Ελληνικό Μεσογειακό Πανεπιστήμιο
Σχολή Μηχανικών / Τμήμα Ηλεκτρολόγων Μηχανικών & Μηχανικών Υπολογιστών

HMY01K06
Επιστημονικός Προγραμματισμός με Python

7 +1 ερωτήσεις επανάληψης για τη Διάλεξη 10

Κανονικές εκφράσεις

Φθινόπωρο 2025

Ερώτηση 1/7

Ποιοι είναι οι ειδικοί χαρακτήρες σε μια Κανονική Έκφραση;

[ ]	class	Ορίζουν μια κλάση χαρακτήρων (character class) μέσα σε μια Κ.Ε.
{ }	repeat	Καθορίζουν το πόσες φορές μπορεί να επαναλαμβάνεται ο προηγούμενος χαρακτήρας, κλάση ή ομάδα χαρακτήρων
? + *		Ειδικοί χαρακτήρες για συντομογραφία απλών επαναλήψεων
^	start	Ταίριασμα μόνο αν η συμβολοσειρά "αρχίζει από"
$	end	Ταίριασμα μόνο αν η συμβολοσειρά "τελειώνει σε"
.	any	Οποιοσδήποτε χαρακτήρας εκτός από NEWLINE (/n)
|	or	Συνένωση δυο κανονικών εκφράσεων
( )	capturing groups	Ορίζουν περιοχές της Κ.Ε. για τμηματική επιστροφή κατά το ταίριασμα
/	/x (Esc chars)	Ειδικοί χαρακτήρες διαφυγής. *(Το x παίρνει μόνο συγκεκριμένες τιμές)*

Ερώτηση 2/7

Πως ορίζονται οι κλάσεις χαρακτήρων σε μια Κ.Ε. ;

Ορίζονται μέσα σε [ ] και χρησιμοποιούνται όταν θέλουμε να δηλώσουμε ότι ενας χαρακτήρας σε μια Κ.Ε. μπορεί να είναι *οποιοσδήποτε* από μια συγκεκριμένη *συλλογή χαρακτήρων*

Πχ. η Κ.Ε. `'χά[πλδ]ι'` "ταιριάζει με" *(εντοπίζει)* τις συμβολοσειρές `'χάπι' 'χάλι' 'χάδι'`

Μια κλάση μπορεί να περιέχει είτε μεμονωμένους χαρακτήρες πχ. `[asdf]` είτε περιοχές χαρακτήρων *(από - έως)* πχ. `[a-k]`, είτε οποιονδήποτε συνδυασμό τους

Πχ. `[a-zA-Z0-9]` *(όλοι οι Αγγλικοί αλφαριθμητικοί χαρακτήρες – κεφαλαία & πεζά)*

Παράδειγμα: η Κ.Ε. `'test[1-4!]0'` "ταιριάζει με" *(μας εντοπίζει)* τις συμβολοσειρές `'test10' 'test20' 'test30' 'test40'` και `'test!0'`

Ερώτηση 3/7

Τι γνωρίζετε για τους ειδικούς χαρακτήρες επανάληψης { } σε μια Κ.Ε.

Τοποθετούνται αμέσως μετά έναν χαρακτήρα ή μια κλάση [ ] για να ορίσουν μια *επανάληψη*, δηλαδή το πόσες φορές μπορεί να επαναλαμβάνεται

Γενικά `x{n1,n2}` σημαίνει ότι ο χαρακτήρας *(ή η κλάση)* `x` μπορεί να επαναλαμβάνεται από `n1` μέχρι *και* `n2` φορές

Αν στις { } : παραλείψουμε *το κάτω όριο*, εννοείται το *μηδέν*, παραλείψουμε *το πάνω όριο* εννοείται το *άπειρο*

Ειδικοί χαρακτήρες επανάληψης ? + και *

`?` : Συντομογραφία του `{0,1}`

Επανάληψη του χαρακτήρα, της κλάσης ή του τμήματος 0 ή 1 φορές. (*)

`+` : Συντομογραφία του `{1,}`

Επανάληψη του χαρακτήρα, της κλάσης ή του τμήματος 1 ή περισσότερες φορές

`*` : Συντομογραφία του `{0,}`

Επανάληψη του χαρακτήρα, της κλάσης ή του τμήματος 0 ή περισσότερες φορές

Ερώτηση 4/7

Τι γνωρίζετε για τις παρενθέσεις (capturing groups) σε μια Κ.Ε.

