

**Universitatea Tehnică a Moldovei**

**Facultatea Calculatoare Informatică și Microelectronică**

**Departamentul Informatică și Ingineria Sistemelor**

**Disciplina: Structuri de Date și Algoritmi**

**Lucrare de laborator nr.3**

**Tema: Operații asupra Fișierelor, Tuplurilor și Dicționare**

A efectuat:

Pleșca Virgiliu,

*student*

IA-201

A controlat:

Srună Vadim,

*Asistent. Univ.*

**Chișinău 2021**

# Sarcina

1. Scrieți un program care adaugă două numere primite de la utilizator (de la consolă) cu verificare de erori. Scrieți codul astfel încît să ceară utilizatorului un număr întreg și apoi verifică dacă ce a introdus utilizatorul e un număr valid. Afișează un mesaj de eroare în caz că numărul nu e valid și continuă afișarea mesajului ce promptează utilizatorului să introducă un număr pînă cînd introduce unul valid. Sfat: Folosiți input pentru a citi numărul ca un șir de caractere și apoi folosiți instrucțiunea while pentru a verifica că fiecare caracter este o cifră. Realizați același lucru și pentru al doilea număr și apoi adunați-le.

2. Scrieți o funcție care ar face liniarizarea unei liste. Aveti o listă imbricată (adică listă în listă) și creați o listă nouă.

De ex. liniarizare([ [1, 2, [3, 4] ], [5, 6], 7])

[1, 2, 3, 4, 5, 6, 7]

3. Scrieți un program pentru a găsi anagrame într-o anumită listă de cuvinte. Două cuvinte sunt numite anagrame dacă un cuvînt poate fi format prin rearanjarea literelor primului cuvînt. De exemplu, A, C, E, S, T: ACEST = ASCET = CASET = CASTE = CESTA = CETAS = SECAT = SECTA = SETCA = TEASC = TESAC sunt anagrame.

4. Scrieți o funcție *invertdict* care ar interschimba cheile și valorile într-un dicționar. Pentru simplificare, presupunem că toate valorile sunt unice.

5. Scrieți o funcție *mult\_lists(a,b)* care va lua ca parametri două liste de aceeași lungime și va returna suma produselor elementelor corespunzătoare a fiecărei liste

6. Se dă câteva tupluri care reprezintă persoane care folosesc laboratorul Visio, folosind numele și prenume:

brendan = ("Brendan", "McCane")

sandy = ("Sandy", "Garner")

nick = ("Nick", "Meek")

student1 = ("Allan", "Anderson")

student2 = ("Barry", "Byars")

student3 = ("Christine", "Carver")

student4 = ("Delia", "de Wattinger")

(a) Pune tuplele într-o listă, apoi utilizați un ciclu pentru a afișa doar fiecare prenume.

(b) Atribuie toate numele unei variabile de tip set numit Visio.

(c) Atribuie primele 3 tupluri unei variabile de tip set numit profesori.

(d) Atribuie restul la o variabilă set numit studenți.

(e) utilizează for pentru a imprima numele fiecărui elev, de exemplu corect formatat

Allan Anderson

(f) Utilizați un ciclu pentru a imprima numele fiecărui profesor folosind doar prima literă a prenumelui de ex B.

McCane

7. Scrieți un program care să citească un fișier ce conține un text. Programul va crea și va afișa pe ecran un dicționar în care vor fi stocate cuvintele și numărul de apariții ale acestora în textul din fișier. Îmbunătățește programul prin imprimarea cuvintele în ordinea descrescătoare a numărului de apariții.

8. Se dă un șir de caractere. Șirul de caractere de împărțit în fragmente a câte trei simboluri consecutive. În fiecare fragment simbolul de mijloc se înlocuiește cu un caracter aleatoriu care nu coincide cu oricare dintre caracterele acestui fragment. Fragmentele să fie sortate în ordine alfabetică.

9. Scrieți o funcție care are un dicționar d{int: int} ca intrare, și șterge din acest dicționar toate intrările căror chei sunt mai mari decît cel puțin una dintre valorile asociate cu celelalte chei din d, și returnează numărul de intrări care au fost eliminate.

