Ε.Α.Π./ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΣ

4η ΓΡΑΠΤΗ ΕΡΓΑΣΙΑ

ΑΚΑΔΗΜΑΪΚΟΥ ΕΤΟΥΣ 2021-2022

xx/xx/2022

ΕΝΔΕΙΚΤΙΚΑ ΣΧΕΔΙΑ ΛΥΣΕΩΝ - ΥΠΟΔΕΙΞΕΙΣ

ΥΠΟΕΡΓΑΣΙΑ 1. (βαθμοί 25)

```
import pickle # Χρήση της βιβλιοθήκης pickle για τη σειριοποίηση του λεξικού

class Store():
    '''κλάση για ανάκτηση και αποθήκευση δεδομένων σε ένα αρχείο pickle'''
    def __init__(self, f):
        self.store = f

    def load(self):
        try:
            with open(self.store, "rb") as f:
                return pickle.load(f)
        except: # error in opening file
            return {}

    def save(self, data):
        try:
            with open(self.store, "wb") as f:
                pickle.dump(data, f)
            print('επιτυχής αποθήκευση δεδομένων')
        except Exception as e:
            print ('σφάλμα στην αποθήκευση των δεδομένων', e)

class AddressBook():
    '''κλάση για διαχείριση επαφών'''
    def __init__(self, f):
        self.store = Store(f)
        self.epafes = self.store.load()
        self.contactsManagement()
        self.store.save(self.epafes)

    def contactsManagement(self):
        ''''Απλή διαχείριση επαφών με εισαγωγή νέων επαφών και εκτύπωση
επαφών.'''
        while True:
            reply = input('πατήστε + για νέα επαφή, ? για λίστα επαφών ή
<enter> για έξοδο\n>>> ')
            if not reply: break
            if reply == '+':
                name = input('όνομα: ')
                phone = input('τηλέφωνο: ')
                if name and phone: self.epafes[name] = phone
                else:print('παρακαλώ δώστε στοιχεία')
            elif reply == '?':
                self.showContacts()

    def showContacts(self):
        for e in sorted(self.epafes):
            print(f'{e}: {self.epafes[e]}')

if __name__ == '__main__':
    addressbookFilename = "addressbook"
    AddressBook(addressbookFilename)
```

ΥΠΟΕΡΓΑΣΙΑ 2. (βαθμοί 25)

```
import pandas as pd # Χρήση της βιβλιοθήκης pandas
import matplotlib.pyplot as plt # Χρήση της βιβλιοθήκης matplotlib.pyplot

def printMeteoData(file,city):
    '''Ανάγνωση μετεωρολογικών δεδομένων, για μια πόλη, και εκτύπωσή τους'''
    # Υποερώτημα α
    # Ανάγνωση με την pandas του φύλλου για την πόλη σε ένα dataframe
    city_df = pd.read_excel(file,sheet_name=city)
    # Εκτύπωση του dataframe με τα στοιχεία της πόλης
    print(city)
    print('-'*len(city))
    print(city_df)

def printMeteoStats(file,metric,city):
    '''Ανάγνωση μετεωρολογικών δεδομένων, για μια πόλη, και εκτύπωση βασικών
στατιστικών στοιχείων μιας μετρικής'''
    # Υποερώτημα β
    # Ανάγνωση με την pandas του φύλλου για την πόλη σε ένα dataframe
    city_df = pd.read_excel(file,sheet_name=city)
    # Υπολογισμός και εκτύπωση μέσης τιμής και οριακών τιμών της μετρικής
    mean_value = city_df[metric].mean()
    min_value = city_df[metric].min()
    max_value = city_df[metric].max()
    print(city)
    print('-'*len(city))
    print(f'Μέση {metric}: {mean_value:.2f}')
    print(f'Ελάχιστη {metric}: {min_value}')
    print(f'Μέγιστη {metric}: {max_value}')

def plotMeteoData(file,x_axis,metric,*list_of_cities):
    '''Ανάγνωση μετεωρολογικών δεδομένων και εμφάνιση συγκριτικού γραφήματος
μιας μετρικής'''
    # Υποερώτημα γ
    # Επανάληψη για όλες τις πόλεις
    for city in list_of_cities:
        # Ανάγνωση με την pandas του φύλλου για την πόλη σε ένα dataframe
        city_df = pd.read_excel(file,sheet_name=city)
        # Δημιουργία γραφήματος με τα δεδομένα της μετρικής για την πόλη
        plt.plot(city_df[x_axis],city_df[metric],label=city)
    # Προσθήκη τίτλου, ετικετών αξόνων και υπομνήματος
    plt.title(f'{metric} για {len(list_of_cities)} πόλεις')
    plt.xticks(fontsize=8,rotation=45,ha="right")
    plt.xlabel(x_axis)
    plt.ylabel(metric)
    plt.legend(fontsize=8)
    # Εμφάνιση του γραφήματος
    plt.show()

# Κυρίως πρόγραμμα

# Ορισμός του αρχείου με τα δεδομένα προς ανάγνωση
excel_file='weatherdata.xlsx'

# Υποερώτημα δ

# Εκτύπωση, με την printMeteoData, των στοιχείων για την Αθήνα
printMeteoData(excel_file,'Αθήνα')
# Εκτύπωση, με την  printMeteoStats, βασικών στατιστικών τιμών θερμοκρασίας
# για τη Θεσσαλονίκη
```

```
printMeteoStats(excel_file,'Θερμοκρασία','Θεσσαλονίκη')
# Δημιουργία συγκριτικού διαγράμματος για την υγρασία στις τρεις πόλεις
plotMeteoData(excel_file,'Μήνας','Υγρασία','Αθήνα','Θεσσαλονίκη','Πάτρα')
```

