**SEMINARUL 3**

**str.translate(tabel\_traducere)**

* **tabel\_ traducere** – de obicei obținut cu **str.maketrans(x[, y[, z]])**

(sau dicționar cu coduri Ascii)

**str.maketrans(x[, y[, z]])**

* **x** - dicționar sau șirul cu caracterele de înlocuit
* **y** – șirul cu caracterele noi (de aceeași lungime cu x) => caracterul **x[i]** se va înlocui cu **y[i]**
* **z** – șir cu caracterele care vor fi șterse

s=**"abcdeabsdcabc"**   
*#inlocuim a->x, b->y*   
tabel=**""**.maketrans(**"ab"**,**"xy"**)   
*#tabel=str.maketrans("ab","xy")*   
print(tabel,type(tabel))   
s=s.translate(tabel)   
print(s)   
   
s=**"abcdeabsdcabcef"**   
*#inlocuim a->x, b->y si stergem literele e si s*   
tabel=str.maketrans(**"ab"**,**"xy"**,**"es"**)   
print(tabel,type(tabel))   
s=s.translate(tabel)   
print(s, type(s))

1.Se citește un text conținând separatorii uzuali( ,.;:) Sa se înlocuiască toți separatorii cu spațiu.

*s=***"ab, .. as: sdf;sdf"**   
**tabel=**str.maketrans(**",.;:"**,**" "**\*4)   
s=s.translate(tabel)   
print(s)

2. Se citește un cuvânt format cu litere mici. Sa se înlocuiască fiecare vocală din cuvânt cu următoarea literă din alfabet.

s = **"aextruotedf"**vocale = **"aeiou"**

consoane=[]  
**for** x **in** vocale:   
 consoane.append(chr(ord(x)+1)) ***#ord(x)-codul ascii al lui x, chr(x) - caracterul cu codul x****#consoane = [chr(ord(x) + 1) for x in vocale] - echivalentul for-ului*tabel = str.maketrans(vocale, **""**.join(consoane))  
s = s.translate(tabel)  
print(s)

3.  Aceeași cerință ca la 2, dar în plus să se șteargă semnele: virgula, punct, două puncte.

s=**"aextr ,uoted:;f '"**vocale=**"aeiou"  
  
"""consoane=[]   
for x in vocale:   
 consoane.append(chr(ord(x)+1)) #ord(x)-codul lui x, chr(x) - caracterul cu codul x   
"""**consoane=[chr(ord(x)+1) **for** x **in** vocale]  
print(consoane)  
tabel=str.maketrans(vocale,**""**.join(consoane),**" ,:;'"**)  
s=s.translate(tabel)  
print(s)

4. Se citește o propoziție. Să se înlocuiască fiecare cifră < 5 care apare în text cu denumirea ei (1-unu, 2-doi, 3- trei, 4 -patru)

s=**"1 ora de seminar 21 ore de pauza"**d={**"1"**:**"unu"**, **"2"**:**"doi"**, **"3"**:**"trei"**,**"4"**:**"patru"**}  
tabel=str.maketrans(d)  
s=s.translate(tabel)  
print(s)

**PROBLEME CU ȘIRURI ȘI LISTE. COMPREHENSIUNE**

**OPERATII LISTE**

1. Se dau două liste l1 si l2 de lungime n. Să se înlocuiască elementele de pe poziții pare din l1 cu cele de pe poziția corespunzătoare din l2 folosind feliere (slice) .

n=int(input())  
l1=[int(input()) **for** x **in** range (0,n)] *#citire cu enter*l2=[int(x) **for** x **in** input().split()] *#citire de pe aceeasi linie*l1[::2]=l2[::2]  
print(l1)

2. Se dă o listă de numere naturale. Să se șteargă din listă subsecvența delimitată de primele două zerouri din listă (inclusiv zerourile).

l = [int(x) **for** x **in** input().split()] *#citire pe o singura linie***"""if l.count(0)>=2:   
 i1=l.index(0)   
 i2=l.index(0,i1+1) #a doua aparitie   
 l[i1:i2+1]=[] # del l[i1:i2+1]   
else:   
 print("Nu are doua cel putin doua zerouri")   
print (l)   
"""**