# Mersul lucrării

## Codul programului

import random

import string

# 1.--------------------------------------------------------------

print("\n 1. ")

# Introducere suma a doua cifre de la tastatura cu verificarea erorilor

def \_1():

while True:

try:

# Incercam sa introducem

a = int(input(" a = "))

b = int(input(" b = "))

print(" a + b =", a + b)

break

except:

# Daca apar careva exceptii sau erori se va afisa urmatorul mesaj

# Apoi va avea loc urmatoarea iteratie

print(" Nu ati introdus un numar intreg.\nIncercati din nou")

\_1()

# 2.------------------------------------------------

# Lista imbricata

l = [[1, 2, [3, 4]], [5, 6], 7]

print("\n 2. ")

# Functia care liniarizeaza lista

def \_2(l):

for i in l:

if not isinstance(i, int):

yield from \_2(i)

else:

yield i

print(l)

print(" ", list(\_2(l)))

# 4.---------------------------------------------------

# Dictionarul

d = {

"a": 1,

"b": 2,

"c": 3,

"d": 4,

"e": 5

}

print("\n 4. ")

# Functia care interschimba cheile

def \_4(d):

# Extragem cheile si valorile in liste diferite

keys = list(d.keys())

values = list(d.values())

#

for i in range(len(keys)):

d[values[i]] = keys[i] # Cream un element cu cheia si valoarea inversata

d.pop(keys[i]) # Stergem cheia veche

print(" ", d)

\_4(d)

# 5.-----------------------------------------------------

# Doua liste cu lungimi egale

a = [1, 5, 8, 6, 4, 7, 8, 9, 5]

b = [5, 7, 8, 7, 4, 6, 1, 8, 7]

print("\n 5. ")

# Produsul unei liste

def prod(l):

p = 1

for i in l:

p \*= i

return p

# Suma produselor listelor

def \_5\_1(a, b):

if len(a) != len(b):

raise Exception("Liste cu lungimi diferite")

else:

return prod(a) + prod(b)

# Suma produselor fiecarui element din ambele liste

def \_5\_2(a, b):

if len(a) != len(b):

raise Exception("Liste cu lungimi diferite")

else:

s = 0

for i in range(len(a)):

s += a[i] \* b[i]

return s

print(" 5\_1. ", \_5\_1(a, b))

print(" 5\_2. ", \_5\_2(a, b))

# 6.------------------------------------------------------------

print("\n 6. ")

# Tuplurile date

brendan = ("Brendan", "McCane")

sandy = ("Sandy", "Garner")

nick = ("Nick", "Meek")

student1 = ("Allan", "Anderson")

student2 = ("Barry", "Byars")

student3 = ("Christine", "Carver")

student4 = ("Delia", "de Wattinger")

# -----------------------------------------

# (a) Pune tuplele într-o listă, apoi utilizați un ciclu pentru a afișa doar fiecare prenume.

print("a)")

l = [brendan, sandy, nick, student1, student2, student3, student4]

for i in l:

print(i[1]) # Printam elementul de pe poz 1 (al doilea element) din fiecare tuplu, aflat in lista

# (b) Atribuie toate numele unei variabile de tip set numit Visio.

print("b)")

Visio = [i[0] for i in l] # Folosind List comprehansion punem in Visio primul element din fiecare tuplu

print(Visio)

# (c) Atribuie primele 3 tupluri unei variabile de tip set numit profesori.

print("c)")

profesori = l[:3] # Folosind Slicing tool atribuim unei liste primele trei elemente (tupluri)

print(profesori)

# (d) Atribuie restul la o variabilă set numit studenți.

print("d)")

studenti = l[3:] # Folosind Slicing tool atribuim unei liste restul elementelor, adica elementele incepand cu indexul 3

print(studenti)

# (e) utilizează for pentru a imprima numele fiecărui elev, de exemplu corect formatat

print("e)")

for i in studenti:

print(i[0], i[1])

