`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

EXEMPLE

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

b) Se citește un text conținând separatorii uzuali(,.;:) Sa se înlocuiască toți separatorii cu spațiu + să se șteargă vocalele din text (variantă: doar să se șteargă vocalele din text).

3. Cifrul lui Cezar, păsărească – cu **`translate`**

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)

MULȚIMI + DICȚIONARE + FIȘIERE

1. Fișierul text “numere_comune.in” conține numere naturale despărțite prin spații și scrise pe mai multe linii.
   Să se afișeze pe ecran numerele care apar pe toate liniile din fișier (folosind set + intersecție). Numerele vor fi afișate crescător.
   Exemplu: dacă fișierul “numere_comune.in” conține
   2 1 50 1 3 15
   1 4 2 2 15
   2 1 1 60 8 15
   atunci fișierul “comune.out” va conține:
   1 2 15
2. Se citesc din fișierul “puncte.in” informații despre puncte în plan: cele două coordonate (numere întregi) și eticheta (care poate fi un șir ce conține și spatii). Mai exact, structura fișierului este următoarea: pe fiecare linie a fișierului sunt date informațiile despre un punct: doua coordonate întregi si eticheta punctului.

```
1 2 punctul 1
1 3 punctul 2
2 5 punctul 3
1 2 punctul 1 nou
4 1 punctul 4
```

Asupra punctelor din fișier se fac operațiile codificate in fișierul "interogari.in" astfel: pe fiecare linie se dau doua numere reprezentând coordonatele unui punct si o valoare din mulțimea {1,0} cu semnificația interogare (1)/ștergere(0).

```
1 2 1
2 4 1
1 2 0
1 3 0
4 1 1
10 20 1
20 40 0
```

Ca rezultat al operațiilor se va crea fișierul interogari.out în care în dreptul punctelor cu 1 se va scrie eticheta (sau mesajul *nu exista*), iar la final se va afișa mesajul: "punctele ramase" și apoi vor afișa punctele rămase după ștergerea punctelor cu 0 (sub forma (x,y): eticheta). Dacă un punct apare de mai multe ori în puncte.in se păstrează ultima(variante: prima/toate) etichetă asociată lui.

```
(1,2) punctul 1 nou
(2,4) nu exista
(4,1) punctul 4
(10,20) nu exista
punctele ramase
(2, 5): punctul 3
(4, 1): punctul 4
```

3. a) Scrieți o funcție care, dat 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 (minimul frecvențelor cu care apar în cele două fișiere)

   d) Să se modifice punctul c) pentru a afișa caracterele care apar în cel puțin unul dintre fișiere, cu frecvența totală.

4. Se dă un fișier cu cuvinte pe mai multe linii separate prin spații. 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**, iar *grupurile se vor scrie în fișier în ordinea descrescătoare a numărului de elemente din mulțimile literelor*.

   Pentru fișierul de intrare:

   ```
   apar mare
   si amara  rapa para
   par isi rama
   ```

   fișierul de ieșire va fi

   ```
   par apar para rapa
   rama amara
   si isi
   ```

5. Să se verifice dacă două șiruri de caractere formate doar din litere mici sunt anagrame sau nu. Două șiruri sunt anagrame dacă sunt formate din aceleași litere, dar așezate în altă ordine (sau, echivalent, unul dintre șiruri se poate obține din celălalt printr-o permutare a caracterelor sale). De exemplu, șirurile **`emerit`** și **`treime`** sunt anagrame, dar șirurile **`emerit`** și **`treimi`** nu sunt!

Temă: Dacă două șiruri s și t sunt anagrame, să se afișeze o permutare p prin care șirul t se obține din șirul s, respectiv o permutare q prin care șirul s se obține din șirul t. De exemplu, pentru șirurile s = "emerit" și t = "treime" o permutare p prin care se poate obține șirul t din șirul s este $p = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 \\ 3 & 5 & 6 & 2 & 4 & 1 \end{pmatrix}$, deoarece prima literă din șirul s (litera 'e') trebuie să fie a treia literă din șirul t, a doua literă din șirul s (litera 'm') trebuie să fie a cincea literă din șirul t, ș.a.m.d.

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

   Magazin cod_magazin nume_magazin
   cantitate nume_produs
   ...
   cantitate nume_produs
   Magazin cod_magazin nume_magazin
   cantitate nume_produs
   ...
   cantitate nume_produs

   Exemplu:

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

   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 (nume_magazin, stoc_marfa) 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)

SUPLIMENTAR

`re.split(tipar_separator, sir, maxsplit)`

- **`tipar_separator –`** expresie regulată care poate fi formată din caractere simple sau caractere speciale prin care putem crea tipare de căutare, precum :
  \D - se potrivește cu orice caracter care nu este cifră
  \d - se potrivește cu orice caracter care este cifră
  [] – delimitează o mulțime de caractere
  \+ - expresia se poate repeta, cel puțin o dată

https://docs.python.org/3/library/re.html

EXEMPLE

1. Se dă o propoziție în care cuvintele sunt separate prin unul sau mai multe semne de punctuație uzuale.

   a) Să se afișeze cuvintele din propoziție ordonate lexicografic.

   b) Se citesc două cuvinte s și t. Să se afișeze propoziția obținută înlocuind toate aparițiile lui s cu t.

2. (laborator) Jurnalul electronic al Anei conține, în fiecare zi, câte o frază cu informații despre cheltuielile pe care ea le-a efectuat în ziua respectivă. Scrieți un program care să citească o frază de acest tip din jurnalul Anei și apoi să afișeze suma totală cheltuită de ea în ziua respectivă. De exemplu, pentru fraza “Astăzi am cumpărat pâine de 5 RON, pe lapte am dat 10 RON, iar de 15 RON am cumpărat niște cașcaval. De asemenea, mi-am cumpărat și niște papuci cu 50 RON!”, programul trebuie să afișeze suma totală de 80 RON. Fraza se consideră corectă, adică toate numerele care apar în ea sunt numere naturale reprezentând sume cheltuite de Ana în ziua respectivă!