**try**:  
 i1 = l.index(0) *#prima aparitie* i2 = l.index(0, i1 + 1) *# a doua aparitie* l[i1:i2 + 1] = [] *# del l[i1:i2+1]***except**:  
 print(**"Nu are doua cel putin doua zerouri"**)  
print(l)

3. Se dă o listă de numere naturale. Să se șteargă din listă toate zerourile.

l = [int(x) **for** x **in** input().split()]  
**while** 0 **in** l:  
 l.remove(0)   
print (l)   
  
**"""try:  
 i1 = l.index(0)  
 del l[i1] # l.pop (i1) - intoarce pe cine elimina  
 while True:  
 i1 = l.index(0, i1)  
 del l[i1]  
except:  
 pass  
print(l)  
"""**

sau:

l = [int(x) **for** x **in** input().split()]  
l1=[x **for** x **in** l **if** x!=0]  
print(l1)

4.Se dă o listă de numere naturale și un număr natural k. Să se elimine din listă subsecveța de lungime k de sumă minimă (dacă sunt mai multe se va elimine prima = cea mai din stânga) – fără a folosi liste suplimentare

s=[int(x) **for** x **in** input().split()]  
k=int(input())  
suma=suma\_min=sum(s[:k])  
poz=0  
**for** i **in** range(k,len(s)):  
 suma=suma+s[i]-s[i-k]  
 **if** suma<suma\_min:  
 suma\_min=suma  
 poz=poz+1  
s[poz:poz+k]=[]  
print(s)

6. Se dă o listă de numere reale (toate elementele sale se vor da pe o linie separate prin spațiu). Să se insereze câte un 0 după fiecare element negativ (fără a folosi liste suplimentare).

lista=[float(x) **for** x **in** input().split()]  
**for** i **in** range(len(lista)):  
 **if** lista[i] < 0:  
 lista.insert(i+1,0)  
print (lista)

**COMPREHENSIUNE**

7. Se citește o propoziție cu cuvintele separate prin spatii (unul sau mai multe). Să se creeze o listă cu cuvintele care încep cu o vocală (folosind și comprehensiune).

simplu:

prop=input()  
lista\_cuvinte=prop.split()  
lista\_noua=[]  
**for** x **in** lista\_cuvinte:  
 **if** x[0] **in "aeiouAEIOU"**:  
 lista\_noua.append(x)  
print(lista\_noua)

cu comprehensiune:

prop=input()  
lista\_noua=[x **for** x **in** prop.split() **if** x[0] **in "aeiouAEIOU"**]  
print(lista\_noua)

8. Se citește o propoziție cu cuvintele separate prin spațiu.

Să se creeze o listă cu cuvintele care conțin minim 3 vocale (folosind și comprehensiune).

simplu:prop=input()  
lista\_cuvinte=prop.split()  
lista\_noua=[]  
**for** cuv **in** lista\_cuvinte:  
 nr=0  
 **for** i **in** range(0, len(cuv)):  
 **if** cuv[i] **in "aeiouAEIOU"**:  
 nr=nr+1  
 **if** nr>=3:  
 lista\_noua.append(cuv)  
 **break**print(lista\_noua)

cu comprehensiune:

nr=0  
prop=input()  
lista\_cuvinte=prop.split()  
lista\_noua=[cuv **for** cuv **in** lista\_cuvinte **if** sum([cuv.count(i) **for** i **in "aeiouAEIOU"**])>2 ]  
print(lista\_noua)

9. **Cifrul lui Cezar**

a)Să se creeze în memorie  o listă cu literele mici ale alfabetului.

*#a) lista cu literele mici***import** string

**"""nr\_litere=ord('z')-ord('a')+1  
litere\_mici=[chr(x+ord('a')) for x in range(nr\_litere)]  
print(litere\_mici)"""**

litere\_mici = [chr(x) **for** x **in** range(ord(**'a'**), ord(**'z'**) + 1)]  
print(litere\_mici)

**"""litere\_mici = list(string.ascii\_lowercase) #este str  
print(litere\_mici)"""**

b)Se citește un text ce conține numai litere mici ale alfabetului englez și semnele de punctuație uzuale și un număr natural k. Să se afișeze textul cifrat cu cifrul lui Cezar, prin care fiecare literă dintr-un text dat este înlocuită cu litera aflată peste 𝑘 poziții la dreapta în alfabet în mod circular (valoarea 𝑘 reprezintă cheia secretă comună pe care trebuie să o cunoască atât expeditorul, cât și destinatarul mesajului criptat). 

