ΕΡΓΑΛΕΙΑ ΑΝΑΠΤΥΞΗΣ ΛΟΓΙΣΜΙΚΟΥ

Αναφορά 2ης άσκησης

Βιριράκης Γεώργιος (2016030035)
Γαλάνης Μιχάλης (2016030036)

ΕΙΣΑΓΩΓΗ

Στη δεύτερη άσκηση ασχοληθήκαμε με τη γλώσσα Python κατασκευάζοντας ένα πρόγραμμα σκοπό την εξοικείωσή μας με αυτή. Αυτό περιείχε εισαγωγικές έννοιες, δομές δεδομένων, ελέγχους ροής προγράμματος, συναρτήσεις και επεξεργασία αρχείων.

Η εργασία έχει υλοποιηθεί πλήρως από κοινού.

ΠΡΟΓΡΑΜΜΑ COMPUTESALES.PY

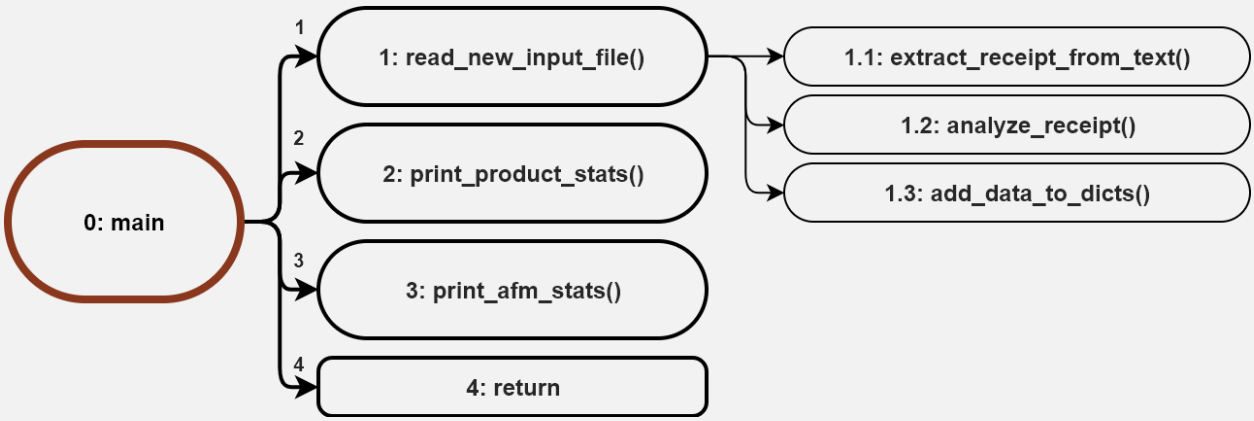
Επισκόπηση

Το πρόγραμμα αποσκοπούσε στην ανάγνωση αρχείων αποδείξεων, την ανάλυσή τους και εκτύπωση στατιστικών τους στοιχείων. Διαθέτει ένα μενού χρήστη με επιλογή ανάγνωσης αρχείου, εκτύπωσης στατιστικών για συγκεκριμένο προϊόν ή ΑΦΜ και έξοδος του προγράμματος. Το πρόγραμμα έχει επίσης τη δυνατότητα να αγνοήσει τις λανθασμένες αποδείξεις ενώ λαμβάνει υπόψιν τις ορθές.

ℹ️ **Σημείωση**: στην αναφορά επισυνάπτονται τμήματα του κώδικα που θεωρούμε σημαντικά.

Υλοποίηση

Το πρόγραμμα ξεκινάει εκτυπώνοντας επαναληπτικά το μενού επιλογών που αναλύθηκε παραπάνω. Παρακάτω παρουσιάζεται σε πολύ επιφανειακό επίπεδο η δομή των συναρτήσεων του προγράμματος και ακολουθεί η ανάλυσή τους.



