

SEMINAR 4

Funcții

1.

- a) Scrieți o funcție care returnează o matrice triunghiulară de dimensiune n , având forma următoare:
- prima coloană conține numerele $1, 2, 3, \dots, n$;
 - ultima linie conține numerele $n, n - 1, \dots, 2, 1$;
 - restul elementelor aflate în triunghiul de sub diagonala principală se calculează ca sumă a elementelor vecine de la vest, sud și sud-vest.
- b) Scrieți o funcție care afișează o matrice triunghiulară, memorată sub forma unei liste de liste, în formă matriceală, cu elementele de pe fiecare coloană aliniate la dreapta.

Exemplu: pentru $n = 4$ matricea cerută este $M = [[1], [2, 15], [3, 10, 15], [4, 3, 2, 1]]$ sau, afișată în formă matriceală:

$$M = \begin{pmatrix} 1 & & & \\ 2 & 15 & & \\ 3 & 10 & 15 & \\ 4 & 3 & 2 & 1 \end{pmatrix}$$

Rezolvare:

Vom construi matricea triunghiulară de forma cerută pas cu pas, astfel:

- vom construi matricea fără ultima linie (exemplul este pentru $n = 4$) și elementele care nu se află pe prima coloană egale cu 0:

$$M = \begin{pmatrix} 1 & & & \\ 2 & 0 & & \\ 3 & 0 & 0 & \end{pmatrix}$$

Se observă faptul că fiecare linie i , unde $i \in \{0, 1, \dots, n - 2\}$, este o listă de forma $\left[i + 1, \underbrace{0, \dots, 0}_{\text{de } i \text{ ori}} \right]$.

- vom adăuga la matricea construită anterior ultima linie, adică lista $[n, n - 1, \dots, 1]$:

$$M = \begin{pmatrix} 1 & & & \\ 2 & 0 & & \\ 3 & 0 & 0 & \\ 4 & 3 & 2 & 1 \end{pmatrix}$$

- pentru fiecare linie i , începând cu penultima ($i = n - 2$) și terminând cu a doua ($i = 1$), calculăm elementele aflate pe coloanele $\{1, 2, \dots, i\}$ ca sumă a elementelor vecine de la vest, sud și sud-vest.

Ținând cont de observațiile de mai sus, funcția cerută este următoarea:

```
def matrice(n):
    M = [[i] + [0]*(i-1) for i in range(1, n)]
    M.append([i for i in range(n, 0, -1)])

    for i in range(n-2, 0, -1):
        for j in range(1, i+1):
            M[i][j] = M[i][j-1] + M[i+1][j-1] + M[i+1][j]

    return M
```

Pentru a afișa o matrice triunghiulară, memorată sub forma unei liste de liste, în formă matriceală, cu elementele de pe fiecare coloană aliniate la dreapta, vom calcula mai întâi numărul maxim de cifre *ncmax* pe care îl are un element al matricei (de exemplu, pentru $n = 4$ vom obține $ncmax = 2$) și apoi vom afișa fiecare element al matricei aliniat la dreapta pe *ncmax* spații folosind metoda *rjust* pentru șiruri de caractere:

```
def afisare(M):
    ncmax = max([len(str(max(linie))) for linie in M])
    for linie in M:
        for elem in linie:
            print(str(elem).rjust(ncmax), end=' ')
        print()
```

Cum s-ar putea calcul mai eficient numărul maxim de cifre *ncmax* pe care îl are un element al matricei?

2. Scrieți o funcție și un generator care primesc ca parametrii un număr variabil de liste și o valoare *x* și furnizează toate listele care conțin valoarea *x*.

Rezolvare:

În cazul funcției, vom construi rezultatul sub forma unei liste *rez* care va conține listele în care apare valoarea *x*:

```
def cauta(x, *liste):
    rez = []
    for lcrt in liste:
        if x in lcrt:
            rez.append(lcrt)
    return rez

x = 7
r = cauta(x, [5, 1, 7, 3, 7], [2, 3], [-3, 7, 1])
```

```

if r == []:
    print("Valoarea", x, "nu se gaseste in nicio lista!\n")
else:
    print("Valoarea", x, "se gaseste in listele urmatoare:")
    print(*r, sep="\n")

```

În cazul generatorului, vom returna, pe rând, fiecare listă în care apare valoarea x:

```

def cauta(x, *liste):
    for lcrt in liste:
        if x in lcrt:
            yield lcrt

x = 7
r = cauta(x, [5, 1, 7, 3, 7], [2, 3], [-3, 7, 1])

lst = next(r, None)
if lst is None:
    print("Valoarea", x, "nu se gaseste in nicio lista!\n")
else:
    print("Valoarea", x, "se gaseste in listele urmatoare:")
    while lst is not None:
        print(lst)
        lst = next(r, None)

```

Observați modul în care am verificat faptul că generatorul nu a returnat nicio listă!

