

Regular Expressions / Κανονικές Εκφράσεις

Κεφάλαιο 11



Python για Όλους
www.py4e.com



Κανονικές Εκφράσεις

Στους υπολογιστές, μια κανονική έκφραση, που αναφέρεται επίσης ως «regex» ή «regexpr», παρέχει ένα συνοπτικό και ευέλικτο μέσο για την αντιστοίχιση συμβολοσειρών, όπως συγκεκριμένους χαρακτήρες, λέξεις ή μοτίβα χαρακτήρων. Μια κανονική έκφραση γράφεται σε μια επίσημη γλώσσα που μπορεί να ερμηνευτεί από έναν επεξεργαστή κανονικών εκφράσεων.

http://en.wikipedia.org/wiki/Regular_expression

Κανονικές Εκφράσεις

Πραγματικά έξυπνες εκφράσεις με
«μπαλαντέρ» για αντιστοίχιση και ανάλυση
συμβολοσειρών

http://en.wikipedia.org/wiki/Regular_expression

Regular expression - Wikipedia, the free encyclopedia

W http://en.wikipedia.org/wiki/Regular_expression Reader Google

More than 100 matches regular Done

Log in / create account

Article Discussion Read Edit View history Search

Regular expression

From Wikipedia, the free encyclopedia

In **computing**, a **regular expression**, also referred to as **regex** or **regexp**, provides a concise and flexible means for matching **strings** of text, such as particular characters, words, or patterns of characters. A regular expression is written in a **formal language** that can be interpreted by a regular expression processor, a program that either serves as a **parser generator** or examines text and identifies parts that match the provided **specification**.

The following examples illustrate a few specifications that could be expressed in a regular expression:

- The sequence of characters "car" appearing consecutively in any context, such as in "car", "cartoon", or "bicarbonate"
- The sequence of characters "car" occurring in that order with other characters between them, such as in "Icelander" or "chandler"

Πραγματικά έξυπνη «Εύρεση» ή «Αναζήτηση»

Κατανόηση των Κανονικών Εκφράσεων

- Πολύ ισχυρές και αρκετά αινιγματικές
- Διασκεδαστικές άπαξ τις καταλάβετε
- Οι κανονικές εκφράσεις είναι μια γλώσσα από μόνες τους
- Μια γλώσσα «χαρακτήρων δεικτών» - προγραμματισμός με χαρακτήρες
- Είναι μια γλώσσα «παλιάς σχολής» - συμπαγής

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!

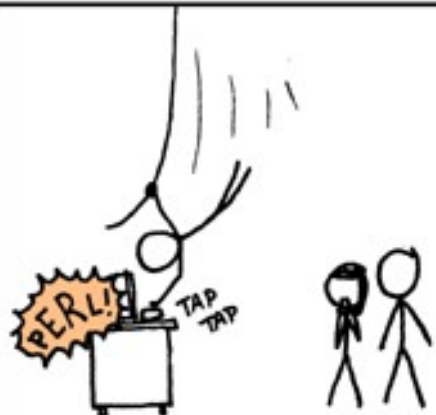
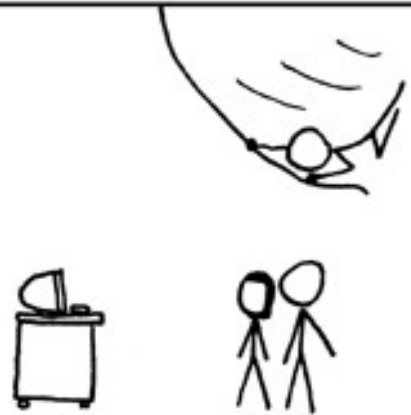


IT'S HOPELESS!

EVERYBODY STAND BACK.



I KNOW REGULAR EXPRESSIONS.