1: read_new_input_file() - Ανάγνωση Αρχείων

Αρχικά ζητάμε από το χρήστη ένα όνομα αρχείου για ανάγνωση. Επιβεβαιώνουμε ότι το αρχείο αυτό υπάρχει και είναι αρχείο με την **os.path.isFIle()** και το ανοίγουμε με υποστήριξη για ελληνικούς χαρακτήρες (UTF-8). Για να διαβάσουμε το αρχείο χρησιμοποιούμε μία εμφολευμένη επανάληψη while, όπου η εσωτερική διαβάζει γραμμή – γραμμή μία απόδειξη ενώ η εξωτερική επαναλαμβάνει τη διαδικασία για όλες τις αποδείξεις.

```
36  #F1 : INPUT FILE READ
37  #     reads a file and saves receipts information so that
38  #     various stats can be generated
39  def read_new_input_file():
40
41      #Asks user for file name and opens file if it exists
42      file_name = input('Please enter file name: ')
43      if (not os.path.isfile(file_name)):
44          print('No such file found.')
45          return
46      f = open(file_name, 'r', encoding='utf-8')
47      #-------------------------------------------------
48
49      #Scans file, extracts receipt, analyzes its information and adds it to the dictionaries
50      line = ' '
51      while (line): #Repeats for every receipt
52          #Step 1 - Extracts Receipt Text without dashes (EX: [ΑΦΜ: 2196911440][TZATZIKI: 1   1.29    1.29][ΣΥΝΟΛΟ:   1.29])
53          enabled, line, receipt_string = extract_receipt_from_text(line, f)
54          if (receipt_string == ""):
55              continue
56          #Step 2 - Analyzes Receipt Information (EX: {[ΑΦΜ][2196911440]}{[TZATZIKI][1][1.29][1.29]}{[ΣΥΝΟΛΟ][1.29]})
57          enabled, temp_dict, temp_afm = analyze_receipt(enabled, receipt_string)
58          #Step 3 - Adds Data to Dictionaries if receipt is enabled
59          add_data_to_dicts(enabled, temp_dict, temp_afm)
60      #-------------------------------------------------------------------------------------
61      f.close()
```

Σημειώνουμε επίσης ότι κάθε απόδειξη που διαβάζεται, αναλύεται και αποθηκεύεται επιτόπου. Αυτές οι τρείς λειτουργίες βρίσκονται σε συναρτήσεις και θα εξηγηθούν σύντομα παρακάτω.

Η κύρια δομή δεδομένων που χρησιμοποιούμε είναι dictionaries για δύο λόγους. Είναι γρήγορα (Get(1), Set(1), Delete O(1)), ιδανικά για μεγάλο όγκο δεδομένων και αξιοποιούμε τη συσχέτιση μεταξύ key, value. Τα dictionaries στη περίπτωσή μας είναι τα εξής:

prod_dict = { όνομα προϊόντος, { ΑΦΜ, συνολικό κόστος } } -> χρήσιμο για ερώτημα 2
afm_dict = { ΑΦΜ, {όνομα προϊόντος, συνολικό κόστος } } -> χρήσιμο για ερώτημα 3

1.1: extract_receipt_from_text()

Αυτή η συνάρτηση είναι υπεύθυνη για την εξαγωγή πληροφοριών μιας απόδειξης σε μια μεταβλητή receipt_string. Χρησιμοποιούμε μια μεταβλητή enabled για να γνωρίζουμε αν θα λάβουμε υπόψιν την συγκεκριμένη απόδειξη η οποία ξεκινάει ως έγκυρη. Μέσα στη while ελέγχουμε αν η πρώτη γραμμή μιας απόδειξης περιέχει