3.

- a) Pentru un student se cunosc următoarele informații: numele, grupa și o listă cu creditele obținute la toate examenele din anul respectiv. Considerând o listă de studenți, scrieți o funcție care să adauge la fiecare student situația sa școlară: promovat (True) sau nepromovat (False). Pentru a fi considerat promovat, un student trebuie să nu aibă nici un examen nepromovat (adică o valoare egală cu 0 în lista creditelor), iar suma creditelor obținute să fie mai mare sau egală decât un număr minim de credite dat.
- b) Scrieți câte o funcție care să furnizeze cheia necesară sortării listei cu informații despre studenți în raport de fiecare dintre următoarele criterii:
 - crescător după grupă și în fiecare grupă în ordine alfabetică;
 - întâi studenții promovați, apoi cei nepromovați și în fiecare categorie în ordine alfabetică;
 - descrescător după suma creditelor, iar în cazul unor sume egale în ordinea crescătoare a grupei și în ordine alfabetică în cadrul grupei;
 - în ordinea crescătoare a grupelor, în cadrul fiecărei grupe mai întâi studenții promovați, iar apoi cei nepromovați, în fiecare categorie (promovat/nepromovat) în ordinea descrescătoare a sumei creditelor și, în cazul unor sume egale, în ordine alfabetică.

Rezolvare:

Pentru testarea celor două funcții cerute, în continuare vom considera următoare listă cu informații despre studenți:

```
lstud = [("Popescu Ion", 135, (5, 7, 3, 0, 4)),
         ("Popa Anca", 131, (5, 7, 3)),
         ("Mihai Ana", 135, (7, 3, 3, 2, 4)),
         ("Mihai Ana", 132, (7, 0, 3, 0)),
         ("Ionescu Clara", 131, (5, 5, 3)),
         ("Bobescu Iulia", 135, (5, 5, 5, 3, 3))]
```

Pentru a calcula situația școlară a unui student, vom parcurge lista element cu element și vom verifica dacă studentul respectiv îndeplinește condițiile de promovare sau nu:

```
# ls = lista cu studenți
# minc = numărul minim de credite necesare
def situatie_scolara(ls, minc):
    for i in range(len(ls)):
        if 0 not in ls[i][2] and sum(ls[i][2]) >= minc:
            ls[i] += (True,)
        else:
            ls[i] += (False,)
```

Dacă vom testa funcția prin

```
situatie_scolara(lstud, 15)
print(*lstud, sep="\n")
```

se vor afișa următoarele informații:

```
('Popescu Ion', 135, (5, 7, 3, 0, 4), False)
('Popa Anca', 131, (5, 7, 3), True)
('Mihai Ana', 135, (7, 3, 3, 2, 4), True)
('Mihai Ana', 132, (7, 0, 3, 0), False)
('Ionescu Clara', 131, (5, 5, 3), False)
('Bobescu Iulia', 135, (5, 5, 5, 3, 3), True)
```

Atenție, următoare funcție este incorectă, respectiv nu va actualiza informațiile despre un student prin adăugarea la tuplul respectiv a valorii True sau False:

```
def situatie_scolara(ls, minc):
    for student in ls:
        if 0 not in student[2] and sum(student[2]) >= minc:
            student += (True,)
        else:
            student += (False,)
```

Prin parcurgerea unei colecții folosind instrucțiunea for nu se accesează direct un element al colecției, ci o copie a elementului respectiv (în acest caz, variabila student)!