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

Σύντομος Οδηγός Κανονικών Εκφράσεων

<code>^</code>	Ταιριάζει την αρχή μιας γραμμής
<code>\$</code>	Ταιριάζει το τέλος μιας γραμμής
<code>.</code>	Ταιριάζει οποιοδήποτε χαρακτήρα
<code>\s</code>	Ταιριάζει κενό διάστημα (μη ορατό χαρακτήρα)
<code>\S</code>	Ταιριάζει οποιοδήποτε μη-κενό διάστημα (ορατό χαρακτήρα)
<code>*</code>	Επαναλαμβάνει ένα χαρακτήρα καμία ή περισσότερες φορές
<code>*?</code>	Επαναλαμβάνει ένα χαρακτήρα καμία ή περισσότερες φορές (μη-άπληστα)
<code>+</code>	Επαναλαμβάνει ένα χαρακτήρα μία ή περισσότερες φορές
<code>+?</code>	Επαναλαμβάνει ένα χαρακτήρα μία ή περισσότερες φορές (μη-άπληστα)
<code>[aeiou]</code>	Ταιριάζει έναν μόνο χαρακτήρα από το δοθέν σύνολο
<code>[^XYZ]</code>	Ταιριάζει έναν μόνο χαρακτήρα που δεν περιέχεται στο δοθέν σύνολο
<code>[a-z0-9]</code>	Το σύνολο των χαρακτήρων μπορεί να περιλαμβάνει ένα εύρος/διάστημα
<code>(</code>	Υποδεικνύει από πού θα ξεκινήσει η εξαγωγή της συμβολοσειράς
<code>)</code>	Υποδεικνύει πού θα τελειώσει η εξαγωγή της συμβολοσειράς

Η Ενότητα Κανονική Έκφραση

- Πριν χρησιμοποιήσετε τις κανονικές εκφράσεις στο πρόγραμμά σας, πρέπει να εισαγάγετε την αντίστοιχη βιβλιοθήκη χρησιμοποιώντας το «`import re`»
- Μπορείτε να χρησιμοποιήσετε το `re.search()` για να δείτε αν μια συμβολοσειρά ταιριάζει με μια κανονική έκφραση, παρόμοια με τη χρήση της μεθόδου `find()` για συμβολοσειρές
- Μπορείτε να χρησιμοποιήσετε `re.findall()` για την εξαγωγή τμημάτων μιας συμβολοσειράς που ταιριάζει με κανονική έκφραση σας, παρόμοια με ένα συνδυασμό `find()` και τεμαχισμού: `var[5:10]`

Χρήση του `re.search()` Αντί του `find()`

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

```
import re

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

Χρήση του `re.search()` Αντί του `startswith()`

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

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

Ρυθμίζουμε με ακρίβεια αυτό που ταιριάζει προσθέτοντας ειδικούς χαρακτήρες στη συμβολοσειρά

Χαρακτήρες Μπαλαντέρ

- Ο χαρακτήρας **τελεία** ταιριάζει με οποιονδήποτε χαρακτήρα
- Αν προσθέσετε τον **αστερίσκο**, ο χαρακτήρας είναι «όσες φορές»

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

Ταιριάζει την αρχή της γραμμής

Πολλές φορές

Ταιριάζει οποιονδήποτε χαρακτήρα

^ X . * :

Βελτιστοποιήστε το Ταίριασμα

Ανάλογα με το πόσο «καθαρά» είναι τα δεδομένα σας και τον σκοπό της εφαρμογής σας, μπορεί να θέλετε να περιορίσετε λίγο την αντιστοίχιση

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

Ταιριάζει την
αρχή της
γραμμής

Πολλές
φορές

^ X . * :

Ταιριάζει
οποιοδήποτε
χαρακτήρα

Βελτιστοποιήστε το Ταίριασμα

Ανάλογα με το πόσο «καθαρά» είναι τα δεδομένα σας και τον σκοπό της εφαρμογής σας, μπορεί να θέλετε να περιορίσετε λίγο την αντιστοίχιση

`X-Sieve: CMU Sieve 2.3`

`X-DSPAM-Result: Innocent`

`X-: Very Short`

`X-Plane is behind schedule: two weeks`

Ταιριάζει την
αρχή της
γραμμής

Μία ή
περισσότερες
φορές

`^X-\S+:`

Ταιριάζει οποιονδήποτε μη-κενό
χαρακτήρα

Ταίριασμα και Εξαγωγή Δεδομένων

- `re.search()` επιστρέφει Αληθές/Ψευδές ανάλογα με το αν η συμβολοσειρά ταιριάζει με την κανονική έκφραση
- Στην πραγματικότητα εάν θέλουμε να εξάγονται οι συμβολοσειρές που ταιριάζουν, χρησιμοποιούμε το `re.findall()`

`[0-9]+`