Βάζοντας παρενθέσεις ( ) σε ένα -ή και περισσότερα- τμήματα μιας Κ.Ε. *δεν αλλάζει σε κάτι ο τρόπος λειτουργίας της Κ.Ε.*, όμως σε περίπτωση εντοπισμού μας επιστρέφει *μόνο εκείνα τα κομμάτια που αντιστοιχούν στις παρενθέσεις (capturing groups*)*

Επομένως μια Κ.Ε. με *παρενθέσεις* όπως η `'TH(209.5)'` αν εφαρμοστεί στην αρχική συμβολοσειρά, θα ταιριάσει *(εντοπίσει)* και πάλι το ίδιο τμήμα της , όμως αυτή τη φορά θα μας επιστρέψει *μόνο το κομμάτι που αντιστοιχεί στις παρενθέσεις*, δηλ. το `'20945'`

Αντίστοιχα μια Κ.Ε. με *δυο ζεύγη παρενθέσεων* όπως η `'(TH)20(9.5)'` θα μας επιστρέψει *μια πλειάδα με τα δυο κομμάτια* που αντιστοιχούν σε αυτές τις παρενθέσεις `('TH','945')`

*Τα capturing groups επιστρέφονται είτε μέσω της ιδιότητας `.group()` του αντικειμένου `matchobj` που επιστρέφουν οι συναρτήσεις `re.search()` και `re.match()`, είτε ως πλειάδες στη λίστα που επιστρέφει η `re.findall()`

Non-capturing groups: Απλές παρενθέσεις που δεν επιστρέφουν τμήμα της Κ.Ε. Ορίζονται προσθέτοντας *ερωτηματικό και άνω-κάτω τελεία* αμέσως μετα την αριστερή παρένθεση `'(?: έκφραση )'`

Ερώτηση 5/7

Ποιοι είναι οι χαρακτήρες διαφυγής στις Κ.Ε. της Python;

\w : (πεζό w) Ενας οποιοσδήποτε χαρακτήρας που είναι αγγλικό γράμμα (κεφαλαίο ή πεζό) , ψηφίο ή κάτω παύλα *(ισοδύναμο με την κλάση χαρακτήρων [a-zA-Z0-9_])*

\W : (Κεφαλαίο W) Ενας οποιοσδήποτε χαρακτήρας που *δεν ανήκει* στο προηγούμενο σύνολο \w

\s : (πεζό s) Ενας οποιοσδήποτε χαρακτήρας *τύπου whitespace* όπως το *κενό, tab, return, newline*

\S : (Κεφαλαίο S) Ενας οποιοσδήποτε χαρακτήρας που *δεν ανήκει* στο προηγούμενο σύνολο \s

\d : (πεζό d) Ενα οποιοδήποτε ψηφίο από 0...9 *(ισοδύναμο με την κλάση [0-9])*

\D : (Κεφαλαίο D) Ενας οποιοσδήποτε χαρακτήρας που *δεν ανήκει* στο προηγούμενο σύνολο \d

\t : Ταιριάζει *(εντοπίζει)* τον χαρακτήρα *tab* \b : Ταιριάζει *(εντοπίζει)* τον χαρακτήρα *backspace*

\n : Ταιριάζει *(εντοπίζει)* τον χαρακτήρα *newline* \r : Ταιριάζει *(εντοπίζει)* τον χαρακτήρα *return*

\A : Ταιριάζει *(εντοπίζει)* ένα μοτίβο στην αρχή μιας συμβολοσειράς (ισοδύναμο με ^) (*)

\Z : Ταιριάζει *(εντοπίζει)* ένα μοτίβο στο τέλος μιας συμβολοσειράς (ισοδύναμο με $) (*)

(*) μόνο στην περίπτωση που η συμβολοσειρά δεν εκτείνεται σε περισσότερες από γραμμές (multiline).