Pentru rezolvarea celei de-a doua cerințe, vom indica, în fiecare caz, funcția care va furniza cheia corespunzătoare și rezultatul sortării listei obținute după apelul `situatie_scolara(lstud, 15)` folosind cheia respectivă:

- crescător după grupă și în fiecare grupă în ordine alfabetică:

Funcția pentru cheie	Rezultatul sortării
<pre>def criteriu_1(t): return t[1], t[0] lstud.sort(key=criteriu_1) print(*lstud, sep="\n")</pre>	<pre>('Ionescu Clara', 131, (5, 5, 3), False) ('Popa Anca', 131, (5, 7, 3), True) ('Mihai Ana', 132, (7, 0, 3, 0), False) ('Bobescu Iulia', 135, (5, 5, 5, 3, 3), True) ('Mihai Ana', 135, (7, 3, 3, 2, 4), True) ('Popescu Ion', 135, (5, 7, 3, 0, 4), False)</pre>

- întâi studenții promovați, apoi cei nepromovați și în fiecare categorie în ordine alfabetică:

Funcția pentru cheie	Rezultatul sortării
<pre>def criteriu_2(t): return -t[3], t[0] lstud.sort(key=criteriu_2) print(*lstud, sep="\n")</pre>	<pre>('Bobescu Iulia', 135, (5, 5, 5, 3, 3), True) ('Mihai Ana', 135, (7, 3, 3, 2, 4), True) ('Popa Anca', 131, (5, 7, 3), True) ('Ionescu Clara', 131, (5, 5, 3), False) ('Mihai Ana', 132, (7, 0, 3, 0), False) ('Popescu Ion', 135, (5, 7, 3, 0, 4), False)</pre>

Observați faptul că am sortat descrescător după situația școlară, deoarece valoarea True este echivalentă cu 1, iar valoarea False cu 0!

- descrescător după suma creditelor, iar în cazul unor sume egale în ordinea crescătoare a grupei și în ordine alfabetică în cadrul grupei:

Funcția pentru cheie	Rezultatul sortării
<pre>def criteriu_3(t): return -sum(t[2]), \ t[1], t[0] lstud.sort(key=criteriu_3) print(*lstud, sep="\n")</pre>	<pre>('Bobescu Iulia', 135, (5, 5, 5, 3, 3), True) ('Mihai Ana', 135, (7, 3, 3, 2, 4), True) ('Popescu Ion', 135, (5, 7, 3, 0, 4), False) ('Popa Anca', 131, (5, 7, 3), True) ('Ionescu Clara', 131, (5, 5, 3), False) ('Mihai Ana', 132, (7, 0, 3, 0), False)</pre>

- în ordinea crescătoare a grupelor, în cadrul fiecărei grupe mai întâi studenții promovați, iar apoi cei nepromovați, în fiecare categorie (promovat/nepromovat) în ordinea descrescătoare a sumei creditelor și, în cazul unor sume egale, în ordine alfabetică:

Funcția pentru cheie	Rezultatul sortării
<pre>def criteriu_4(t): return t[1], -t[3], \ -sum(t[2]), t[0] lstud.sort(key=criteriu_4) print(*lstud, sep="\n")</pre>	<pre>('Popa Anca', 131, (5, 7, 3), True) ('Ionescu Clara', 131, (5, 5, 3), False) ('Mihai Ana', 132, (7, 0, 3, 0), False) ('Bobescu Iulia', 135, (5, 5, 5, 3, 3), True) ('Mihai Ana', 135, (7, 3, 3, 2, 4), True) ('Popescu Ion', 135, (5, 7, 3, 0, 4), False)</pre>

Observați faptul că în fiecare cheie (de fapt, un tuplu) am precizat componentele în ordinea cerută de criteriile de sortare, deoarece compararea a două tupluri se realizează lexicografic. De asemenea, o sortare descrescătoare în raport de o componentă numerică a cheii se realizează prin schimbarea semnului său.

4.

- a) Scrieți o funcție generică care să furnizeze numărul elementelor dintr-o colecție iterabilă care au o anumită proprietate, implementată sub forma unei funcții cu un singur parametru care returnează True dacă acesta are proprietatea cerută sau False în caz contrar.
- b) Scrieți câte o funcție care să implementeze proprietatea necesară pentru ca funcția de numărare să furnizeze:
- numărul valorilor pare dintr-o listă/tuplu de numere întregi;
 - numărul vocalelor dintr-un șir de caractere;
 - numărul perechilor (x, y) cu proprietatea că $x = y$ dintr-o listă de tuple;
 - numărul șirurilor de lungime k dintr-o listă de șiruri de caractere;
 - numărul valorilor x dintr-o listă de numere întregi pentru care $\text{cmmdc}(x, y) = t$, unde y și t sunt date.