παύλες και αν η δεύτερη γραμμή περιέχει το ΑΦΜ και τερματίζει μόλις εντοπίσει το σύνολο.

```
#f_extract_receipt_from_text : reads each line and tries to construct a string
#                              (receipt_string) with valid receipt information
def extract_receipt_from_text(line, f):
    line = f.readline() #Reads Line
    enabled = True #Each receipt starts as a valid one. If a mistake is detected, it becomes invalid (false)
    receipt_string = "" #This is the string where the receipt information will be stored. We only store the
                        #internal part of a receipt (without the dashes)
    line_counter = 0  #Needed to check which line of a specific receipt we are in
    while (line and line != "\n"): #Repeats every line in receipt
        #Dashes have to be in first line, ΑΦΜ has to be in second line
        if ((line_counter == 0 and "-" not in line) or (line_counter == 1 and "ΑΦΜ" not in line)):
            break
        #Inserts line into receipt_string
        if ("-" not in line):
            receipt_string += line
        #After this line we exit the loop
        if (line[0:6] == "ΣΥΝΟΛΟ"):
            break
        else: #Proceeds to next line
            line_counter += 1
            line = f.readline()
    return enabled, line, receipt_string
```

1.2: analyze_receipt()

Αυτή η συνάρτηση δέχεται ως όρισμα το receipt_string που δημιουργήθηκε στο προηγούμενο στάδιο και αναλύει την πληροφορία της απόδειξης. Αυτό το string (που περιέχει αρχικά την απόδειξη) γίνεται split σε πίνακες που περιέχουν τις γραμμές της και αυτές γίνονται split σε επιμέρους πίνακες που περιέχει ο καθένας κάποιο χαρακτηριστικό της απόδειξης.

Επιπλέον έλεγχοι που επιτυγχάνονται στο στάδιο της ανάλυσης είναι:

- Το ΑΦΜ να αποτελείται από 10 αριθμούς
- Ο πολλαπλασιασμός του τεμαχίου με τη τιμή/τεμάχιο να είναι ορθός
- Το άθροισμα της τιμής των προϊόντων να είναι ορθός

Καθένα από τα προϊόντα αποθηκεύονται σε ένα προσωρινό **temp_dict** (dictionary) με μορφή {όνομα προϊόντος, κόστος προϊόντος} το οποίο και επιστρέφεται για να προχωρήσει στο επόμενο στάδιο, στη προσθήκη δεδομένων στα κύρια dictionaries.

```
#f_analyze_receipt: takes as input a receipt_string and analyzes its information
#
def analyze_receipt(enabled, receipt_string):
    temp_lines = receipt_string[:-1].splitlines() #Splits based on new line character
    #Evaluating first (afm) line
    temp_afm = temp_lines[0].split(":") #Splits based on ":" [ΑΦΜ: 7370682019] -> [ΑΦΜ][ 7370682019]
    temp_afm[1] = str(temp_afm[1]).strip() #Removes whitespace on afm number
    if (len(temp_afm[1]) != 10 or parse(temp_afm[1]) == -1): #Checking for: afm being 10 digits long
        enabled = False
    #Evaluating intermediate (product) lines
    temp_sum = 0.0
    temp_dict = {} #temporary dictionary to store {product_name, accumulative_cost}
```

```
    for i in range(1, len(temp_lines)-1):
        temp_products = re.split(':|\t| ', temp_lines[i]) #Splits every ":, tab or space"
        #print(temp_products)
        real_temp_products = [] #real_temp_products[0]= name,  real_temp_products[1]= quantity
        for j in range(len(temp_products)):
            #print(temp_products[j])
            if (temp_products[j] != ""):
                real_temp_products.append(temp_products[j])
        # print(real_temp_products)
        #Evaluating product multiplication
        if (round(float(real_temp_products[1]) * float(real_temp_products[2]), 2) != round(float(real_temp_products[3]), 2)):
            enabled = False
        temp_sum += round(float(real_temp_products[3]), 2)
        #Insert data to temp_dict. If product already exists, adds up to the current cost
        if (real_temp_products[0].upper() in temp_dict):
            temp_dict[real_temp_products[0].upper()] += round(float(real_temp_products[3]), 2)
        else:
                temp_dict[real_temp_products[0].upper()] = round(float(real_temp_products[3]), 2)
    #Evaluating last (total) line
    temp_total = temp_lines[len(temp_lines)-1].split(":") # temp_total[1] is the actual total amount
    if ("ΣΥΝΟΛΟ" not in temp_total[0] or round(float(temp_total[1]), 2) != round(temp_sum, 2)): #Checking for: correct sum and sum text
        enabled = False
    #Returns
    return enabled, temp_dict, temp_afm[1]
```

