

SEMINAR 2

Șiruri de caractere

1. Să se verifice dacă un șir de caractere t apare ca subșir într-un șir s , iar în caz afirmativ să se afișeze toate pozițiile la care începe t în s . De exemplu, șirul $t = \text{"abc"}$ apare ca subșir în șirul $s = \text{"abccabcababc"}$ începând cu pozițiile 0, 4 și 9.

Rezolvare:

Vom căuta în mod repetat șirul t în șirul s folosind metoda `find(șir)`:

```
s = input("s = ")
t = input("t = ")

p = s.find(t)
if p == -1:
    print("Șirul '" + t + "' nu apare în șirul '" + s + "'!")
else:
    print("Pozitiile pe care incepe sirul '" + t +
          "' în șirul '" + s + "':")
    while p != -1:
        print(p)
        p = s.find(t, p + len(t))
```

Dacă șirul t se găsește în șirul s începând cu o poziție p , atunci căutarea sa se poate relua cu una dintre următoarele două poziții (vezi ultima linie din programul de mai sus):

- $p + \text{len}(t)$, caz în care se vor găsi aparițiile nesuprapuse ale șirului t în s (de exemplu, pentru $s = \text{"ababababa"}$ și $t = \text{"aba"}$ se vor afișa pozițiile 0 și 4);
- $p + 1$, caz în care se vor găsi aparițiile suprapuse ale șirului t în s (de exemplu, pentru $s = \text{"ababababa"}$ și $t = \text{"aba"}$ se vor afișa pozițiile 0, 2, 4 și 6).

2. Se citește un cuvânt w , un număr natural nenul p și un șir format din n cuvinte. Să se afișeze toate cuvintele care sunt p -rime cu w , respectiv ultimele p caractere ale sale coincid cu ultimele p caractere ale lui w . De exemplu, pentru $w = \text{"mere"}$, $p = 2$ și cuvintele "pere", "teste" și "programare" (decă $n = 3$), trebuie să fie afișate cuvintele "pere" și "programare".

Rezolvare:

Extragem în șirul r ultimele p caractere din cuvântul w și apoi verificăm, pentru fiecare dintre cele n cuvinte citite, dacă ultimele p caractere ale sale coincid cu șirul r :

```
w = input("Cuvântul w: ")
p = int(input("Numărul p: "))
n = int(input("Numărul de cuvinte: "))
```

```

sol = ""
rima = w[-p:]
for i in range(n):
    cuv = input("Cuvânt: ")
    if rima == cuv[-p:]:
        sol = sol + cuv + " "
sol = sol.strip()

if sol != "":
    print(str(p) + "-rimele cuvântului '" + w + "': " + sol)
else:
    print("Niciun cuvânt dat nu este " + str(p) + "-rimă " +
        "a cuvântului '" + w + "':")

```

Pentru a verifica dacă ultimele p caractere ale cuvântului `cuv` coincid cu șirul `rima` putem folosi metoda `endswith(șir)`, astfel:

```

.....
if cuv.endswith(rima):
    sol = sol + cuv + " "
.....

```

3. Se citește o propoziție sau o frază în care cuvintele sunt despărțite între ele prin spații și semnele de punctuație uzuale. Să se afișeze toate cuvintele distincte de lungime maximă din propoziția sau fraza dată. De exemplu, în propoziția "Ana are prune și gutui verzi, dar mai multe prune decât gutui!" cuvintele distincte de lungime maximă, egală cu 5, sunt: "prune", "gutui", "verzi", "multe" și "decât".

Rezolvare:

Pentru a împărți propoziția sau fraza dată în cuvinte, vom folosi metoda `split(separator)`. Deoarece această metodă poate să împartă un șir în subșiruri doar în funcție de un singur separator, mai întâi vom înlocui toate semnele de punctuație uzuale cu spații, după care vom utiliza această metodă fără parametri. Cuvintele de lungime maximă le vom păstra în șirul de caractere `rez`, astfel: în cazul în care cuvântul curent `cuv` are o lungime strict mai mare decât lungimea maximă `lmax` determinată până în acel moment, atunci înlocuim `rez` cu `cuv` și `lmax` cu lungimea cuvântului curent, iar în cazul în care cuvântul curent are lungimea egală cu `lmax` în concatenăm la șirul `rez`, încadrat între două spații. Pentru a verifica dacă un cuvânt de lungime maximă a fost deja adăugat în șirul `rez`, îl vom căuta în acesta, încadrat între două spații, pentru a evita găsirea sa ca subșir al altui cuvânt.

```

text = input("Propoziția sau fraza: ")

separatori = ",. ;?!"
for x in separatori:
    text = text.replace(x, " ")

rez = " "
lmax = 0