*#b)-Codificare  
#varianta 1- ls= lista cu caracterele modificate; sirul codificat se obtine din ls cu join*s=input()  
s1=s  
k=int(input())  
ls=[]  
nr\_litere=ord(**'z'**)-ord(**'a'**)+1  
**for** x **in** s:  
 **if** x.isalpha():  
 c=(ord(x)-ord(**'a'**)+k)%nr\_litere *#a cata litera este* ls.append(chr(c+ord(**'a'**)))  
 **else**:  
 ls.append(x)  
s=**""**.join(ls)  
print(s)  
  
*#varianta 2 - cu comprehensiune*  
ls=[chr(ord(**'a'**)+(ord(x)-ord(**'a'**)+k)%nr\_litere) **if** x.isalpha() **else** x **for** x **in** s ]  
s=**""**.join(ls)  
print(s)  
  
*#varianta 3- cu maketrans*  
litere\_mici\_transf=[chr(ord(**'a'**)+(ord(x)-ord(**'a'**)+k)%nr\_litere) **for** x **in** litere\_mici]  
tabel=str.maketrans(**""**.join(litere\_mici),**""**.join(litere\_mici\_transf))  
s=s.translate(tabel)  
print(s)

10. Se dă o propoziție care conține cuvinte separate prin separatorii uzuali ( ,.;:). O parte din cuvinte reprezintă numere naturale. Să se determine media aritmetică a cuvintelor din șir care sunt numere. 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.**

propozitie=input()  
tabel=str.maketrans(**",.;:"**,**" "**\*4)  
propozitie=propozitie.translate(tabel)  
lista\_cuvinte=propozitie.split()  
suma=0  
nr=0  
**for** i **in** lista\_cuvinte:  
 **if** i.isdigit():  
 suma=suma+int(i)  
 nr+=1  
print(**"Media aritmetica este "**, end=**''**)  
print(**"{:.2f}"**.format(suma/nr))

11. Se citesc doua cuvinte formate doar din litere mici. Să se verifice dacă sunt anagrame.

*"""Anagrame"""*a=input(); b=input()  
ok=**False  
if** len(a)==len(b):  
*#varianta 1 - numaram litere cu count O(n2)* **for** x **in** a: *#suficient in set(a)* **if** b.count(x)!=a.count(x):  
 **break  
 else**:  
 ok=**True**print(ok)  
  
*#varianta 2 - cu sortari*ok= (sorted(a) ==sorted(b))  
print(ok)  
  
*#varianta 3 - dictionar de frecventa O(n)  
#3a*d1 = {}  
**for** x **in** a:  
 **if** x **in** d1:  
 d1[x] += 1  
 **else**:  
 d1[x] = 1  
print(**"d1="**,d1)  
  
*#3b*d2 = {x:0 **for** x **in** set(a)}  
*#SAU: d2=dict.fromkeys(set(s),0)***for** x **in** a:  
 d2[x] = d2[x]+ 1  
print(**"d2="**,d2)

*#3c*da = {}  
**for** x **in** a:  
 da[x] = da.get(x, 0) + 1  
print(da)  
  
db = {}  
**for** x **in** b:  
 db[x] = db.get(x, 0) + 1  
print(db)  
  
ok= (da==db)  
print(ok)  
  
  
*#varianta 4 - vector de frecvente O(n)+stabila*nr\_litere=ord(**'z'**)-ord(**'a'**)+1  
fa=[0]\*nr\_litere  
fb=[0 **for** i **in** range(nr\_litere)]  
**for** x **in** a:  
 fa[ord(x)-ord(**'a'**)]+=1  
**for** x **in** b:  
 fb[ord(x) - ord(**'a'**)] += 1  
ok=(fa==fb)  
print(ok)