# (f) Utilizați un ciclu pentru a imprima numele fiecărui profesor folosind doar prima literă a prenumelui de ex B.

print("f)")

ch = "S"

for i in profesori:

if i[0][0] == ch:

print(i[1])

# 7. Facem un dictionar cu toate cuvintele din text si numarul lor de aparitie

# Citeste continutul fisierului si il atribuie unei variabile

f = open("file.txt", "r", encoding = "utf-8")

s = f.read()

f.close()

# Folosim metoda split() pentru a separa cuvintele

l = s.split()

d = { }

for word in l:

if not d.\_\_contains\_\_(word): # Astfel initialiam un element pentru a putea in continuare sa il incrementam

d[word] = 1

else:

d[word] += 1

# Dictionarul nu este un tip de date ordonat, sau care poate fi parcurs ca o liista simpla

# De aceea pentru a sorta dictionarul dupa valori cream un dictionar nou, unde vom plasa

# deja elementele in ordine crescatoare

d2 = { }

# Extragem valorile si cheile din dictionar pentru a putea parcurge si sorta valorile acestuia

keys = list(d.keys())

values = list(d.values())

ind = 0

max = 0

# Selection sort pentru dictionare

for i in range(len(d)):

for j in range(len(d)):

if d2.\_\_contains\_\_(keys[j]):

continue

if max < values[j]:

max = values[j]

ind = j

d2[keys[ind]] = max # Adaugam in alt dictionar elementele in ordine descrescatoare

max = 0

print("\n")

for item in d2.items():

print(item)

# 8.-------------------------------------------------

str = "Programul va crea și va afișa pe ecran un dicționar în care vor fi stocate cuvintele și numărul de apariții ale acestora în textul din fișier. Îmbunătățește programul prin imprimarea cuvintele în ordinea descrescătoare a numărului de apariții."

# Stergem spatiile goale din sirul de caractere

str = str.replace(" ", "")

l = []

# Facem o lista care contine sirul de caractere separat in cate 3 caractere

for i in range(2, len(str), 3): # Ciclu cu pasul de 3

l.append(str[i - 2] + str[i - 1] + str[i]) # Concatenam cate 3 caractere din sir

# Schimbam din fiecare element caracterul din mijloc cu unul aleator

for index, fragment in enumerate(l):

q = 1

while q == 1:

rand\_chr = random.choice(string.ascii\_letters)

if rand\_chr in fragment: # Verifica daca sirul nu contine acel caracter care urmeaza a fi introdus

continue

else:

new\_frag = fragment[0] + rand\_chr + fragment[2]

l[index] = new\_frag

q = 0

print(l)

# 9.------------------------------------------------------------------

# Dictionarul

d = {

1: 2,

2: 2,

5: 9,

7: 5,

9: 1

}

# Extragem elementele din dictionar pentru a putea opera cu ele in continuare

l = list(d.items())

print(d)

# Verifica fiecare element cu fiecare element daca cheia primului este mai mare decat

# valoare celui deal doilea

for i in range(len(l)):

for j in range(len(l)):

if l[i][0] > l[j][1]: # Daca a fost gasit un astfel de element el va fi sters

d.pop(l[i][0])

break

print(d)

# 3.---------------------------------------------------

print(" 3.-----------------------------------------------")

# Gaseste toate cuvintele anagrame din lista

def isAnagram(str1, str2):

str1\_list = list(str1)

str1\_list.sort()

str2\_list = list(str2)

str2\_list.sort()

return (str1\_list == str2\_list)

l = ['ACEST', 'ASCET', 'CAR', 'IEPURE', 'CASET', 'CASTE', 'CESTA', 'ALBUM', 'MARCEL']

a = []

for word1 in l:

for word2 in l:

if (word1 == word2):

continue

if (a.\_\_contains\_\_(word1) and a.\_\_contains\_\_(word2)):

continue

if isAnagram(word1, word2):

if (a.\_\_contains\_\_(word1)):

a.append(word2)

elif(a.\_\_contains\_\_(word2)):

a.append(word1)

else:

a.append(word2)