Ερώτηση 6/7

Ποιες οι *βασικότερες* συναρτήσεις του δομοστοιχείου re;

`re.search()`: Διατρέχει μια συμβολοσειρά, και εντοπίζει *το πρώτο σημείο της* που ταιριάζει με μια δοσμένη Κ.Ε. *Τα αποτελέσματα επιστρέφονται μέσω ενός αντικειμένου match object*

`re.match()`: Ελέγχει αν *η αρχή μιας συμβολοσειράς* ταιριάζει με μια δοσμένη Κ.Ε. *Τα αποτελέσματα επιστρέφονται μέσω ενός αντικειμένου match object*

`re.sub()`: διατρέχει μια συμβολοσειρά, και αφού εντοπίσει *όλα τα σημεία της* (substrings) που ταιριάζουν με μια δοσμένη Κ.Ε., τα αντικαθιστά με μια συμβολοσειρά της επιλογής του χρήστη και *επιστρέφει την τροποποιημένη αρχική συμβολοσειρά.*

`re.findall()`: Διατρέχει μια συμβολοσειρά, και εντοπίζει *όλα τα σημεία της* που ταιριάζουν με μια δοσμένη Κ.Ε. τα οποία επιστρέφει σαν μια *λίστα από συμβολοσειρές*. Αν η Κ.Ε. περιέχει δυο ή περισσότερα *capturing groups*, τότε κάθε στοιχείο της λίστας είναι μια πλειάδα συμβολοσειρών

`re.split()`: *Διαχωρίζει* (σπάει) μια συμβολοσειρά στα σημεία που ταιριάζουν με μια δοσμένη Κ.Ε. και *επιστρέφει μια λίστα* με τα κομμάτια που προέκυψαν από το σπάσιμο

Ερώτηση 7/7

Τι γνωρίζετε για το ταίριασμα greedy / non greedy ;

greedy: *άπληστος, αχόρταγος, αυτός που τα θέλει όλα δικά του*

	Default		
{a,b}	*Greedy matching.* Ο τελεστής επανάληψης θα προσπαθήσει να ταιριάσει *όσο περισσότερους χαρακτήρες μπορέσει*	{a,b}?	*Non greedy matching*. Ο τελεστής επανάληψης θα *σταματήσει να ψάχνει* μόλις επιτύχει το πρώτο ταίριασμα
*		*?	
+		+?	

```
# Greedy matching. Ταιριάζει ολόκληρη την συμβολοσειρά
a = re.findall( '<.*>',  '<H1> title </H1> <H1> subtitle </H1>')
print(a)
['<H1> title </H1> <H1> subtitle </H1>']
# Non-greedy matching. Ταιριάζει τέσσερα διαφορετικά τμήματα
a = re.findall( '<.*?>',  '<H1> title </H1> <H1> subtitle </H1>')
print(a)
['<H1>', '</H1>', '<H1>', '</H1>']
```

Bonus Ερώτηση 8

Δώστε ένα παράδειγμα μιας Κ.Ε. με σχόλια μέσα στον ορισμό της

```
import re
# Αυτή η Κ.Ε. ταιριάζει μια απλή διεύθυνση email
email_pattern = '''
    ^           # Start of the string
    [\w.-]+     # Username: one or more word characters, dots, or hyphens
    @           # At symbol
    [\w.-]+     # Domain: one or more word characters, dots, or hyphens
    $           # End of the string
'''

# Παράδειγμα χρήσης της παραπάνω Κ.Ε (το re.VERBOSE απαραίτητο)
example_email = "John_Doe_23@gmail.com"
if re.match(email_pattern, example_email, re.VERBOSE):
    print(f"Το {example_email} είναι ενα έγκυρο email.")
else:
    print(f"Το {example_email} δεν είναι  έγκυρο email.")
```

Όταν κατά την χρήση μιας Κ.Ε. είναι ενεργοποιημένο το flag `re.VERBOSE`,
τότε όλα τα *κενά*, *newlines* και *σχόλια τύπου Python* μέσα σε αυτήν την Κ.Ε. *αγνοούνται*