```

```

for cuv in text.split():
    if len(cuv) > lmax:
        rez = " " + cuv + " "
        lmax = len(cuv)
    elif len(cuv) == lmax:
        if " " + cuv + " " not in rez:
            rez += cuv + " "

rez = rez.strip()

print("Cuvintele distincte de lungime maximă:\n" +
      "\n".join(rez.split()))

```

O soluție mai eficientă se poate obține folosind în locul șirului rez o structură de date de tip mulțime.

4. Considerăm un șir de caractere reprezentând un titlu în limba engleză. Să se formateze titlul respectiv conform următoarelor reguli:

- primul și ultimul cuvânt se vor scrie întotdeauna cu majusculă;
- toate cuvintele formate din cel puțin 5 litere se vor scrie cu majusculă;
- toate cuvintele formate din cel mult 4 litere care nu fac parte dintr-o listă de cuvinte exceptate (articole, prepoziții, conjuncții etc.) se vor scrie cu majusculă.

Titlul poate să conțină și alte caractere în afară de litere mari sau mici (i.e., cifre sau semne de punctuație), iar cuvintele sunt despărțite între ele printr-un spațiu sau printr-un semn de punctuație și un spațiu. Lista cuvintelor exceptate poate conține, de exemplu, cuvintele "a", "an", "by", "on", "in", "at", "to", "for", "ago", "the", "past", "over", "into" și "onto".

Exemple:

- "where are WE GOING to?" -> "Where Are We Going To?"
- "GOING To California By My Car" -> "Going to California by My Car"
- "by my SIDE, at The SeaSide" -> "By My Side, at the Seaside"
- "walking over the rainbow" -> "Walking over the Rainbow"

Rezolvare:

Pentru formatarea unui titlu conform celor reguli de mai sus, vom identifica fiecare cuvânt al său (mai puțin primul) în funcție de spațiile care îl delimitează. Astfel, ultimul cuvânt va începe după ultimul spațiu din titlu (dacă acesta este format din cel puțin două cuvinte), iar orice alt cuvânt va fi delimitat de două spații consecutive (ale căror poziții le vom reține în două variabile p și q). Cuvintele exceptate de la cazul c) le vom păstra într-un șir de caractere, încadrate între două spații fiecare (de ce?).

```

s = input("Titlul: ")

# eliminăm spațiile de la începutul și sfârșitul șirului,
# după care transformăm toate literele în minuscule
s = s.strip().lower()

```

```

# transformăm prima literă în majusculă - cazul a)
s = s[0].upper() + s[1:]

# căutăm ultimul cuvânt din titlu (începe după ultimul spațiu)
# și îi transformăm prima literă în majusculă - cazul a)
p = s.rfind(" ")
if p != -1:
    s = s[:p+1] + s[p+1].upper() + s[p+2:]

exceptii = " a an by on in at to for ago the past over into onto "

# căutăm primul spațiu
p = s.find(" ")
while p != -1:
    # căutăm următorul spațiu
    q = s.find(" ", p + 1)
    if q != -1:
        # extragem în cuvântul curent și eliminăm eventualul semn
        # de punctuație de la sfârșitul său
        cuv = s[p + 1: q].strip(" ,. ; ? !")
        # transformăm în majusculă prima literă a cuvântului curent
        # dacă suntem într-unul dintre cazurile b) sau c)
        if len(cuv) >= 5 or (len(cuv) < 5 and
                           " " + cuv + " " not in exceptii):
            s = s[:p+1] + s[p+1].upper() + s[p+2:]
        p = q
    else:
        break

print("Titlul formatat: " + s)

```

Pentru formatare unui titlul în limba engleză sunt definite mai multe stiluri standard (APA, Chicago, NY Times etc.): <https://capitalizemytitle.com/>. Regulile de formatare prezentate în aceasta problemă sunt o variantă (simplificată) a celor mai des întâlnite reguli.

