

# Expressions régulières

## Chapitre 11



Python for Informatics: Exploring Information  
[www.pythonlearn.com](http://www.pythonlearn.com)



# Expressions régulières

En informatique, une expression régulière, aussi dénommée “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](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](http://fr.wikipedia.org/wiki/Expression_rationnelle)

W Expression rationnelle — V x

fr.wikipedia.org/wiki/Expression\_rationnelle

Créer un compte Se connecter

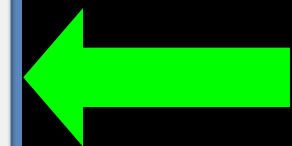
Article Discussion Lire Modifier Modifier le code Historique expression régulière

# Expression rationnelle

*Pour les articles homonymes, voir régulier et rationnel.*

Une **expression rationnelle** (ou **expression régulière** par traduction de l'anglais *regular expression*) est en informatique une chaîne de caractères que l'on appelle parfois un motif et qui décrit un ensemble de chaînes de caractères possibles selon une syntaxe précise. Les expressions rationnelles sont issues des théories mathématiques des langages formels des années 1940. Leur puissance à décrire des ensembles réguliers explique qu'elles se retrouvent dans plusieurs domaines scientifiques dans les années d'après-guerre et justifie leur adoption en informatique. Les expressions rationnelles sont aujourd'hui utilisées par les informaticiens dans l'édition et le contrôle de texte ainsi que dans la manipulation des langues formelles que sont les langages de l'informatique.

Sommaire [masquer]



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

WHENEVER I LEARN A NEW SKILL I CONCOCT ELABORATE FANTASY SCENARIOS WHERE IT LETS ME SAVE THE DAY.

OH NO! THE KILLER MUST HAVE FOLLOWED HER ON VACATION!



BUT TO FIND THEM WE'D HAVE TO SEARCH THROUGH 200 MB OF EMAILS LOOKING FOR SOMETHING FORMATTED LIKE AN ADDRESS!



IT'S HOPELESS!

EVERYBODY STAND BACK.



I KNOW REGULAR EXPRESSIONS.



<http://xkcd.com/208/>

# Guide rapide des expressions régulières

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

Correspond au début  
de la ligne

Plusieurs  
fois

Correspond à tout caractère

^ X . \* :

# 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
```

Correspond au début  
de la ligne

Plusieurs  
fois

Correspond à tout caractère

^ X . \* :

# 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

Correspond au début  
de la ligne

Plusieurs  
fois

^X-\\S+ :

Correspond à tout caractère

# 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()`

`[0-9]+`



Un ou plusieurs chiffres

```
>>> 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:' ?

Premier caractère,  
correspondant à un F

Dernier caractère  
correspondant à un :

Un ou plusieurs  
caractères

^ F . + :

# 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']
```

`\S+@\S+`



Au moins un  
caractère sans  
espace

# 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
```

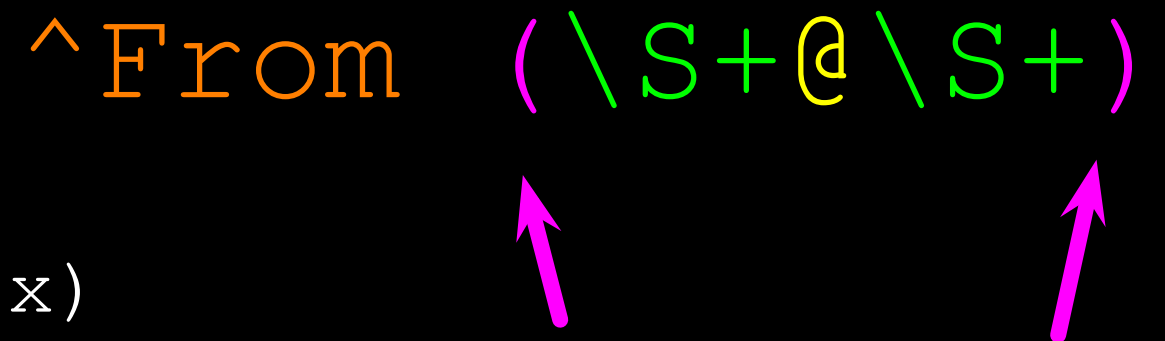
```
['stephen.marquard@uct.ac.za']
```

```
>>> y = re.findall('^From:.*? (\S+@\S+)', x)
```

```
>>> print y
```

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

^From (\S+@\S+)

The diagram shows the regex pattern `^From (\S+@\S+)`. The text `^From` is in orange. The opening parenthesis `(` is in pink. The text `\S+@\S+` is in green. The closing parenthesis `)` is in pink. Two pink arrows point upwards from the opening and closing parentheses to the text `stephen.marquard@uct.ac.za` in the output of the first code block, illustrating that the parentheses define the boundaries of the captured group.

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

21 ↓ 31 ↓

```
>>> 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']
```

'@([ ^ ]\*)'

Recherche des caractères non vide Recherche beaucoup d'  
entre eux



# 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('@([^\s]*)', lin)  
print y['uct.ac.za']
```

'@([^\s]\*)'

Extrait les caractères non vide

# 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']
```

'<sup>^</sup>From .\*@([ ^ ]\*)'

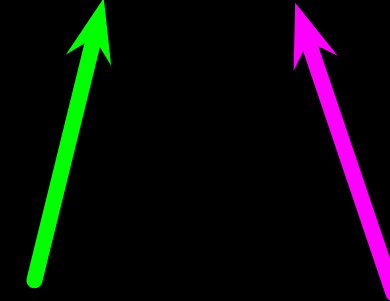
Commence au début de la ligne, recherche la chaîne 'From'

# 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

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

Le symbole réel du dollar

Un chiffre ou un point

Au moins un  
ou plus

`\$ [0-9.] +`

# 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



These slides are Copyright 2010- Charles R. Severance ([www.dr-chuck.com](http://www.dr-chuck.com)) of the University of Michigan School of Information and [open.umich.edu](http://open.umich.edu) and made available under a Creative Commons Attribution 4.0 License. Please maintain this last slide in all copies of the document to comply with the attribution requirements of the license. If you make a change, feel free to add your name and organization to the list of contributors on this page as you republish the materials.

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

Translation: Frederic Foiry