ΥΠΟΕΡΓΑΣΙΑ 3. (βαθμοί 25)

```
import sqlite3  #Χρήση της βιβλιοθήκης sqlite3

def show_records(table): #Συνάρτηση προβολής των εγγραφών του πίνακα table
    sql="SELECT * from {}" #Ερώτημα (query) για την προβολή των εγγραφών
ενός πίνακα
    # Υποερώτημα α
    try:
        with sqlite3.connect('library.db') as conn: #Σύνδεση με τη βάση
δεδομένων library
            cursor = conn.cursor()
            cursor.execute(sql.format(table)) #Εκτέλεση του ερωτήματος
(query) προβολής των εγγραφών του πίνακα table
            names = [description[0] for description in cursor.description]
            print(names)  #Προβολή των ονομασιών των πεδίων του πίνακα table
            for row in cursor:
                print(row) #Προβολή των εγγραφών του πίνακα table
            print('\n')
    except sqlite3.Error as err:
        print ("¨Λάθος:", err) #Προβολή μηνύματος λάθους
        return False

def insert_student(name,surname): #Συνάρτηση εισαγωγής μαθητή
    sql="INSERT INTO students(name,surname) VALUES (?,?)" #Ερώτημα (query)
για την εισαγωγή του ονόματος (name) και επώνυμου (surname) του νέου μαθητή
στον πίνακα students
    # Υποερώτημα β
    try:
        with sqlite3.connect('library.db') as conn:  #Σύνδεση με τη βάση
δεδομένων library
            cursor=conn.cursor()
            cursor.execute(sql,(name,surname)) #Εκτέλεση του ερωτήματος
(query) για την εισαγωγή του ονόματος και του επώνυμου του νέου μαθητή στον
πίνακα students
            conn.commit() #Αποθήκευση των αλλαγών στη βάση δεδομένων library
    except sqlite3.Error as err:
        print ("Λάθος:", err) #Προβολή μηνύματος λάθους
        return False

def delete_student(code): #Συνάρτηση διαγραφής μαθητή
    sql="DELETE FROM students WHERE id==(?)" #Ερώτημα (query) για τη
διαγραφή του μαθητή βάσει του κωδικού του(id) από τον πίνακα students
    # Υποερώτημα γ
    try:
        with sqlite3.connect('library.db') as conn:  #Σύνδεση με τη βάση
δεδομένων library
            cursor=conn.cursor()
            cursor.execute(sql,(code,)) #Εκτέλεση του ερωτήματος (query) για
τη διαγραφή του μαθητή από τον πίνακα students
            conn.commit() #Αποθήκευση των αλλαγών στη βάση δεδομένων library
    except sqlite3.Error as err:
        print ("Λάθος:", err) #Προβολή μηνύματος λάθους
        return False
```

```
# Κυρίως πρόγραμμα

#Συμβολοσειρά με τις επιλογές του μενού
library_menu='''Επιλογές συστήματος:
1) Προβολή μαθητών
2) Προβολή βιβλίων
3) Προβολή δανεισμών
4) Καταχώρηση μαθητή
5) Διαγραφή μαθητή
6) Έξοδος
Η επιλογή σας: '''

# Κεντρικό μενού της εφαρμογής
while True:
    entry=input(library_menu) #Εισαγωγή επιλογής μενού
    if entry =='1':
        show_records('students') #Χρήση της συνάρτησης show_records για τον
πίνακα students
    elif entry == '2':
        show_records('books')  #Χρήση της συνάρτησης show_records για τον
πίνακα books
    elif entry == '3':
        show_records('lending') #Χρήση της συνάρτησης show_records για τον
πίνακα lending
    elif entry == '4':
        on=input("Καταχώρησε το όνομα του μαθητή: \n")
        ep=input("Καταχώρησε το επώνυμο του μαθητή: \n")
        insert_student(on,ep) #Χρήση της συνάρτησης insert_student
    elif entry == '5':
        student_code=input("Καταχώρησε τον κωδικό του μαθητή προς διαγραφή:
\n")
        delete_student(student_code) #Χρήση της συνάρτησης delete_student
    elif entry=='6':
        break
    else:
      print ("Λανθασμένη επιλογή. Παρακαλώ επιλέξετε 1 έως 6 \n" ) #Μήνυμα
λάθους σε περίπτωση λανθασμένης επιλογής menu
```

