**SEMINARUL 4**

**EXEMPLU** – **Fișiere**

|  |  |
| --- | --- |
| **f=open("citire.in") *#implicit deschis pt citire* s=f.readline() *#include in s si sfarsitul de linie* print(repr(s)) *#str(S)* print(s) print("-------------") f.close()** | **f=open("citire.in") s=f.read() *#returneaza in s tot fisierul(ca str)***  **print(repr(s)) print(s) print("-------------") f.close()** |
| **f=open("citire.in") *#implicit deschis pt citire***  **#f=open("citire.in","r")**  **ls=f.readlines() #*returneaza o lista avand ca elemente liniile din fisier***  **print(repr(ls)) #str(s)**  **print(ls)**  **print("---------------")**  **f.close()** | **#*fisierul standard de intrare (cin)-citire sir pe mai multe linii***  **import sys**  **s=sys.stdin.read()**  **print(s)**  **for linie in sys.stdin:  #print(linie,end="")  sys.stdout.write(linie)** |
| ***#citire linie cu linie* with open("citire.in", "r") as f:**  ***#nu mai trebuie f.close()*  linie = f.readline()  while linie !="":  print(linie, end='')**  ***#linie contine \n* linie = f.readline()** | **#citire linie cu linie f=open("citire.in")  for linie in f:  print(linie, end='')**  **f.close()**  **#linie contine \n** |
| **#scriere - exemplu g=open("citire.out","w")**  **#nu adauga \n**  **g.write("un sir") #un sir de caractere g.writelines(["o linie","alta linie"]) #o lista de siruri de caractere**  **g.write("\n")**  **g.write("un sir\n") #un sir de caractere g.writelines(["o linie\n","alta linie"]) #o lista de siruri de caractere**  **g.close()** | **with open("citire.in", 'r') as f, open("citire.out", 'w') as g:  for linie in f:  g.write(linie) with open("citire.in", 'r') as f, open("citire.out", 'w') as g:  linii = f.read()  g.write(linii)** |

1. Se citește din fișierul "numere.in" un șir de cuvinte separate prin spațiu. Unele dintre cuvinte sunt numere naturale. Să se afișeze în fișierul “numere.out” media aritmetică a cuvintelor din șir care sunt numere naturale. Mai exact, rezultatul va fi un șir de forma: “(nr1+...nr\_k)/n=media”, unde nr1,...,nrk sunt numerele care apar în șir și media este afișată cu 2 zecimale

**with open("numere.in") as f,open("numere.out","w") as g:  
 ls=[x for x in f.readline().split() if x.isdigit()]  
 rez = "+".join(ls)  
 ls\_numere=(int(x) for x in ls) #GENERATOR  
 rez=f"({rez})/{len(ls)} ={sum(ls\_numere)/len(ls):.2f}"  
 g.write(rez)**

**sau:**

f = open(**"numere.in"**)  
g = open(**"numere.out"**, **"w"**)  
p = f.readline()  
p = [x **for** x **in** p.split() **if** x.isdigit()]  
v = (int(x) **for** x **in** p)  
g.write(**f'({ "+"**.join(p)**})/{** len(p)**} ={** sum(v)/len(p)**:.2f}'**)  
f.close()  
g.close()

**Problema 1, varianta in care se iau toate numerele din fisier(si cele negative):**

f = open(**"numere.in"**)  
g = open(**"numere.out"**, **"w"**)  
p = f.readline()  
lsa = [chr(i) **for** i **in** range(97, 123)] *#97 e codul ascii al lui a, iar 123 al lui z*lsa = **""**.join(lsa)  
lsA = [c.upper() **for** c **in** lsa]  
lsA = **""**.join(lsA) *#ABCDEFGHIJKLMNOPQRSTUVWXYZ*s = lsa+lsA+**" "** *#s=abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ*tabel = str.maketrans(s, **""**.join([**" "**]\*53))  
p = p.translate(tabel) *#stergem literele astfel incat sa ramana doar numerele din fisier*p = p.split() *#le punem intr-o lista (numerele din fisier)*v = [int(x) **for** x **in** p]  
g.write(**f'({"+"**.join(p)**})/{**len(v)**}={**sum(v)/len(v)**:.2f}'**)  
f.close()  
g.close()

2. Fișierul text “numere\_comune.in” conține numere naturale despărțite prin spații și scrise pe mai multe linii.

a) Să se afișeze în fișierul “comune.out” numerele care apar pe toate liniile din fișier (folosind set + intersectie).

