Regular Expressions

ΓΛΩΣΣΙΚΉ ΤΕΧΝΟΛΟΓΊΑ

Regular Expressions - γενικά

- Βασική ιδέα: έχουμε ένα pattern και ένα κείμενο εισόδου. Εφαρμόζουμε το pattern στο κείμενο και μπορούμε:
  - Να ελέγξουμε αν μέρος του κειμένου συμφωνεί με το pattern.
  - Να εξάγουμε κομμάτια του κειμένου που επιθυμούμε
  - Να αντικαταστήσουμε κομμάτια του κειμένου κλπ.
- Πολύ δυνατό εργαλείο διαχείρισης strings.
- Επιτρέπει τις λειτουργίες find και replace σε string, με πολύ περισσότερες δυνατότητες, όχι απλό string matching.
- Πολύ γρήγορα!!!

Regular Expressions #1

Pattern	
.	Ταιριάζει σε οποιονδήποτε χαρακτήρα. πχ το pattern a.c ταιριάζει στα strings abc, a_c, afc, a0c κλπ
^	Ταιριάζει στην αρχή του string. πχ το pattern ^abc ταιριάζει στα strings που ξεκινούν από abc
$	Ταιριάζει στο τέλος του string. πχ το pattern abc$ ταιριάζει στα strings που τελειώνουν με abc
*	Ταιριάζει σε 0 ή περισσότερες εμφανίσεις του pattern που προηγείται. πχ το pattern a* ταιριάζει στο άδειο string και στα a, aa, aaa, aaaa κλπ.
+	Ταιριάζει σε 1 ή περισσότερες εμφανίσεις του pattern που προηγείται. πχ το pattern a+ ταιριάζει στα a, aaa, aaaa, κλπ
?	Ταιριάζει σε 0 ή 1 εμφανίσεις του pattern που προηγείται. πχ το pattern a?bc ταιριάζει στα strings abc και bc
{m}	Ταιριάζει σε m εμφανίσεις του pattern που προηγείται. πχ το pattern a{3} ταιριάζει στο string aaa
{m,n}	Ταιριάζει σε m έως n εμφανίσεις του pattern που προηγείται. πχ το pattern a{1,3} ταιριάζει στα a, aa, aaa

Regular Expression #2

Pattern	
[]	Ταιριάζει σε σύνολο χαρακτήρων. πχ [0-9] είναι ένα οποιοδήποτε αριθμητικό ψηφίο. [a-zA-Z] ένα οποιοδήποτε γράμμα. [134] είναι ένα οποιοδήποτε ψηφίο από τα 1 3 ή 4. [0-9]+ ταιριάζει σε οποιονδήποτε ακέραιο [a-zA-Z]+ ταιριάζει σε οποιαδήποτε λέξη έχει μόνο γράμματα.
[^]	Ταιριάζει σε οτιδήποτε δεν περιέχεται στο σύνολο χαρακτήρων. πχ το [^0-9] ταιριάζει σε οτιδήποτε δεν είναι ψηφίο.
|	x|y ταιριάζει είτε στο x είτε στο y
( )	Ομαδοποίηση ενός υποσυνόλου του pattern. πχ (01)+ ταιριάζει στα strings 01,0101,010101 κλπ ενώ το 01+ ταιριάζει στα 01,011, 0111 κλπ
\	Escape character. Αν θέλω να συμπεριλάβω κάποιον ειδικό χαρακτήρα για την τιμή του, χρησιμοποιώ τον escape. πχ το \+ ταιριάζει στον χαρακτήρα + και όχι σε επαναλήψεις του προηγούμενου.

Regular Expressions #3

Pattern	
\t	Tab
\n	New line
\s	Ταιριάζει στα κενά. (whitespace, tab, new line κλπ)
\d	Ταιριάζει σε ψηφία. Ισοδύναμο με [0-9]
\w	Ταιριάζει σε ψηφία, γράμματα ή underscore. Ισοδύναμο με το [0-9a-zA-Z_]

Regular Expressions – look(ahead|behind)

Pattern	
(?=pattern)	Positive lookahead – ταιριάζει με το string που ακολουθείται από το pattern που περιγράφεται. πχ το readme(?=\.txt) ταιριάζει με το readme μόνο αν έχει κατάληξη .txt Προσοχή: αυτό που ταιριάζει είναι το readme! Το .txt δεν καταναλώνεται!
(?!pattern)	Negative lookahead – ταιριάζει με το string που δεν ακολουθείται από το pattern που περιγράφεται.
(?<=pattern)	Positive lookbehind – ταιριάζει με το string από το οποίο προηγείται το pattern που περιγράφεται. πχ. το (?<=readme) \.txt ταιριάζει με το .txt μόνο αν προηγείται το readme.
(?<!pattern)	Negative lookbehind – ταιριάζει με το string από το οποίο δεν προηγείται το pattern που περιγράφεται.

