractère ? - le + et * se détendent un peu...

```
>>> import re
>>> x = 'From: Using the : character'
>>> y = re.findall('^F.+?:', x)
>>> print y
['From:']
```

Un ou plusieurs caractères mais non vorace

^F.+?:

Premier caractère, correspondant à un F

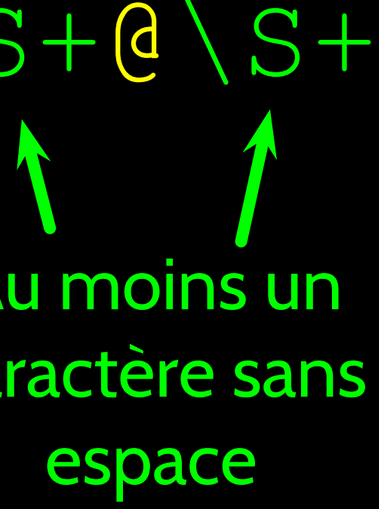
Dernier caractère correspondant à un :

Améliorer l'extraction de chaînes

- Vous pouvez affiner votre recherche avec re.findall() et déterminer séparément quelle portion de celle-ci est à extraire en utilisant des parenthèses

```
From stephen.marquard@uct.ac.za Sat Jan  5 09:14:16 2008
```

```
>>> y = re.findall('\S+@\S+',x)
>>> print y
['stephen.marquard@uct.ac.za']
```



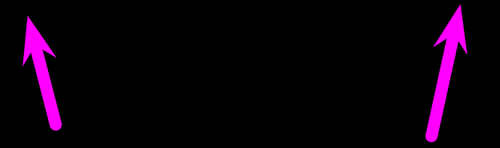
Améliorer l'extraction de chaînes

- Les parenthèses ne font pas partie de la recherche - mais elles précisent où commence et termine la chaîne à extraire

```
From stephen.marquard@uct.ac.za Sat Jan  5 09:14:16 2008
```

```
>>> y = re.findall('\S+@\S+',x)
>>> print y
['stephen.marquard@uct.ac.za']
>>> y = re.findall('^From:.*? (\S+@\S+)',x)
>>> print y
['stephen.marquard@uct.ac.za']
```

```
^From (\S+@\S+)
```




```
>>> data = 'From stephen.marquard@uct.ac.za Sat Jan  5 09:14:16 2008'
>>> atpos = data.find('@')
>>> print atpos
21
>>> sppos = data.find(' ',atpos)
>>> print sppos
31
>>> host = data[atpos+1 : sppos]
>>> print host
uct.ac.za
```

Extrait le nom d'hôte - à l'aide de la recherche et la coupe de chaînes

Le modèle de double Division

Parfois nous divisons une ligne d'une façon, et puis nous prenons une des parties de la ligne et la divisons à nouveau

```
From stephen.marquard@uct.ac.za Sat Jan  5 09:14:16 2008
```

```
words = line.split()
email = words[1]
pieces = email.split('@')
print pieces[1]
```

```
stephen.marquard@uct.ac.za
['stephen.marquard', 'uct.ac.za']
'uct.ac.za'
```

La Version Regex

```
From stephen.marquard@uct.ac.za Sat Jan  5 09:14:16 2008
```

```
import re
lin = 'From stephen.marquard@uct.ac.za Sat Jan  5 09:14:16 2008'
y = re.findall('@([^ ]*)',lin)
print y['uct.ac.za']
```

'@([^]*)'

Parcourir la chaîne jusqu'à ce que vous trouviez un arobase

La Version Regex

```
From stephen.marquard@uct.ac.za Sat Jan  5 09:14:16 2008
```

```
import re
lin = 'From stephen.marquard@uct.ac.za Sat Jan  5 09:14:16 2008'
y = re.findall('@([^ ]*)',lin)
print y['uct.ac.za']
```

La Version Regex

```
From stephen.marquard@uct.ac.za Sat Jan  5 09:14:16 2008
```