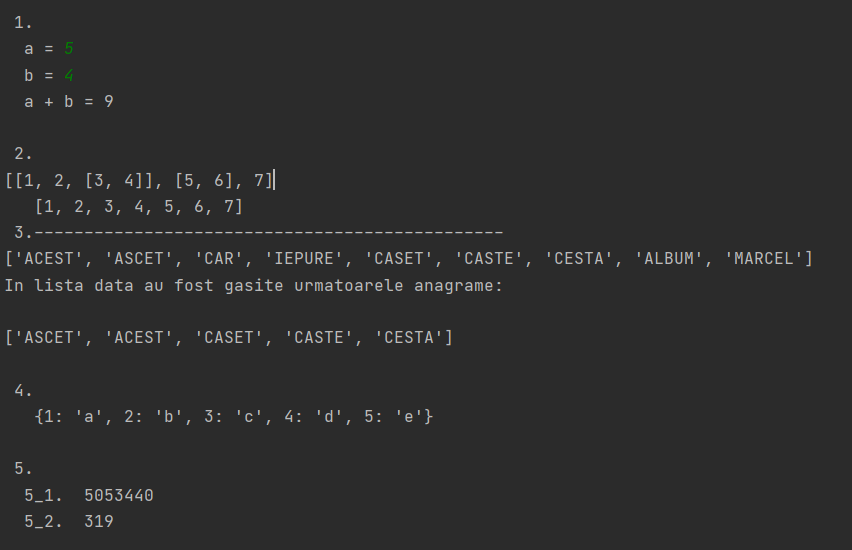
a.append(word1)

print(l)

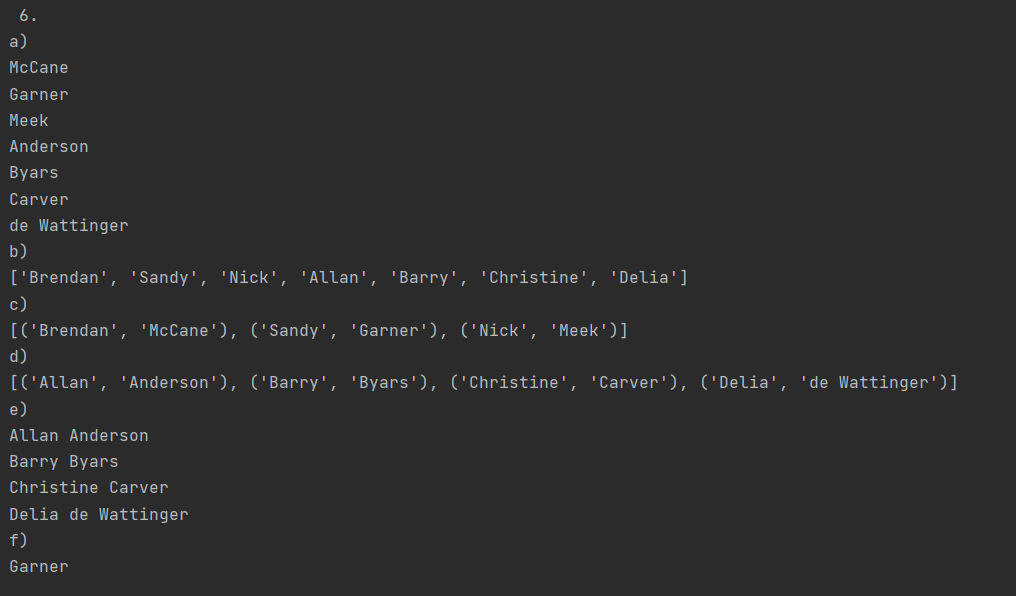
print("In lista data au fost gasite urmatoarele anagrame:\n")

print(a)