```
>>> import re
>>> x = 'Τα 2 αγαπημένα μου νούμερα είναι το 19 και το 42'
>>> y = re.findall('[0-9]+', x)
>>> print(y)
['2', '19', '42']
```

Ένα ή περισσότερα
ψηφία

Αντιστοίχιση και Εξαγωγή Δεδομένων

Όταν χρησιμοποιούμε το `re.findall()`, μας επιστρέφει μια λίστα με καμία ή περισσότερες υπο-συμβολοσειρές που ταιριάζουν με την κανονική έκφραση

```
>>> import re
>>> x = 'Τα 2 αγαπημένα μου νούμερα είναι το 19 και το 42'
>>> y = re.findall('[0-9]+', x)
>>> print(y)
['2', '19', '42']
>>> y = re.findall('[ΑΕΙΟΥ]+', x)
>>> print(y)
[]
```

Προειδοποίηση: Άπληστο

Ταίριασμα

Οι χαρακτήρες επανάληψης (* και +) ωθούν προς τα έξω και προς τις δύο κατευθύνσεις (άπληστοι) για να ταιριάξουν με τη μεγαλύτερη δυνατή συμβολοσειρά

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

Γιατί όχι 'From:' ;

Ο πρώτος χαρακτήρας που θα ταιριάξει είναι το F

Ένας ή περισσότεροι χαρακτήρες

^F

.

+

:

Ο τελευταίος χαρακτήρας που θα ταιριάξει είναι το :

Μη-Άπληστο Ταίριασμα

Δεν είναι άπληστοι όλοι οι κωδικοί επανάληψης των κανονικών εκφράσεων! Αν προσθέσετε ένα χαρακτήρα **?**, το **+** και το ***** χαλαρώνουν λίγο...

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

Ο πρώτος χαρακτήρας που θα ταιριάξει είναι το F

^F . **+** **?** **:**

Ο τελευταίος χαρακτήρας που θα ταιριάξει είναι το :

Ένας ή περισσότεροι χαρακτήρες αλλά μη άπληστο

Καλύτερη Ρύθμιση Εξαγωγής Συμβολοσειρών

Μπορείτε να βελτιώσετε την αντιστοίχιση του `re.findall()` και να καθορίσετε ξεχωριστά ποιο τμήμα της αντιστοίχισης πρόκειται να εξαχθεί χρησιμοποιώντας παρενθέσεις

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+`
↑ ↑
Τουλάχιστον
έναν μη κενό
χαρακτήρα

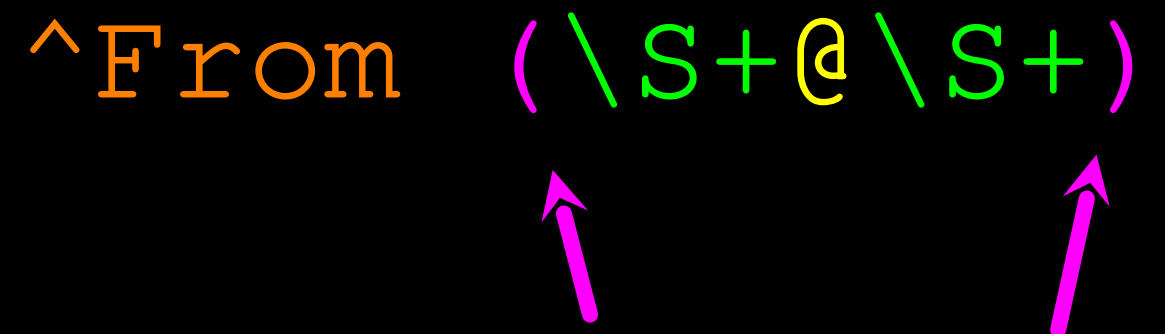
Καλύτερη Ρύθμιση Εξαγωγής Συμβολοσειρών

Οι **Παρενθέσεις** δεν αποτελούν τμήμα της αντιστοίχισης – αλλά λένε από να **αρχίσει** και που να **σταματήσει** η εξαγόμενη συμβολοσειρά

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+)`



Παραδείγματα Ανάλυσης Συμβολοσειρών ...

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

```
>>> 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
```

Εξαγωγή ονόματος
κεντρικού υπολογιστή -
χρησιμοποιώντας
εύρεση και κατάτμηση
συμβολοσειράς

Το Μοτίβο Διπλής Διάσπασης

Μερικές φορές χωρίζουμε μια γραμμή με έναν τρόπο και στη συνέχεια παίρνουμε ένα από τα κομμάτια που προέκυψαν και το χωρίζουμε ξανά

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