**Exemplu:** dacă fișierul “numere\_comune.in” conține

2 1 5 1 3

1 4 2 2

2 1 1 6 8

atunci fișierul “comune.out” va conține (numerele nu neapărat în această ordine):

1 2

b) Modificați programul astfel încât numerele să fie afișate crescător.

f = open(**"numere\_comune.in"**)  
g = open(**"numere\_comune.out"**, **"w"**)  
s = {int(x) **for** x **in** f.readline().split()} *#elemente distincte de pe prima linie***for** linie **in** f:  
 s = s & set((int(x) **for** x **in** linie.split()))

*#in set-uri vom avea elementele distincte de pe celelalte linii  
#s&set... face intersectia tuturor elementelor***for** x **in** s:  
 g.write(str(x)+**" "**) *# g.write(f"{x} ")*f.close()  
g.close()

4. a) Scrieți o funcție care, dat ca parametru numele unui fișier, determină și returnează frecvența caracterelor din fișier

b) Se consideră fișierele caractere1.in si caractere2.in. Să se afișeze pentru fiecare fișier frecvența caracterelor.

c) Să se afișeze caracterele comune celor două fișiere și frecvența cu care se repetă în ambele fișiere (se vor afișa întâi caracterele alfanumerice, apoi celelalte). **frecventa\_Caractere\_fisier.py**

**#frecvente caractere fisier**

**def** frecventa\_caractere(nume\_fisier):  
 d={} *#d=dict() , d[caracter]=frecventa* **with** open(nume\_fisier) **as** f:  
 **for** linie **in** f:  
 **for** caracter **in** linie:  
 d[caracter]=d.get(caracter, 0)+1  
 **return** d  
  
frecv1=frecventa\_caractere(**"caractere1.in"**)  
print(frecv1)  
frecv2=frecventa\_caractere(**"caractere2.in"**)  
print(frecv2)  
*#frecv1.keys() - multime***for** x **in** sorted(frecv1.keys()&frecv2.keys(), key=**lambda** x: 0 **if** x.isalnum() **else** 1):  
 print(**f"{**repr(x)**} frecventa {**min(frecv1[x], frecv2[x])**}"**)

**sau:**

**def** frecventa(nume\_fisier):  
 frecv = {}  
 f = open(nume\_fisier)  
 **for** line **in** f:  
 **for** i **in** line:  
 **if** i **in** frecv:  
 frecv[i] += 1  
 **else**:  
 frecv[i] = 1  
 f.close()  
 **return** frecv  
**def** cheie(i):  
 **if** i.isalnum():  
 **return** (0, i)  
 **else**:  
 **return** (1, i)  
f1 = frecventa(**"caractere.in"**)  
f2 = frecventa(**"caractere2.in"**)  
**for** i **in** sorted(f1, key=cheie):  
 **if** i **in** f2:  
 print(repr(i), min(f1[i], f2[i]))  
   
 *# f=f1.keys() & f2.keys()   
# for i in sorted(f,key=cheie):   
# print(repr(i),min(f1[i],f2[i])*

5. Se citesc de la tastatură două numere naturale n și m.

a) Să se scrie în fișierul matrice.in o matrice cu n linii și m coloane cu elemente mai mici decât 100 generate aleator (fără a o memora).

b) Se citește din fișierul matrice.in matricea generata la a). Să se genereze în memorie si să se scrie în fișierul matrice.out transpusa matricei și matricea obținută din matricea inițială ordonând crescător elementele de pe ultima coloana prin interschimbări de linii

**import** random *#pentru a genera numere random***def** afis(a,g): *#a este matricea* **for** linie **in** a:  
 **for** x **in** linie:  
 g.write(**f'{**x**:4} '**)  
 *#str\_linie=" ".join([f'{x:4} ' for x in linie])  
 #g.write(str\_linie)* g.write(**"\n"**)  
  