ΥΠΟΕΡΓΑΣΙΑ 4. (βαθμοί 25)

```
import numpy as np # Χρήση της βιβλιοθήκης numpy
import time as tm # Χρήση της βιβλιοθήκης time

def internalProduct(v1,v2):
    '''Τυπική υλοποίηση εσωτερικού γινομένου διανυσμάτων.'''
    # Υποερώτημα α
    #
    # Υπολογισμός του γινομένου των v1 και v2 με την τυπική
    # υλοποίηση
    d = 0
    for i in range(len(v1)):
        d += v1[i]* v2[i]
    return d

def internalProduct_np(v1,v2):
    # Υποερώτημα β
    #
```

```
    # Υπολογισμός του γινομένου των v1 και v2 με κλήση της dot() της numpy
    return np.dot(v1, v2)

def timeit(mode, rep,v1,v2):
    '''Χρονομέτρηση επαναληπτικού υπολογισμού εσωτερικού γινομένου με την
τυπική υλοποίηση'''
    # Αποθήκευση στην μεταβλητή start του χρόνο έναρξης του επαναληπτικού
    # υπολογισμού με την τυπική υλοποίηση του πολλαπλασιασμού διανυσμάτων
    start_time = tm.perf_counter()
    # Επανάληψη πολλές φορές, ώστε η ακρίβεια του ρολογιού να μην επηρεάζει
    # το αποτέλεσμα
    # Υποερώτημα γ
    for _ in range(rep):
        # Κλήση της συνάρτησης, για τον υπολογισμό του γινομένου των v1 και
v2
        if mode == "τυπική υλοποίηση":
            prod = internalProduct(v1,v2)
        else:
            prod = internalProduct_np(v1,v2)
    # Εμφάνιση του αποτελέσματος
    print (f"Εσωτερικό γινόμενο διανυσμάτων ({mode}): {prod}")
    # Αποθήκευση στην μεταβλητή finish_time του χρόνου ολοκλήρωσης του
    # υπολογισμού
    finish_time = tm.perf_counter()
    # Επιστροφή χρονικού διαστήματος
    return finish_time-start_time

# Κυρίως πρόγραμμα

# Αρχικοποίηση παραμέτρων
random_num_range = 100 # το εύρος των παραγόμενων τυχαίων αριθμών
vector_size = 10000 # το πλήθος των στοιχείων ενός διανύσματος
repetitions = 5000 # ο αριθμός επαναλήψεων του υπολογισμού
# Αρχικοποίηση δύο διανυσμάτων, v1 και v2, με τυχαίους αριθμούς
v1 = np.random.randint(random_num_range,size=vector_size)
v2 = np.random.randint(random_num_range,size=vector_size)

# Υποερώτημα δ

# Υπολογισμός και εμφάνιση χρόνων αναμονής
t = []
for mode in ["τυπική υλοποίηση", "numpy"]:
    print("\n", f"Υπολογισμός με {mode}. Παρακαλώ περιμένετε ...")
    # Χρονομέτρηση και εμφάνιση του χρόνου εκτέλεσης του επαναληπτικού
    # υπολογισμού με την τυπική υλοποίηση και με την numpy
    t.append(timeit(mode, repetitions,v1,v2))
    print(f"χρόνος ({mode}) = {t[-1]:0.5f}")

# Υπολογισμός και εμφάνιση του λόγου των χρόνων εκτέλεσης
ratio = t[0]/t[1]
print(f"Η numpy είναι {ratio:0.1f} φορές γρηγορότερη")
```