```
λέξεις = γραμμή.split()  
email = λέξεις[1]  
κομμάτια = email.split('@')  
print(κομμάτια[1])
```

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

Η Έκδοση με Κανονική Έκφραση

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

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

```
['uct.ac.za']
```

'@([^\s]*)'

Ψάξε στην συμβολοσειρά μέχρι να βρεις το σύμβολο @

Η Έκδοση με Κανονική Έκφραση

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

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

```
['uct.ac.za']
```

'@([^]*)'

Ταιριάζει με μη κενούς
χαρακτήρες

Ταιριάζει με
οσοσδήποτε από
αυτούς

Η Έκδοση με Κανονική Έκφραση

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

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

```
['uct.ac.za']
```

'@([^]*)'

Εξάγει τους μη κενούς χαρακτήρες

Ακόμη Καλύτερη Regex

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

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

```
['uct.ac.za']
```

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

Ξεκινά στην αρχή της γραμμής, ψάχνει για τη συμβολοσειρά
'From'

Ακόμη Καλύτερη Regex

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

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

```
['uct.ac.za']
```

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

Προσπερνά ένα σωρό χαρακτήρες, ψάχνει για το σύμβολο

@

Ακόμη Καλύτερη Regex

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

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

```
['uct.ac.za']
```

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

Αρχή της εξαγωγής

Ακόμη Καλύτερη Regex

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

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

['uct.ac.za']

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

Μη-κενοί χαρακτήρες

Ταιριάζει
οποιοδήποτε από
αυτούς

Ακόμη Καλύτερη Regex

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

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

```
['uct.ac.za']
```

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

↑
Τέλος εξαγωγής

Spam Confidence

```
import re
hand = open('mbox-short.txt')
λίστααριθμών = list()
for γραμμή in hand:
    γραμμή = γραμμή.rstrip()
    διάφορα = re.findall('^X-DSPAM-Confidence: ([0-9.]+)', γραμμή)
    if len(διάφορα) != 1 : continue
    αριθμός = float(διάφορα[0])
    λίστααριθμών.append(αριθμός)
print('Μεγαλύτερο:', max(λίστααριθμών))
```

X-DSPAM-Confidence: 0.8475

python ds.py

Μεγαλύτερο:

0.9907

Χαρακτήρας Διαφυγής

Εάν θέλετε ένας ειδικός χαρακτήρας μιας κανονικής έκφρασης να συμπεριφέρεται **κανονικά** (τις περισσότερες φορές) του προθέτετε το `'\'`

```
>>> import re
>>> x = 'Μόλις λάβαμε $10.00 για μπισκότα.'
>>> y = re.findall('\$[0-9.]+', x)
>>> print(y)
['$10.00']
```

Τουλάχιστον ένα
ή περισσότερα



`\$ [0-9.] +`

Ένα πραγματικό σύμβολο
δολαρίου

Ένα ψηφίο ή
υποδιαστολη

Σύνοψη

- Οι κανονικές εκφράσεις είναι μια αινιγματική αλλά ισχυρή γλώσσα για την αντιστοίχιση συμβολοσειρών και την εξαγωγή στοιχείων από αυτές τις συμβολοσειρές
- Οι κανονικές εκφράσεις έχουν ειδικούς χαρακτήρες που υποδηλώνουν πρόθεση



Ευχαριστίες / Συνεισφορές



Αυτές οι διαφάνειες είναι Πνευματική ιδιοκτησία 2010- Charles R. Severance (www.dr-chuck.com) του University of Michigan School of Information και είναι διαθέσιμες υπό την άδεια Creative Commons Attribution 4.0. Παρακαλώ να διατηρήσετε αυτήν την τελευταία διαφάνεια σε όλα τα αντίγραφα του εγγράφου για να συμμορφωθείτε με τις απαιτήσεις απόδοσης της άδειας. Εάν κάνετε κάποια αλλαγή, μη διστάσετε να προσθέσετε το όνομα και τον οργανισμό σας στη λίστα των συντελεστών αυτής της σελίδας καθώς αναδημοσιεύετε το υλικό.

Συνέχεια...

Αρχική ανάπτυξη : Charles Severance, University of Michigan School of Information

Απόδοση στα Ελληνικά: Κιουρτίδου Δ. Κωνσταντία

... Εισαγάγετε νέους Μεταφραστές και άτομα που έχουν συνεισφέρει εδώ