 *#varianta cu o singura scriere:  
 #g.write("\n".join([ " ".join([f'{x:4} ' for x in linie]) for linie in a]))*f=open(**"matrice.in"**,**"w"**)  
n,m=[int(x) **for** x **in** input(**"dati dimensiunile matricei separate prin spatiu: "**).split()]  
f.write(**f'{**n**} {**m**}\n'**)  
**for** i **in** range(n):  
 **for** j **in** range(m):  
 x=random.randint(1,99)  
 f.write(**f'{**x**} '**)  
 f.write(**'\n'**)  
f.close()  
  
f=open(**"matrice.in"**)  
n,m=(int(x) **for** x **in** f.readline().split())  
a=[]  
**for** linie **in** f:  
 a.append([int(x) **for** x **in** linie.split()])  
*#varianta condensata:  
# a=[[int(x) for x in linie.split()] for linie in f]*print(a) *#se afiseaza o lista cu liniile matricei*f.close()   
  
*#ordonare dupa ultima coloana*g=open(**"matrice.out"**,**"w"**)  
g.write(**"matricea ordonata\n"**)  
**def** cheie\_ult\_coloana(linie):  
 **return** linie[-1]  
*#aord=sorted(a,key=lambda linie:linie[-1])*aord=sorted(a,key=cheie\_ult\_coloana)  
afis(aord,g)  
  
g.write(**"matricea transpusa varianta 1\n"**)  
tr=[[0 **for** i **in** range(n)] **for** j **in** range(m)]  
**for** i **in** range(n):  
 **for** j **in** range(m):  
 tr[j][i]=a[i][j]  
afis(tr,g)  
  
g.write(**"matricea transpusa varianta 2\n"**)  
tr2=[[linie[j] **for** linie **in** a] **for** j **in** range(m)]  
afis(tr2,g)  
  
g.write(**"matricea transpusa varianta 3\n"**)  
tr3=[[a[i][j] **for** i **in** range(n)] **for** j **in** range(m)]  
afis(tr3,g)  
  
g.close()

6. Se dă un fișier cu cuvinte separate prin spatii. Scrieți un program care să determine grupurile de cuvinte din fișier care au aceleași litere (nu neapărat cu aceeași frecvență). Numele fișierului de intrare se va citi de la tastatură, iar grupurile formate din cel puțin două cuvinte se vor scrie în fișierul text “litere.txt”, câte un grup pe o linie. Cuvintele din fiecare grup vor fi sortate după lungime, **iar în caz de lungimi egale, lexicografic**.

Pentru fișierul de intrare:

apar mare  
si amara rapa para  
par isi rama

fișierul de ieșire va fi

par apar para rapa  
si isi  
rama amara

**Varianta cu dictionar**

f = open(input())

g = open("litere.txt", "w")

d = {}

for linie in f:

    for x in linie.split():

        litere = frozenset(x)

        if litere in d:

            d[litere].append(x)

        else:

            d[litere] = [x]

for ls in d.values():

    ls.sort(key=lambda x: (len(x), x))

    if len(ls) >= 2:

        for i in ls:

            g.write(f"{i} ")

        g.write("\n")

g.close()

f.close()

**Varianta cu liste (compar literele dintr-un cuvant cu cele de dupa + marchez cele deja considerate in grupuri)**

f=open("cuvinte.in")

g=open("cuvinte.out","w")

ls=f.read().split()

v=[0 for i in range(len(ls))]

def cheie(x):

    return (len(x),x)

for i in range(len(ls)):

    if v[i]==0:

        v[i]=1

        lista=[ls[i]]

        for j in range(i+1,len(ls)):

            if set(ls[i])==set(ls[j]):

                lista.append(ls[j])

                v[j]=1

        if len(lista)>1:

            lista.sort(key= cheie)

            for x in lista:

                g.write(x+" ")

            g.write("\n")

f.close()

g.close()

7. Se consideră un fișier de intrare magazin.in cu informații despre magazine, produse și cantitatea de produse din fiecare magazin, sub forma:

**Magazin** cod\_magazin nume\_magazin

cantitate nume\_produs

...