1.3: add_data_to_dicts()

Αν μέχρι τώρα η απόδειξη παραμείνει έγκυρη, επιθυμούμε να εισάγουμε επαναληπτικά τη πληροφορία του temp_dict σε καθένα από τα afm_dict, prod_dict. Γνωρίζοντας ότι τα dictionaries δε δέχονται duplicate κλειδιά, έχουμε τις ακόλουθες περιπτώσεις (θα αναλυθούν για το afm_dict): Αν δεν υπάρχει το ΑΦΜ που θέλουμε να προσθέσουμε, το εισάγουμε, παράλληλα με το προϊόν (όνομα και τιμή). Διαφορετικά ελέγχουμε αν για το ΑΦΜ που υπάρχει, υπάρχει και το όνομα του προϊόντος που θέλουμε να προσθέσουμε. Αν δεν υπάρχει το εισάγουμε μαζί με το κόστος του, αλλιώς απλά ενημερώνουμε τη συνολική τιμή.

Αντίστοιχη διαδικασία ακολουθούμε και για το prod_dict.

```
#f_add_data_to_dicts: If receipt is enabled (is valid), this function fills up
#                     the receipt data into the 2 dictionaries
def add_data_to_dicts(enabled, temp_dict, temp_afm):
    if (enabled == True): #If receipt is valid
        for k in temp_dict.keys(): #For each product_name in temp_dict

            #Filling up afm_dict data (#No3 - {afm, {product_name, total_cost}})
            if (temp_afm in afm_dict): #if afm already exists
                if (k in afm_dict[temp_afm]): #if product_name already exists then adds up product cost
                    afm_dict[temp_afm][k] += temp_dict[k]
                    afm_dict[temp_afm][k] = round(afm_dict[temp_afm][k], 2)
                else: #if product_name doesn't exist, just add another product (with its cost)
                    afm_dict[temp_afm][k] = temp_dict[k]
            else: #if afm doesn't exist, add another afm with a product_name and cost
                afm_dict[temp_afm] = {}
                afm_dict[temp_afm][k] = temp_dict[k]
            #---------------------------------------------------------------------
```

```
#f_add_data_to_dicts: If receipt is enabled (is valid), this function fills up
#                     the receipt data into the 2 dictionaries
def add_data_to_dicts(enabled, temp_dict, temp_afm):
    if (enabled == True): #If receipt is valid
        for k in temp_dict.keys(): #For each product_name in temp_dict

            #Filling up afm_dict data (#No3 - {afm, {product_name, total_cost}})
            if (temp_afm in afm_dict): #if afm already exists
                if (k in afm_dict[temp_afm]): #if product_name already exists then adds up product cost
                    afm_dict[temp_afm][k] += temp_dict[k]
                    afm_dict[temp_afm][k] = round(afm_dict[temp_afm][k], 2)
                else: #if product_name doesn't exist, just add another product (with its cost)
                    afm_dict[temp_afm][k] = temp_dict[k]
            else: #if afm doesn't exist, add another afm with a product_name and cost
                afm_dict[temp_afm] = {}
                afm_dict[temp_afm][k] = temp_dict[k]
            #---------------------------------------------------------------------

            #Filling up prod_dict data (#No2 - {product_name, {afm, total_cost}})
            if (k in prod_dict): #if product_name already exists
                if (temp_afm in prod_dict[k]): #if afm already exists then adds up product cost
                    prod_dict[k][temp_afm] += temp_dict[k]
                    prod_dict[k][temp_afm] = round(prod_dict[k][temp_afm], 2)
                else: #if afm doesn't exist, just add another afm (with product cost)
                    prod_dict[k][temp_afm] = temp_dict[k]
            else: #if product_name doesn't exist, add another product_name with an afm and cost
                prod_dict[k] = {}
                prod_dict[k][temp_afm] = temp_dict[k]
            #---------------------------------------------------------------------
```

