Μεταγλωτιστές 2020
Προγραμματιστική Εργασία #2

Ονοματεπώνυμο: Φουτσιτζή Σοφρονία
ΑΜ: Π2017063

Η υλοποίηση της συγκεκριμένης εργασίας έγινε σε γλώσσα προγραμματισμού Python και χρησιμοποιήθηκε αποκλειστικά η βιβλιοθήκη re των κανονικών εκφράσεων.
Αρχικά, στοχεύοντας στην υλοποίηση της εργασίας μελετήθηκε ιδιαίτερα η θεωρία αλλά και όλα τα εργαστήρια τα οποία έλαβαν χώρα κατά την διάρκεια του εξαμήνου και αφορούσαν την συγκεκριμένη βιβλιοθήκη και το σύνολο των κανονικών εκφράσεων. Έπειτα, ξεκίνησε κατά σειρά η δημιουργία κανονικών εκφράσεων για την υλοποίηση του προγράμματος.

Αρχικά, έγινε άνοιγμα του αρχείου στο οποίο περιέχεται το κείμενο σε html, το οποίο χρησιμοποιήθηκε, συγκεκριμένα, για ανάγνωση. Ένω έγινε άνοιγμα και ενός δεύτερου αρχείου (output.txt), που χρησιμοποιήθηκε για την εξαγωγή/ εκτύπωση των ζητουμένων.

Για την εξαγωγή και εκτύπωση του τίτλου της ιστοσελίδας χρησιμοποιήθηκε η ακόλουθη κανονική έκφραση.

>rexp = re.compile(r'(<title>)(.+?)</title>')

Συγκεκριμένα, η κανονική έκφραση εδώ βρίσκει ότι ξεκινάει με <title>, συνεχίζει με οποιοδήποτε χαρακτήρα, λόγω της τελείας “.”, τον οποίο αναζητά μία ή περισσότερες φορές λόγω του συν “+”. Ο χαρακτήρας αυτός ωστόσο είναι άπλειστος οπότε για την αποφυγή των πιθανών σφαλμάτων χρησιμοποιείται και ο τελεστής “?”, η λειτουργεία του οποίου είναι η αναζήτηση του οποιουδήποτε χαρακτήρα, στην προκειμένη περίπτωση, μηδέν ή παραπάνω φορές. Η αναζήτηση σταματάει όταν γίνει η εύρεση του </title>. Τα μέρη της κανονικής έκφρασης είναι διαχωρισμένα με παρενθέσεις , καθώς επιθυμούμε να εκτυπώσουμε ένα συγκεκριμένο μέρος της (τον τίτλο), κάτι που θα υλοποιηθεί με την χρήση group(). Για την εύρεση και εκτύπωση του τίτλου χρησιμοποιήθηκε μία δομή επανάληψης και η finditer().

Για το επόμενο ζητούμενο δηλαδή την απαλοιφή των σχολίων χρησιμοποίηθηκε, αρχικά η κανονική έκφραση:

>rexp1 = re.compile(r'<!--(.+?)-->',re.DOTALL)

η οποία αναζητά τα σχόλια (το re.DOTALL χρησιμοποιείται καθώς θέλουμε να γίνει αναζήτηση ακόμα και αν έχει newline) και με την χρήση της sub γίνεται η απαλοιφή, δηλαδή η αντικατάσταση του ό,τι βρίσκει η rexp1 με κενό. Αυτό που προκύπτει το “αποθηκέυουμε ως newtext, και στην συνέχεια χρησιμοποιούμε αυτό για τις επόμενες αλλαγές.

>newtext = rexp1.sub('',text)

Η απαλοιφή των script και style tags υλοποιήθηκε κατά παρόμοιο τρόπο καθώς αυτό που θέλουμε να σβήσουμε από το κείμενο είναι οτιδήποτε περιέχεται μεταξύ των <style> και </style> αλλά και <script> και </script>, αντίστοιχα. Για την αναζήτηση χρησιμοποιήθηκε μία κανονική έκφραση που περιέχει δύο μέλη για κάθε ένα <script> και <style>, αντίστοιχα, τα οποία διαχωρίζονται με or (|). Επίσης, χρησιμοποιείται η re.DOTALL(το οποίο χρησιμοποιώ προληπτικά σε όλες μου τις κανονικές εκφράσεις), καθώς δεν γνωρίζουμε πόσες σειρές καταλαμβάνει το κάθε tag.