cantitate nume\_produs

a) Să se memoreze datele astfel încât să răspundă la interogări de tipul: dat codul magazinului și numele unui produs, să se afișeze cantitatea de produs din magazin (să se interogheze pentru un cod și un nume date de la tastatură), dat codul magazinului, numele unui produs și o cantitate de produs care se vinde, actualizează stocul de produs din magazin dacă se poate (să se facă o astfel de actualizare pentru cod, nume produs, cantitate date de la tastatură)

b) Să se afișeze o lista de tupluri (magazin, lista de (produse:cantitate)) ordonată după cantitatea totală de marfă din magazin, și, în caz de egalitate, după numele magazinului

c) Să se afișeze o listă a tuturor produselor care se găsesc în magazine (folosind reuniune de mulțimi)

Exemplu:

Magazin 123 magazin1  
5 mere  
7 pere  
2 prune  
Magazin 221 magazin 2  
3.5 pere  
10 banane

**punctul c, facut la consultatii:**

d={}  
f=open (**"magazin.in"**)  
**for** linie **in** f:  
 ls=linie.split()  
 **if** ls[0]==**"Magazin"**:  
 cod\_magazin=int(ls[1])  
 d[cod\_magazin]={}  
 **else**:  
 cantitate=float(ls[0])  
 produs=ls[1]  
 d[cod\_magazin][produs]=cantitate  
print (d) *#se va afisa: {123: {'mere': 5.0, 'pere': 7.0, 'prune': 2.0}, 221: {'pere': 3.5, 'banane': 10.0}}*print (d[123][**"pere"**]) *#se va afisa 7.0*

*#c)Lista tuturor produselor din magazin, fara cantitate(doar numele)*prod\_reun=set()  
**for** magazin **in** d.values():  
 prod\_reun = prod\_reun | magazin.keys()  
print (prod\_reun) *#{'banane', 'mere', 'pere', 'prune'}*

**varianta intreaga:**

f=open(**"magazin.in"**)  
cod=0  
d={}  
**for** linie **in** f:  
 ls=linie.split(maxsplit=1)  
 **if** ls[0]==**"Magazin"**:  
 *#ls[1]="cod nume\_magazin"* lsinfo=ls[1].split(maxsplit=1)  
 cod=int(lsinfo[0])  
 nume\_magazin=lsinfo[1].rstrip(**"\n"**)  
 d[cod]=(nume\_magazin, {})  
 **else**:  
 cantitate=float(ls[0])  
 nume\_produs=ls[1].rstrip(**"\n"**)  
 d[cod][1][nume\_produs]=cantitate  
f.close()  
print(d)  
  
**def** cantitate\_produs\_magazin(d,cod\_magazin,nume\_produs):  
 **if** cod\_magazin **in** d:  
 **return** d[cod\_magazin][1].get(nume\_produs,0)  
 **return** 0  
  
**def** actualizeaza\_stoc(d,cod\_magazin,nume\_produs,cantitate):  
 **if** cod\_magazin **not in** d:  
 **return False  
 if** d[cod\_magazin][1].get(nume\_produs, 0) >= cantitate:  
 d[cod\_magazin][1][nume\_produs] = d[cod\_magazin][1][nume\_produs] - cantitate  
 **if** d[cod\_magazin][1][nume\_produs] == 0:  
 d[cod\_magazin][1].pop(nume\_produs)  
 **return True  
 else**:  
 **return False**print(cantitate\_produs\_magazin(d,221,**"mere"**))  
print(cantitate\_produs\_magazin(d,221,**"pere"**))  
print(cantitate\_produs\_magazin(d,111,**"mere"**))  
  
print(actualizeaza\_stoc(d,221,**"pere"**,1))  
print(d)  
print(actualizeaza\_stoc(d,221,**"pere"**,7))  
print(d)  
print(actualizeaza\_stoc(d,221,**"pere"**,2.5))  
print(d)  
  
**def** cheie\_magazin(x):  
 **return** (sum(x[1].values()),x[0])  
  
lista\_valori=d.values()  
print(sorted(lista\_valori,key=cheie\_magazin ))  
  
s=set() *#multimea vida***for** magazin **in** d.values():  
 s1=magazin[1].keys()  
 s=s|s1  
print(s)