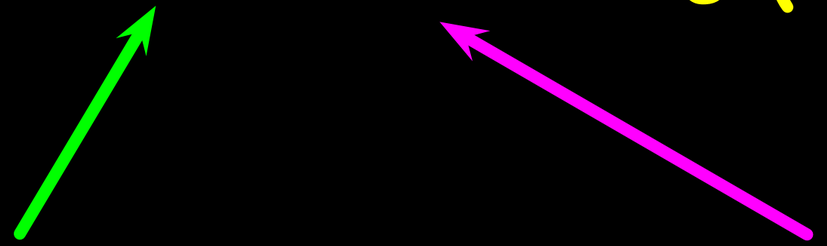
```
import re
lin = 'From stephen.marquard@uct.ac.za Sat Jan  5 09:14:16 2008'
y = re.findall('@([^ ]*)',lin)
print y['uct.ac.za']
```

Une Version Regex encore plus cool

```
From stephen.marquard@uct.ac.za Sat Jan  5 09:14:16 2008
```

```
import re
lin = 'From stephen.marquard@uct.ac.za Sat Jan  5 09:14:16 2008'
y = re.findall('^From .*@([^ ]*)',lin)
print y['uct.ac.za']
```

'^From .*@([^]*)'



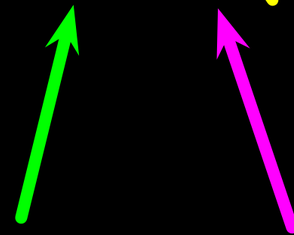
Commence au début de la ligne, recherche la chaîne 'From '

Une Version Regex encore plus cool

```
From stephen.marquard@uct.ac.za Sat Jan  5 09:14:16 2008
```

```
import re
lin = 'From stephen.marquard@uct.ac.za Sat Jan  5 09:14:16 2008'
y = re.findall('^From .*@([^ ]*)',lin)
print y['uct.ac.za']
```

'^From .*@([^]*)'



Saute un tas de caractères, recherche le symbol “at”

Une Version Regex encore plus cool

```
From stephen.marquard@uct.ac.za Sat Jan  5 09:14:16 2008
```

```
import re
lin = 'From stephen.marquard@uct.ac.za Sat Jan  5 09:14:16 2008'
y = re.findall('^From .*@([^ ]*)',lin)
print y['uct.ac.za']
```

'^From .*@([^]*)'

Commence l'extraction

Une Version Regex encore plus cool

```
From stephen.marquard@uct.ac.za Sat Jan  5 09:14:16 2008
```

```
import re
lin = 'From stephen.marquard@uct.ac.za Sat Jan  5 09:14:16 2008'
y = re.findall('^From .*@([^ ]*)',lin)
print y['uct.ac.za']
```

'^From .*@([^]*)'

Recherche des caractères non vide

Recherche beaucoup d'entre eux

Une Version Regex encore plus cool

```
From stephen.marquard@uct.ac.za Sat Jan  5 09:14:16 2008
```

```
import re
lin = 'From stephen.marquard@uct.ac.za Sat Jan  5 09:14:16 2008'
y = re.findall('^From .*@([^ ]*)',lin)
print y['uct.ac.za']
```

'^From .*@([^]*)'

Termine l'extraction

Confiance contre le Spam

```
import re
hand = open('mbox-short.txt')
numlist = list()
for line in hand:
    line = line.rstrip()
    stuff = re.findall('^X-DSPAM-Confidence: ([0-9.]+)', line)
    if len(stuff) != 1 :  continue
    num = float(stuff[0])
    numlist.append(num)
print 'Maximum:', max(numlist)
```

```
python ds.py
Maximum: 0.9907
```

