Αναφορά Άσκησης 2 στο Μάθημα ΠΛΗ211

Πρόγραμμα computeSales.py

Ομάδα Εργασίας : LAB21142945
Αντώνιος - Εμμανουήλ Μαραγκουδάκης
Α.Μ: 2013030093

Μονομελής ομάδα, την άσκηση την έκανα μόνος μου.

Σκοπός της άσκησης ήταν η δημιουργία του προγράμματος computeSales.py σε Python το οποίο θα διαβάζει αρχεία με αποδείξεις πώλησης προϊόντων, θα μπορεί να κάνει έλεγχο ορθότητας και να μπορεί να εκτυπώνει μετά στατιστικά στοιχεία που του ζητούνται.

Αρχικά να αναφερθεί ότι το πρόγραμμα που υλοποίησα κάνει σωστά τους ελέγχους ορθότητας και έχει ελεγχθεί με αρχείο 150 χιλιάδων σειρών. Να αναφερθεί επίσης ότι θεωρεί άκυρες τις αποδείξεις τις οποίες περιέχουν όνομα προϊόντων με κενό διάστημα ('space'). Π.χ: Αν η απόδειξη περιέχει "ΣΑΛΑΤΑ ΧΩΡΙΑΤΙΚΗ:" , η απόδειξη θεωρείται λάθος ενώ αν περιέχει "ΣΑΛΑΤΑ_ΧΩΡΙΑΤΙΚΗ:" , η απόδειξη περνάει σωστά. Το έκανα έτσι αφού το παράδειγμα ορθής απόδειξης στην εκφώνηση της Άσκησης περιείχε μόνο μίας λέξης ονόματα προϊόντων με χρήση κάτω παύλας. π.χ ΟΝΟΜΑ_ΠΡΟΪΟΝΤΟΣ:

Ο κώδικας αναλύει κάθε απόδειξη σε γραμμές και κάθε γραμμή σε λέξεις. Έτσι έχω για κάθε λέξη και τις συντεταγμένες της: line_count και word_count. Έτσι ξέρω για κάθε λέξη τι πληροφορία θα πρέπει να περιέχει και κάνω τους κατάλληλους ελέγχους ορθότητας. Υπάρχει λεπτομερής σχολιασμός στον κώδικα που εξηγεί ακριβώς την λειτουργία του.

Συνοπτική επεξήγηση του κώδικα

Ο κώδικας ελέγχει και δέχεται μόνο το συγκεκριμένο φορμάτ απόδειξης:

```
----------------------------------      (line_count=1, word_count=1)== περιέχει μόνο "-"
ΑΦΜ:   1111111111                       (line_count=2,  word_count=1) == 'ΑΦΜ:'
                                      try: (line_count=2,  word_count=2) == 10ψήφιο int

ONOMA:  1   1.00   1.00                 (line_count>2,  word_count=1) == τελευταίο char =":"
    ….                                try: (line_count>2,  word_count=2) == int        (temp1)
    .....                             try: (line_count>2,  word_count=3) == float      (temp2)
    ….                                try: (line_count>2,  word_count=4) == float      (temp3) `
                                      πρέπει: (temp1*temp2==temp3) τότε total= total+ temp3

ΣΥΝΟΛΟ:    1.00                   if  (line_count>2,  word_count=1)=="ΣΥΝΟΛΟ:"  βάλε flag4=1
                                      if  (flag4=1 and word_count=2)   πρέπει:  float(word) == total
```

---------- (line_count>2, word_count=1)== περιέχει μόνο "-"
Αν ισχύουν όλες οι παραπάνω προϋποθέσεις και κάποια flags τα οποία δεν αναφέρονται στην αναφορά, τότε βάζω line_count=1, word_count=0 και όλα τα flags=0 ώστε να προχωρήσω στην επόμενη απόδειξη.
line _count=1 και όχι με 0 αφού ουσιαστικά η τελευταία σειρά αυτής της απόδειξης που έχω ήδη ελέγξει, ταυτίζεται με την πρώτη σειρά της επόμενης απόδειξης '"-------'" και άρα έχει ελεγχθεί

Έτσι αποδέχομαι το φορμάτ αυτής της απόδειξης και προχωράω στον έλεγχο της επόμενης. Ωστόσο όσο γινόταν αυτός ο έλεγχος, αποθηκεύτηκαν το αμφ (temp_AFM) και τα ονόματα και οι τιμές προϊόντων (products[]). Έτσι αν η απόδειξη είναι είναι έγκυρη αποθηκεύονται οι πληροφορίες αυτές σε δύο dictionaries, τα diction{} και diction2{}.

Όπως αναφέρθηκε όσο γίνεται ο έλεγχος του φορμάτ της απόδειξης αποθηκεύονται σε μια προσωρινή λίστα products[], τα ονόματα και οι τιμές προϊόντων με σειρά ένα προς ένα.
π.χ [ΣΑΛΑΤΑ 7.10 ΠΑΤΑΤΕΣ 4.50 ..... ΜΠΡΙΖΟΛΑ 8.50 ΚΟΥΒΕΡ 0.95]

Η λίστα αυτή ελέγχει και δεν εισάγει ξανά ένα όνομα προϊόντος το οποίο έχει υπάρξει ήδη στην απόδειξη αυτή, αλλά μόνο προσθέτει τις τιμές του στην θέση που ήδη βρίσκονται οι προηγούμενες συνολικές τιμές του στην λίστα products[]. Πριν ο κώδικας περάσει στον έλεγχο επόμενης απόδειξης διαγράφει τα προηγούμενα περιεχόμενα αυτής της λίστας για αυτό και την ανέφερα ως προσωρινή λίστα.
(Διαγράφονται αφού πρώτα περαστούν οι κατάλληλες πληροφορίες που περιέχουν σε δύο Dictionaries).

