**<https://docs.python.org/3/tutorial/datastructures.html>**

**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=**"abc axksdf sdfa"**  
**#a->b, x->y**  
**#tabel="".maketrans("ax","by") #"\*".join(ls)**  
**tabel=**str.maketrans(**"ax"**,**"by"**)  
print(tabel, type(tabel))  
s=s.translate(tabel)  
print(s)  
tabel=str.maketrans(**""**,**""**,**"df"**)  
print(tabel, type(tabel))  
s=s.translate(tabel)  
print(s)

**EXEMPLE**

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

#1. Se citește un text conținând separatorii uzuali( ,.;:) Sa se înlocuiască toți separatorii cu spațiu.  
s=**"a ,.: re:,. ewr,mer"**  
**tabel=**str.maketrans(**",:;."**,**" "**\*4)  
s=s.translate(tabel)  
print(s)

1.2. Se citește un cuvânt format cu litere mici. Sa se înlocuiască fiecare vocală din cuvânt cu următoarea literă din alfabet.  
s=**"abcaeiofgu"**  
**vocale="aeiou"**  
**inlocuit=""**  
**for** x in vocale: *#x de tip str*  
c=chr(ord(x)+1)  
 inlocuit=inlocuit+c  
print(inlocuit)  
#varianta 2  
inlocuit = [chr(ord(x)+1) for x in vocale]  
print(inlocuit)  
inlocuit=**""**.join(inlocuit)  
  
tabel=str.maketrans(vocale,inlocuit)  
s=s.translate(tabel)  
print(s)

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

#3. Aceeași cerință ca la 2, dar în plus să se șteargă semnele: virgula, punct, două puncte.  
s=**"a,.:bca;e.iofgu"**  
**tabel=**str.maketrans(vocale,inlocuit,**",;.:"**)  
s=s.translate(tabel)  
print(s)

1.4. 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)

#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=**"2 ore de seminar si 11 zi 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**

✓ 2.1. 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)

l1 = []  
l2 = []  
n = int(input("n = "))  
l1 = input("l1 = ").split()  
l2 = input("l2 = ").split()  
(l1[0::2],l2[0::2]) = (l2[0::2],l1[0::2])  
print (l1)  
print (l2)

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

ls = [int(x) for x in input().split()]  
try:  
 a = ls.index(0) # pozitia primului 0  
 b = ls.index(0,a+1)  
 del (ls[a:b+1]) # merge pana la b inclusiv  
 print (ls)  
except:  
 print ("nu se afla zerouri in lista")

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

ls = [int(x) for x in input().split()]  
while 0 in ls:  
 ls.remove(0)  
print (ls)

✓ 2.4. 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

ls = [int(x) for x in input("lista: ").split()]  
k = int(input("k: "))  
smin = sum(ls[0:k])  
p = 0  
for i in range(len(ls)-k):  
 s = sum(ls[i:i+k])  
 if s < smin:  
 smin = s  
 p = i  
del (ls[p:p+k])  
print (ls)

✓ 2.5. Se dă un vector de numere naturale ordonat crescător (toate elementele sale se vor da pe o linie separate prin spațiu). Sa se elimine duplicatele din vector.

ls = [int(x) for x in input("lista crescatoare: ").split()]  
for x in ls:  
 a = ls.index(x)  
 b = len(ls)-(ls[::-1].index(x))-1 # ultima aparitie  
 if a != b: # daca sunt egale e un element unic  
 del ls[a:b] # il pastreaza doar pe primul  
print (ls)

✓ 2.6. 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)

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

**COMPREHENSIUNE**

✓ 2.7. 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)

ls = [x for x in input("propozitie: ").split()]  
ls\_vocale = [x for x in ls if x[0] in "aeiou"]  
print(ls\_vocale)

✓ 2.8. Se citește o propoziție cu cuvintele separate prin spațiu. Să se creeze o listă cu cuvintele care conțin minim 3 vocale (folosind și comprehensiune).

ls = [x for x in input("propozitie: ").split()]  
ls\_noua = [x for x in ls if len(x) >= 3]  
print(ls\_noua)

**ȘIRURI DE CARACTERE**

✓ 2.9. Cifrul lui Cezar

a) Să se creeze în memorie o listă cu literele mici ale alfabetului și un cuvânt având ca litere toate literele mici ale alfabetului

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).  
Temă – decriptarea unui text dat pentru care se dă și cheia k, pe cazul general în care textul poate conține orice caracter, dar se codifică doar literele.

# incifrare  
litere = [chr(x) for x in range(ord('a'),ord('z') + 1)] # a,b,c,..  
cuv = "".join(litere) # abcde.. 26  
text = input("text: ")  
k = int(input("k: "))  
ls = []  
for litera in text:  
 if litera in litere:  
 litera = chr(ord(litera) + k)  
 ls.append(litera)  
text = "".join(ls)  
print (text)  
  
# descifrare  
text = input("text: ")  
k = int(input("k: "))  
ls = []  
for litera in text:  
 if litera in litere:  
 litera = chr(ord(litera) - k)  
 ls.append(litera)  
text = "".join(ls)  
print(text)

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

cuv1 = input("cuv1: ")  
cuv2 = input("cuv2: ")  
cuv1 = set(cuv1)  
cuv2 = set(cuv2)  
if cuv1 == cuv2:  
 print("anagrame")  
else:  
 print("nu")

**MATRICE**

✓ 2.12. 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.

n = int(input("n: "))  
m = int(input("m: "))  
a = []  
for i in range(n):  
 linie = []  
 for j in range(m):  
 linie.append(int(input()))  
 a.append(linie)  
for i in range(n):  
 for j in range(i+1,n):  
 if a[i][0] > a[j][0]:  
 (a[i][0],a[j][0]) = (a[j][0],a[i][0])  
for i in range(n):  
 for j in range(m):  
 print(f"{a[i][j]:2} ", end="")  
 print()

2.13. 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)

2.14. 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.

2.15. **Ciurul lui Eratostene.** Se dă un numar natural n. Să se creeze o listă cu numerele prime mai mici sau egale cu n.