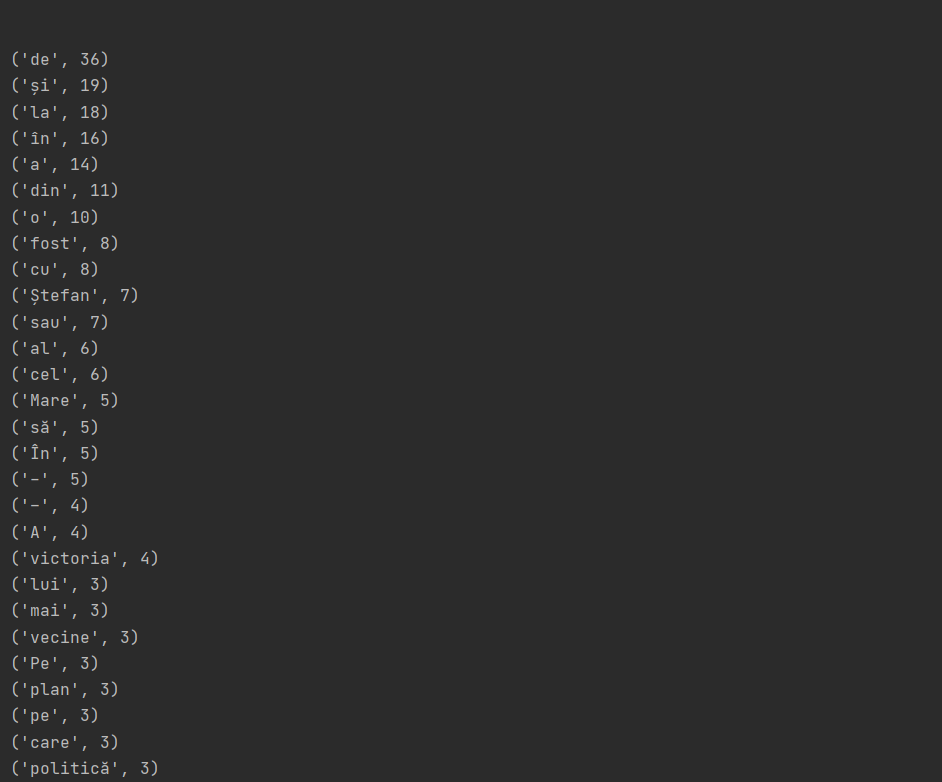
## Output



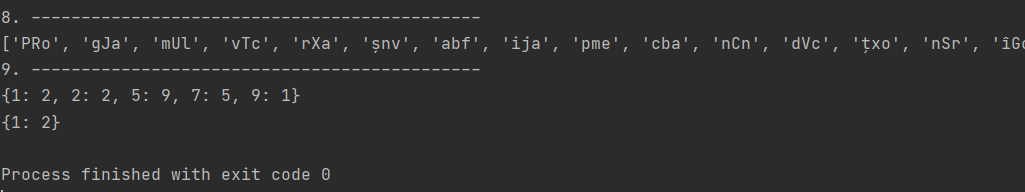
**Figura 1.2.1** – Output pentru problemele 1 – 5



**Figura 1.2.2** – Output pentru problema 6



**Figura 1.2.3** – Output pentru problema 7



**Figura 1.2.3** – Output pentru problemele 8 și 9

# Concluzie

În urma elaborării acestei lucrări de laborator am folosit listele, tuplurile și dicționarele pentru a rezolva problemele propuse. La fel am folosit si functiile pentru deschiderea fisierelor, modificarea, citirea, inchiderea etc. și pot spune că este cu mult mai simplu decât în alte limbaje de programare să operezi cu fișierele. Am înțeles cum lucrează mecanismul try except. Am studiat tipurile de erori care pot apărea într-un program Python și totodată am înțeles cum putem opera cu ele iarăși cu ajutorul try except. În general pot spune că am înțeles destul de bine aceste teme, plus am mai studiat ce este un generator, el defapt este este o functie care returneaza folosind cuvântul predefinit ***yield***, un obiect de tip generator care poate fi folosit ca o lista, doar ca nu ocupă memorie in calculator pentru elemente, el le genereaza dupa formula corespunzatoare.