C# - Regex

- Namespace: System.Text.RegularExpressions
- Η κλάση Regex υποστηρίζει τη λειτουργικότητα των regular expressions
- Βασικά μέλη:
  - IsMatch: επιστρέφει true ή false ανάλογα με το αν το input ταιριάζει με το pattern ή όχι
  - Match: επιστρέφει και το μέρος του input που κάνει match (σε αντικείμενο της κλάσης Match)
  - Matches: επιστρέφει συλλογή από matches (σε αντικείμενο της κλάσης MatchCollection).

C# - Regex Example

```
public static void RegexpExample()
{
    string input = "Hello, my name is Vivi.";
    string pattern = "^Hello";

    if (Regex.IsMatch(input, pattern))
        Console.WriteLine("Hi!");

    pattern = "[^ \\.]+(?=\\.)"; //escaping . character
    //pattern = "[^ \\.]+\\." captures "Vivi."
    //lookahead pattern captures only "Vivi"
    Match m = Regex.Match(input, pattern);
    Console.WriteLine("nice to meet u " + m.Groups[0].Value);

    pattern = "[^ \\.,]+";
    MatchCollection mc = Regex.Matches(input, pattern);
    foreach (Match token in mc)
    {
        Console.WriteLine(token.Groups[0].Value);
    }

    /*prints
     Hello
     my
     name
     is
     Vivi
    */
}
```

C# string literals

- Το pattern \\\\ ταιριάζει με τον χαρακτήρα \. Έχει έναν escape character για τη σύνταξη των regular expressions και επιπλέον escape character για την αποθήκευσή του σε string.
- Λύση: raw strings. Με την αναπαράσταση των string literals με @"" δεν χρειάζεται να κάνουμε escape τους ειδικούς χαρακτήρες.

C# - escape hell

```
public static void EscapeHellExample()
{
    string input = "C:\\test\\temp\\readme.txt";
    string pattern = "txt$";

    Match m = Regex.Match(input, pattern);
    Console.WriteLine(m.Groups[0].Value);
    //Prints txt


    pattern = "[^\\\\]+";
    MatchCollection mc = Regex.Matches(input, pattern);
    foreach (Match step in mc)
    {
        Console.WriteLine(step.Groups[0].Value);
    }
    /*prints
     C:
     test
     temp
     readme.txt
    */
}
```

C# - raw strings

```
public static void EscapeHellExample()
{
    string input = @"C:\test\temp\readme.txt";
    string pattern = "txt$";

    Match m = Regex.Match(input, pattern);
    Console.WriteLine(m.Groups[0].Value);
    //Prints txt


    pattern = @"[^\\]+";
    MatchCollection mc = Regex.Matches(input, pattern);
    foreach (Match step in mc)
    {
        Console.WriteLine(step.Groups[0].Value);
    }
    /*prints
     C:
     test
     temp
     readme.txt
    */
}
```

C# - Named Groups

- Με τη χρήση του pattern (?<a_name>pattern) μπορούμε να ομαδοποιήσουμε και να ονομάσουμε ένα μέρος του string εισόδου, το οποίο μπορούμε να ανακτήσουμε με το όνομά του.

```
public static void NamedGroupExample()
{
    string input = @"C:\test\temp\readme.txt";
    string pattern = @"(?<path>([^\\]+\\)+)(?<filename>.+)";

    Match m = Regex.Match(input, pattern);
    Console.WriteLine(m.Groups["path"].Value);
    //prints C:\test\temp\
    Console.WriteLine(m.Groups["filename"].Value);
    //prints readme.txt
}
```

Python – re module

- Module: re
- Υποστηρίζει:
  - Search
  - match
  - groups
  - named groups.
- Υποστηρίζονται επίσης τα raw strings με τη μορφή: pattern = r"\d+\.\d*"

Python - example

- **>>> import re**
- **>>>** m = re.search('(?<=abc)def', 'abcdef')
- **>>>** m.group(o)
  'def'
- **>>>** m = re.search('(?<=-)\w+', 'spam-egg')
- **>>>** m.group(o)
  'egg'

Python match vs search

- Υποστηρίζονται δύο τρόποι ψαξίματος:
  - re.search : ψάχνει σε οποιοδήποτε μέρος του string
  - re.match : ψάχνει μόνο στην αρχή του string
- **>>>** re.match("c", "abcdef") *# No match*
- **>>>** re.search("c", "abcdef") *# Match* <_sre.SRE_Match object at …>

Python – named groups

- Με τη χρήση του pattern (?P<a_name>pattern) η python υποστηρίζει named groups.
- **>>>** m = re.match(r"(?P<first_name>\w+) (?P<last_name>\w+)", "Malcom Reynolds")
- **>>>** m.group('first_name')
  'Malcom'
- **>>>** m.group('last_name')
  'Reynolds'

Regular Expressions - more

- Και οι 2 γλώσσες υποστηρίζουν επιπλέον μέσω regular expressions:
  - split σε input strings
  - replace
  - Επιλογές για το αν η είσοδος θα είναι multiline, το encoding κλπ.
  - …