>rexp2 = re.compile(r'(<script(.+?)</script>)|(<style(.+?)</style>)',re.DOTALL)

Για την εξαγωγή και εκτύπωση του συνδέσμου (ιδιότητα href) από τα <a> tags του κειμένου χρησιμοποιήθηκαν δύο εκφράσεις. Αρχικά, η πρώτη κανονική έκφραση υλοποιεί αναζήτηση για την εύρεση των <a> tags.

>rexp3 = re.compile(r'(<a)(.+?)/a>',re.DOTALL)

Η δεύτερη είναι αυτή που θα αναζητήση την ιδιότητα href.

>rexp4 = re.compile(r'href="(.+?)">([^<].+?)<',re.DOTALL)

Για την ορθή αναζήτηση τους υλοποιήθηκαν δύο δομές επανάληψης (for m2 in rexp3.finditer(newtext1) και for m3 in rexp4.finditer(m2.group(2))) , όπου η δεύτερη εμπεριέχεται στην πρώτη, με αποτέλεσμα να αναζητά την ιδιότητα που επιθυμούμε μεσα στα συγκεκριμένα tags. Όπως είναι ορατό και στις δύο περιπτώσεις (κανονικές εκφράσεις) χρησιμοποιήθηκαν παρενθέσεις καθώς η αναζήτηση του href θα υλοποιηθεί στο ενδιάμεσο των <a> tags, δηλαδή στο group(2) και αυτό που επιθυμούμε έπειτα να εκτυπώσουμε βρίσκεται σε συγκεκριμένα σημεία της κανονικής έκφρασης.

Η αναζήτηση όλων των tags του κειμένου έγινε με μία κανονική έκφραση της μορφής:

>rexp5 = re.compile(r'<(.+?)>',re.DOTALL)

καθώς παρατηρήθηκε ότι τα tags είναι της μορφής πχ <meta charset="utf-8" > *και αντικασαστήθηκαν με την χρήση της συνάρτησης sub(), ενώ το κείμενο που προέκυψε απόθηκεύτηκε σε μία νέα μεταβλητή newtext2, όπως γίνεται σε κάθε αλλαγή/αντικατάσταση/απαλοιφή που υλοποιήθηκε στο πρόγραμμα.*

>newtext2 = rexp5.sub('',newtext1)

Με παρόμοιο τρόπο με το παραπάνω ζητούμενο υλοποιήθηκε και η αντικατάσταση των HTLM entities (&, >, <,&nbdp;), με τον αντίστοιχο επιθυμητό χαρακτήρα (&, >,<,space). Ωστόσο, καθώς, η δομή στο κείμενο html των όσων έπρεπε να αντικατασθαθούν ήταν παρόμοια χρησιμοποιήθηκς μία κανονική έκφραση η οποία αναζητούσε τα κοινά στοιχεία, δηλαδή το & και το ;, χωρίς να ενδιαφέρεται για το περιεχόμενο.

>rexp6 = re.compile(r'&(.+?);',re.DOTALL)