Έτσι αν ο έλεγχος του φορμάτ της απόδειξης είναι αποδεκτός τότε εισάγονται τα στοιχεία αυτά καθώς και το 10ψήφιο ΑΦΜ σε δύο διαφορετικά Dictionaries. Το diction2 θα χρησιμοποιηθεί για τον υπολογισμό των στατιστικών με κλειδί το όνομα προϊόντος όταν ο χρήστης πατήσει την επιλογή 2. Το diction θα χρησιμοποιηθεί για τον υπολογισμό των στατιστικών με κλειδί το 10ψήφιο ΑΦΜ όταν ο χρήστης πατήσει την επιλογή 3.

Πράδειγμα περιεχομένων των dictionaries:

Diction= { "1111111111": [ΣΑΛΑΤΑ 6.00 ΠΑΤΑΤΕΣ 3.50 ΚΟΥΒΕΡ 3.50],
"2222222222": [ΣΑΛΑΤΑ 6.50 ΠΑΤΑΤΕΣ 3.80 ΜΠΡΙΖΟΛΑ 7.50 ΚΟΥΒΕΡ 4.00] }

Diction2= { "ΣΑΛΑΤΑ": [1111111111 6.00 2222222222 6.50],
"ΠΑΤΑΤΕΣ": [1111111111 3.50 2222222222 3.80],
"ΚΟΥΒΕΡ": [1111111111 3.50 2222222222 4.00],
"ΜΠΡΙΖΟΛΑ": [2222222222 7.50] }

Ο Τρόπος με τον οποίο κατασκευάζονται το κάθε dictionary από την προσωρινή λίστα products[] της κάθε απόδειξης είναι περίπλοκος και εξηγείται με λεπτομερή σχολιασμό στο αρχείο του κώδικα.

Έπειτα που το πρόγραμμα διαβάσει όλο το αρχείο, περιμένει πάλι την επιλογή του χρήστη ή για να διαβαστεί πάλι κάποιο αρχείο ή για να εκτύπωση κάποια στατιστική ή για τερματισμό. Τα δεδομένα από διαφορετικά αρχεία συνυπολογίζονται και γενικότερα όλες οι ΣΗΜΑΝΤΙΚΕΣ ΣΗΜΕΙΩΣΕΙΣ: της εκφώνησης λήφθηκαν υπόψιν.

Όταν ζητηθεί η εκτύπωση κάποιας στατιστικής τα dictionaries περιέχουν έτοιμη άλλα αταξινόμητη την πληροφορία που χρειάζεται. Έτσι απλά με βάση την που δίνει ο χρήστης ταξινομείται η λίστα που είναι συνδεδεμένη με το key αυτό στο κατάλληλο dictionary και εκτυπώνεται. Σχόλια υπάρχουν παντού που εξηγούν καλύτερα.

Επιλογή δομής δεδομένων

Επέλεξα την δομή δεδομένων των dictionaries γιατί είναι πολύ γρήγορες και βολικές στην σύνδεση πληροφορίας μιας λέξης κλειδί με μια πληροφορία που την περιγράφει. Αφού η κάθε λέξη κλειδί πρέπει "ξεκλειδώνει" πολύ πληροφορία τελικά επέλεξα να συνδέεται με μία λίστα με πληροφορίες όπως φαίνεται παραπάνω στο παράδειγμα. Έτσι εξασφάλισα ότι μπορώ να διαβάσω πολύ μεγάλα αρχεία σε συνδυασμό κιόλας με το ότι δεν γράφω κάπου όλο το αρχείο αλλά γράφω μόνο τις πληροφορίες που θα μου χρειαστούν στις εκτυπώσεις των ζητούμενων στατιστικών. Επιπλέον για κάθε απόδειξη που κρατάω δεδομένα χρησιμοποιώ την ίδια προσωρινή λίστα products[] έτσι δεν δεσμεύεται πολύ μνήμη. Έτσι τελικά δεν έχω ούτε προβλήματα άσκοπης διαχείρισης μνήμης και επιτυγχάνω και τη γρήγορη εκτέλεση του προγράμματος, λόγο των dictionaries.

Διάφορες Λεπτομέρειες

Όλες οι πληροφορίες λέξεων αποθηκεύονται σε κεφαλαία και γίνεται και encoding σε utf-8 για το διάβασμα ελληνικών χαρακτήρων. Επιπλέον χρησιμοποιήθηκαν κάποια try: except: προκειμένου να διασφαλισθεί το αν είναι int ή float συγκεκριμένα περιεχόμενα των αποδείξεων. Έγινε χρήση της εντολής round (variable_name , 2) για την στρογγυλοποίηση σε δύο δεκαδικά ψηφία των float τιμών. Το αρχείο 'file' αναλύθηκε σε σειρές με χρήση της εντολής file.readline() , ενώ οι σειρές 'line' αναλύθηκαν σε λέξεις με χρήση της εντολής line.split() .

Συμπεράσματα

Με το πέρας της άσκησης προέκυψε η εξοικείωση με την γλώσσα προγραμματισμού Python καθώς και η κατανόηση της δομής δεδομένων των Dictionaries. Έγινε κατανοητό το πώς να διαβάζουμε και να διαχειριζόμαστε πολύ πληροφορία από ένα αρχείο χωρίς να δεσμεύεται άσκοπη μνήμη και χωρίς να γίνετε αργός ο κώδικάς μας.

Παράδειγμα εκτέλεσης προγράμματος (156.000 σειρών , 3sec)