2,3: print_product_stats(), print_afm_stats() - Εκτύπωση στατιστικών

Ζητάνε και οι δύο από το χρήστη να εισάγουν κάποιο όνομα προϊόντος / ΑΦΜ αντίστοιχα ελέγχοντας αν υπάρχουν. Δημιουργούμε κάποιο temp dictionary στο οποίο θα αποθηκεύσουμε ταξινομημένα τα στοιχεία που μας ενδιαφέρουν. Αυτό επιτυγχάνεται με τη συνάρτηση sorted της python. Τέλος εκτυπώνουμε το ζητούμενο.

```
#F2 : PRINTS STATISTICS FOR A SPECIFIC PRODUCT:
#     displays total cost of product that has been ordered for each AFM
def print_product_stats():
    #Asks product name, converts it to UPPER CASE and checks if exists
    product_name = input("Please enter product name: ").upper()
    if (product_name not in prod_dict):
        return
    #Creates new sorted dict (by key: afm) and prints each afm with its corresponding cost
    sorted_prod_dict = sorted((prod_dict[product_name]).items())
    for k, v in sorted_prod_dict:
        print(k, "%.2f" % v)
```

```python
#F3 : PRINTS STATISTICS FOR A SPECIFIC AFM:
#     displays total cost of each product ordered by this specific AFM
def print_afm_stats():
    #Asks afm number and checks if exists
    afm_number = input("Please enter afm number: ")
    if (afm_number not in afm_dict):

        return
    #Creates new sorted dict (by key: product name) and prints each product name with its corresponding cost
    sorted_afm_dict = sorted((afm_dict[afm_number]).items())
    for k, v in sorted_afm_dict:
        print(k, "%.2f" % v)
```

Αποτελέσματα & Συμπεράσματα

Παρακάτω επισυνάπτουμε μια συνηθισμένη εκτέλεση:

```
PS D:\Documents\Projects\Python\LAB2> & C:/Users/mgala/AppData/Local/Programs/Python/Python36-32/python.exe d:/Documents/Projects/Python/LAB2/computeSales.py

----------MENU----------
1. Read new input file
2. Print statistics for a specific product
3. Print statistics for a specific AFM
4. Exit
Give your preference: 1
Please enter file name: hardInput1
```

```
----------MENU----------
1. Read new input file
2. Print statistics for a specific product
3. Print statistics for a specific AFM
4. Exit
Give your preference: 2
Please enter product name: ποικιλια
1619015286 142313.70
1790057711 141606.42
1912827619 138825.69
1992576512 141381.05
2048410626 143513.73
2268632629 137744.54
2685212752 141481.72
2912127502 141512.17
3146180103 141739.37
```

```
----------MENU----------
1. Read new input file
2. Print statistics for a specific product
3. Print statistics for a specific AFM
4. Exit
Give your preference: 3
Please enter afm number: 1619015286
AMSTEL 41413.51
ΜΠΡΙΖΟΛΑ_ΜΟΣΧΑΡΙΣΙΑ 40413.70
ΜΠΡΙΖΟΛΑ_ΧΟΙΡΙΝΗ 39069.88
ΠΟΙΚΙΛΙΑ 142313.70
ΤΖΑΤΖΙΚΙ 800545.76
ΦΙΛΕΤΟ_ΚΟΤΟΠΟΥΛΟ 39427.54
```

Η χρήση των dictionaries βοήθησε από πλευράς χρόνου το πρόγραμμα να εκτελείται real-time (για ανάγνωση αρχείου 3 εκατομμυρίων γραμμών χρειάζονται 13-18 δευτερόλεπτα, η εκτύπωση στατιστικών είναι άμεση) ενώ η ανάλυση αποδείξεων κατά τη διάρκεια της ανάγνωσης του αρχείου βοήθησε στην εξοικονόμηση μνήμης. Τα παραπάνω εξασφάλισαν ότι μπορούμε να διαβάσουμε μεγάλα αρχεία.