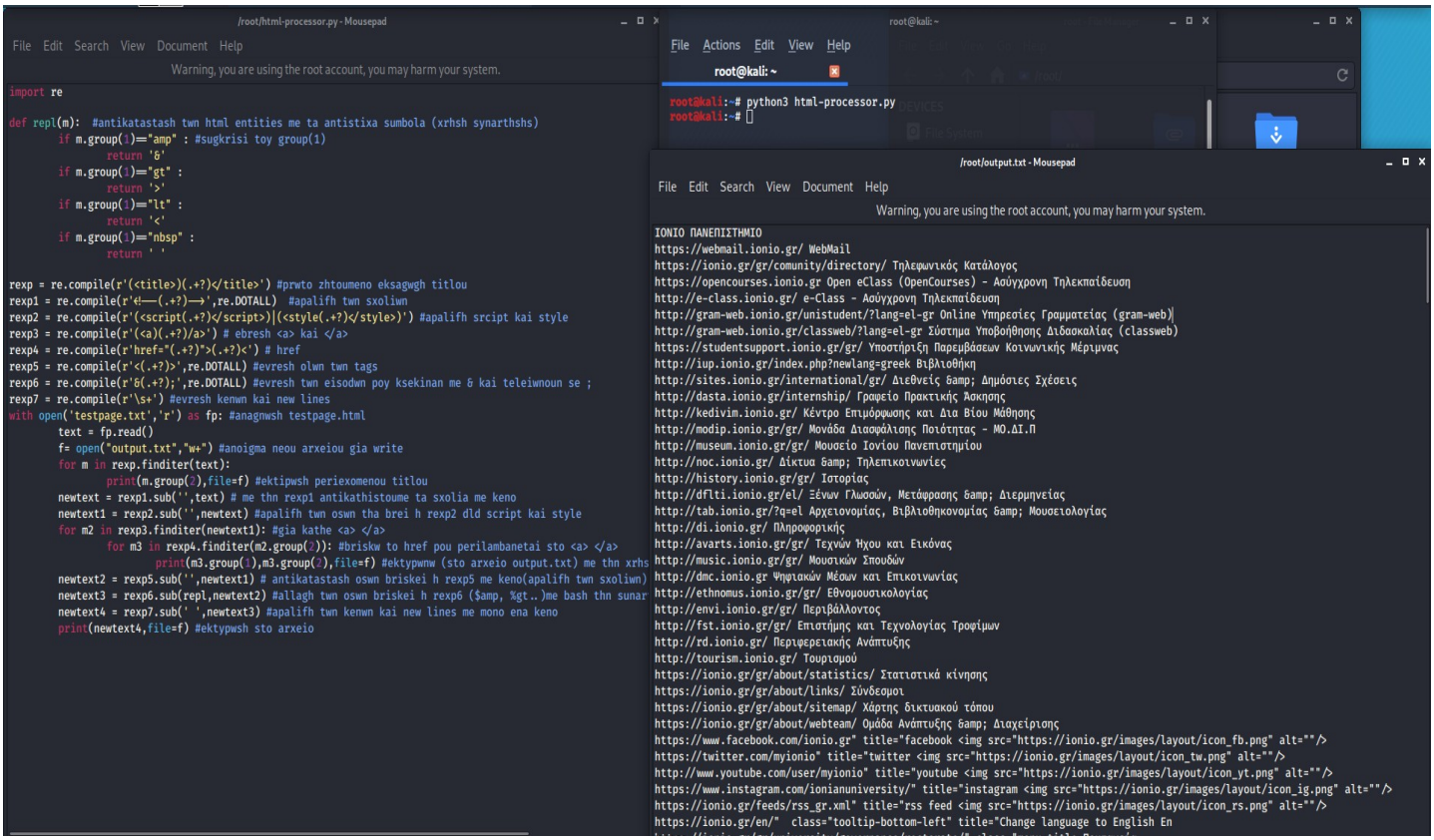
Έπειτα, για την σωστή αντικατάσταση στην sub() χρησιμοποιήθηκε μία συνάρτηση repl, η οποία ανάλογα με το περιεχόμενο (group(2) σύμφωνα με τις παρενθέσεις που τοποθετήθηκαν) επέστρεφε το αντίστοιχο σύμβολο.

>newtext3 = rexp6.sub(repl,newtext2)

Τέλος, γθα την απαλοιφή των κενών αλλά και των επιπλέων γραμμών χρησιμοποίηθηκε μία κανονική έκφραση της μορφής:

>rexp7 = re.compile(r'\s+')

και η αντικατάσταση έγινε με την χρήση της sub(). Το τελικό κείμενο που προέκυψε εκτυπώθηκε σε ένα αρχείο.



Πηγές

- Κανονικές εκφράσεις και Python
“http://mixstef.github.io/courses/compilers/lecturedoc/unit2/module1.html#sub”