**MATRICE**

12. Se citesc n, m și o matrice cu n linii si m coloane (numerele sunt date câte unul pe linie). Să se ordoneze crescător elementele de pe prima coloana prin interschimbari de linii.

matrice=[]  
n=int(input(**"Numar linii: "**))  
m=int(input(**"Numar coloane: "**))  
  
**for** i **in** range(n):  
 linie=[int(x) **for** x **in** input().split()]  
 matrice.append(linie)  
  
**for** i **in** range(m):  
 **for** j **in** range(i + 1, m):  
 **if** matrice[i][0] > matrice[j][0]:  
 matrice[i], matrice[j] = matrice[j], matrice[i]  
  
**for** linie **in** matrice: ***#afisare matrice***  
 **for** x **in** linie:  
 print(**f"{**x**:4} "**,end=**""**)  
 print()

13. Se da un număr natural n>2. Sa se afiseze primele n linii din triunghiul lui Pascal (daca c este numărul maxim de cifre ale unui număr din triunghi, toate numerele se vor afișa pe c+1 caractere)

*"""Triunghi Pascal"""*n=int(input())  
triunghi=[[0 **for** i **in** range(j+1)] **for** j **in** range(n)]  
  
**for** i **in** range(n):  
 triunghi[i][0]=triunghi[i][-1]=1 *#am pus 1 pe marginile triunghiului***for** i **in** range(2,n):  
 lung=len(triunghi[i])  
 **for** j **in** range(1,lung-1):  
 triunghi[i][j] = triunghi[i-1][j] +triunghi[i-1][j-1] *#completam ce a ramas din fiecare linie*nr\_max=triunghi[n-1][(n-1)//2] *#cel mai mare numar din triunghi*nr\_cifre=len(str(nr\_max))+1 *#nr de cifre=lungimea lui privit ca sir***for** linie **in** triunghi:  
 **for** x **in** linie:  
 print(**f"{**x**:{**nr\_cifre**}}"**,end=**""**,sep=**""**)  
 print()  
  
**"""  
#varianta afisare comprehensiune pentru fiecare linie:  
for linie in triunghi:  
 ls=[f"{x:{nr\_cifre}}" for x in linie]  
 print("".join(ls))  
"""  
"""   
#varianta afisare comprehensiune:  
ls="\n".join(["".join([f"{x:{nr\_cifre}}" for x in linie]) for linie in triunghi])  
print(ls)"""**

14. Se citesc m, n și o matrice cu m linii și n coloane, elementele unei linii fiind date pe o linie (elementele unei linii date pe o linie separate cu spațiu). Să se construiască în memorie și să se afiseze matricea transpusă.

m,n=(int(x) **for** x **in** input().split())  
**"""  
#varainta detaliata:  
ls=input().split()  
m=int(ls[0])  
n=int(ls[1])  
"""  
  
""" m,n=(int(x) for x in input().split())  
#var 1 - initializat apoi citit c++ style  
a=[[0 for j in range(n)] for i in range(m)]  
for i in range(m):  
 ls=input().split()  
 for j in range(n):  
 a[i][j]=int(ls[j])  
print(a)"""***#varianta 2*a=[]  
**for** i **in** range(m):  
 a.append([int(x) **for** x **in** input().split()])  
  
print(a)  
  
*#transpusa - varianta 1*t=[[0 **for** j **in** range(m)] **for** i **in** range(n)]  
**for** i **in** range(n):  
 **for** j **in** range(m):  
 t[i][j]=a[j][i]  
  
print(t)  
  
*#transpusa - varianta 2*t=[]  
**for** i **in** range(n):  
 ls=[]  
 **for** linie **in** a:  
 ls.append(linie[i])  
 t.append(ls)  
print(t)  
  
*#transpusa - varianta 2 cu comprehension pe linie*t=[]  
**for** i **in** range(n):  
 t.append([linie[i] **for** linie **in** a])  
print(t)  
  
*#transpusa cu comprehension*t=[[linie[i] **for** linie **in** a] **for** i **in** range(n)]  
print(t)