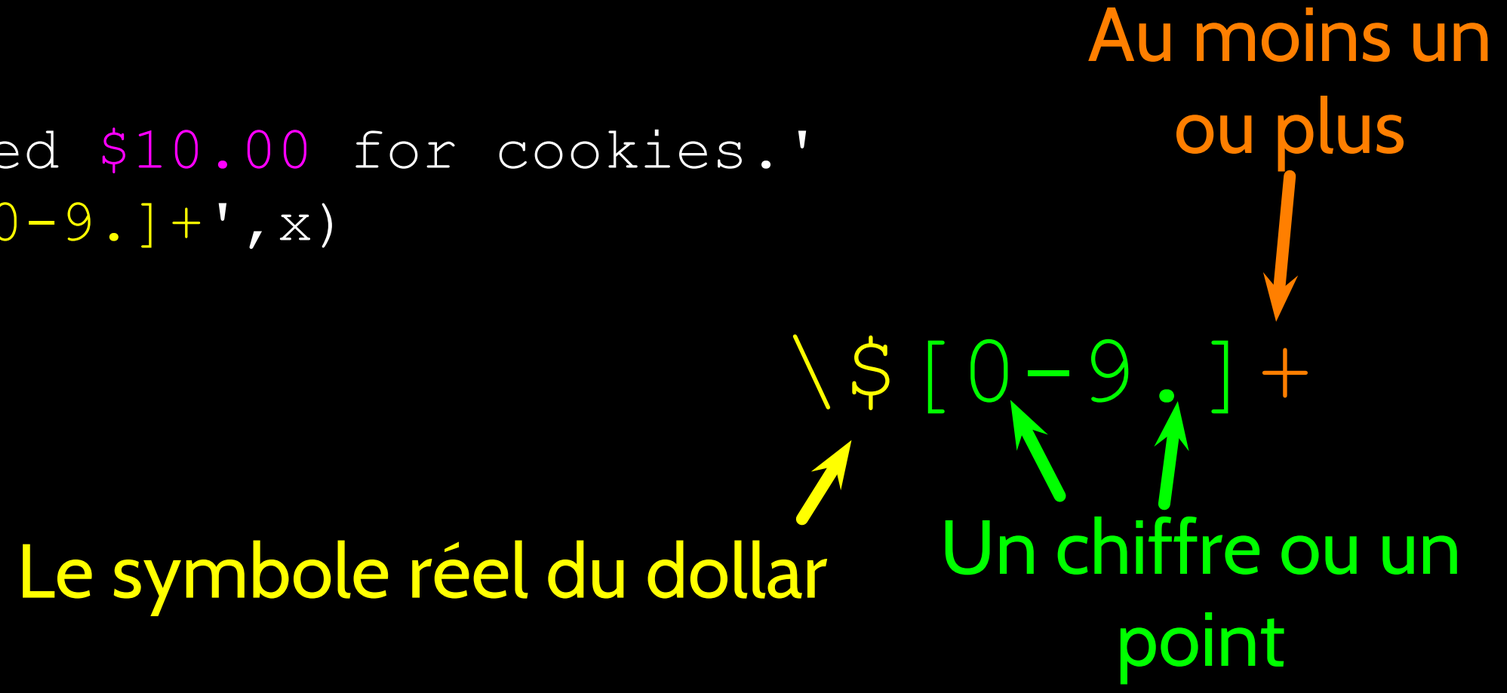
Guide rapide des expressions régulières

^	Correspond au début d'une ligne
$	Correspond à la fin d'une ligne
.	Correspond à n'importe quel caractère
\s	Correspond à un espace
\S	Correspond à n'importe quel caractère sans espace
*	Répète un caractère zéro ou plusieurs fois
*?	Répète un caractère zéro ou plusieurs fois (non vorace)
+	Répète un caractère une ou plusieurs fois
+?	Répète un caractère une ou plusieurs fois (non vorace)
[aeiou]	Correspond à un caractère unique dans une liste
[^XYZ]	Correspond à un caractère unique qui n'est pas dans une liste
[a-z0-9]	Le jeu de caractères peut inclure un éventail
(	Indique où l'extraction de chaîne doit commencer
)	Indique où l'extraction de chaîne doit se terminer

Caractère d'échappement

- Si vous souhaitez qu'un caractère spécial d'expression régulière fonctionne normalement (la plupart du temps) il faut ajouter le préfixe '\'

```
>>> import re
>>> x = 'We just received $10.00 for cookies.'
>>> y = re.findall('\$[0-9.]+',x)
>>> print y
['$10.00']
```



Résumé

- Les expressions régulières sont un langage cryptique mais puissant à utiliser avec les chaînes de caractère, qui les recherche et extrait des éléments de ces chaînes
- Les expressions régulières ont des caractères spéciaux caractères qui indique leur but

Acknowledgements / Contributions

Initial Development: Charles Severance, University of Michigan School of Information

Translation: Frederic Foiry

...