5. Se citește un șir de caractere s . Să se verifice dacă există un șir t , diferit de s , astfel încât șirul s să se poată obține prin concatenarea de un număr arbitrar de ori k a șirului t . De exemplu, pentru $s = "abcabc"$ avem $t = "abc"$ și $k = 2$.

Rezolvare:

Se observă faptul că șirul t poate fi doar un prefix al șirului s și lungimea sa trebuie să fie un divizor d al lungimii n a șirului s . De asemenea, divizorul d trebuie să fie cuprins între 1 și $\text{len}(s)//2$, deoarece șirul t trebuie să fie diferit de șirul s . În consecință, pentru fiecare divizor d al lui n , verificăm dacă prin concatenarea prefixului lui s de lungime d (i.e., subșirul $s[:d]$) cu el însuși de $n//d$ ori obținem șirul s sau nu:

```

s = input("Șirul s: ")
n = len(s)

```

```

for d in range(1, n//2 + 1):
    if n % d == 0:
        t = s[:d] * (n//d)
        if t == s:
            print("t = ", s[:d], "\nk = ", n//d)
            break
    else:
        print("Imposibil!")

```

Programul de mai sus va afișa șirul t de lungime minimă (de exemplu, pentru șirul $s = \text{"abababab"}$ va afișa $t = \text{"ab"}$ și $k = 4$), deoarece posibili divizori d ai lui n sunt considerați în ordine crescătoare. Pentru a afișa șirul t de lungime maximă (de exemplu, pentru șirul $s = \text{"abababab"}$ să se afișeze $t = \text{"abab"}$ și $k = 2$), vom considera posibili divizori d în ordine descrescătoare, respectiv vom înlocui `range(1, n//2 + 1)` cu `range(n//2, 0, -1)`.

6. Într-o propoziție, Ana a sintetizat informațiile despre cumpărăturile pe care le-a efectuat într-o anumită zi, pentru fiecare produs cumpărat ea precizând cantitatea și prețul unitar (e.g., "Astăzi am cumpărat 5 kg de mere cu 2.30 RON kilogramul și 2 pâini a câte 5 RON bucata."). Scrieți un program care să afișeze totalul cheltuielilor Anei din ziua respectivă.

Rezolvare:

Împărțim propoziția în cuvinte, considerând doar spațiul ca separator, și dacă avem un număr în cuvântul curent (i.e., dacă prin conversia cuvântului curent în număr nu apare o eroare) verificăm dacă o variabilă `cantitate` conține deja o valoare (i.e., nu conține valoarea `None`), reprezentând cantitatea dintr-un produs, sau nu. În caz afirmativ, numărul curent este păstrat în variabila `cantitate`, altfel el reprezintă un preț unitar, deci îl stocăm în variabila `pret` și adăugăm la totalul cheltuielilor valoarea produsului dintre cantitate și prețul unitar, după care setăm variabilele `cantitate` și `pret` la valoarea `None`.

```

s = input("Propozitia: ")
cuvinte = s.split(" ")

cantitate = pret = None
total = 0

for cuv in cuvinte:
    try:
        aux = float(cuv)
    except ValueError:
        continue

    if cantitate is None:
        cantitate = aux

```

```

else:
    pret = aux
    total += cantitate * pret
    cantitate = pret = None

print(f"Totalul cheltuielilor: {total}")

```

Probleme propuse

1. Se citește un cuvânt w și un șir format din n cuvinte. Să se afișeze toate cuvintele distincte care au aceeași lungime cu w . De exemplu, pentru $w = \text{"mere"}$ și cuvintele "pere", "teste" și "mure" (deci $n = 3$), trebuie să fie afișate cuvintele "pere" și "mure".
2. Se citește o propoziție sau o frază în care cuvintele sunt despărțite între ele prin spații și semnele de punctuație uzuale. Să se afișeze numărul cuvintelor distincte din propoziția sau fraza dată. De exemplu, în propoziția "Ana are prune și gutui verzi, dar mai multe prune decât gutui!" sunt 10 cuvinte distincte (cuvintele "prune" și "gutui" se iau în considerare o singură dată, deși apar de câte două ori în propoziție).
3. Să se afișeze numărul de litere mici, litere mari și semne de punctuație dintr-o propoziție citită de la tastatură.
4. Implementați cifrul lui Cezar (https://ro.wikipedia.org/wiki/Cifrul_Cezar) pentru a cripta/decripta o propoziție citită de la tastatură.