Rezolvare:

Funcția generică de numărare implementează algoritmul standard de numărare, respectiv se incrementează un contor dacă elementul curent al colecției verifică proprietatea cerută:

```
def numarare(colectie, proprietate):
    contor = 0
    for element in colectie:
        if proprietate(element) == True:
            contor = contor + 1
    return contor
```

Pentru rezolvarea celei de-a doua cerințe, vom indica, în fiecare caz, funcția care va implementa proprietatea cerută și un program de test:

- numărul valorilor pare dintr-o listă de numere întregi:

```
def esteNrPar(numar):
    return numar % 2 == 0

L = [15, 21, 700, 132, -8, 19, -10, 1, 10, 5, 133]

np = numarare(L, esteNrPar)
print("Numar valori pare:", np)          # Numar valori pare: 5
```

- numărul vocalelor dintr-un șir de caractere:

```
def esteVocala(litera):
    return litera in "aeiouAEIOU"

cuvant = "Exemplificare"
nv = numarare(cuvant, esteVocala)
print("Numar vocale:", nv)              # Numar vocale: 6
```

- numărul perechilor (x, y) cu proprietatea că $x = y$ dintr-o listă de tuple:

```
def areComponenteEgale(pereche):
    return pereche[0] == pereche[1]

n = 5
# L = {1, 2, ..., n} X {1, 2, ..., n}
L = [(i, j) for i in range(1, n+1) for j in range(1, n+1)]
print(L)

np = numarare(L, areComponenteEgale)
print("Numar perechi cu componente egale:", np)
```

Pentru un număr n dat, ce valoare va fi afișată întotdeauna în exemplul de mai sus?

- numărul șirurilor de lungime k dintr-o listă de șiruri de caractere, unde k este dat:

```
def verificareLungimeSir(k):
    def aux(sir):
        return len(sir) == k
    return aux

L = "Ana are mere si pere mai multe mere decat pere".split()
lmax = max([len(cuv) for cuv in L])
print(L)

for k in range(1, lmax+1):
    ncuv = numarare(L, verificareLungimeSir(k))
    print("Numarul de cuvinte cu lungimea", k, "este", ncuv)
```

Deoarece funcția care implementează proprietatea pe care trebuie să o îndeplinească un element al colecției pentru a fi numărat trebuie să aibă un singur parametru (i.e., elementul respectiv), nu putem să utilizăm următoarea funcție "naturală":

```
def verificareLungimeSir(sir, k):
    return len(sir) == k
```

Totuși, deoarece se va folosi aceeași valoare k pentru toate șirurile din listă în momentul în care vom apela funcția `numarare`, am utilizat o funcție imbricată în care am testat condiția `len(sir) == k`, eliminând astfel parametrul k !

- numărul valorilor x dintr-o listă de numere întregi pentru care $\text{cmmdc}(x, y) = t$, unde y și t sunt date:

```
def verificareCMMDC(y, t):
    def cmmdc(a, b):
        if b == 0:
            return a
        return cmmdc(b, a % b)
```

```
def aux(x):
    return cmmdc(x, y) == t

return aux

L = [10, 142, 24, 17, 18, 100, 13, 15]
print(numarare(L, verificareCMMDC(4, 2))) # 3
```

În exemplul de mai sus se va afișa valoarea 3, deoarece $y = 4$ și $t = 2$, iar în lista L sunt 3 valori x pentru care $\text{cmmdc}(x, 4) = 2$, respectiv 10, 142 și 18.

Probleme propuse

1. Scrieți o funcție care returnează o matrice pătratică de dimensiune n , având forma următoare:

- pe ultima linie și ultima coloană toate elementele sunt egale cu 1;
- restul elementelor se calculează ca sumă a elementelor vecine de la est și sud.

Exemplu: pentru $n = 4$ matricea M cerută este:

$$M = \begin{pmatrix} 10 & 10 & 4 & 1 \\ 10 & 6 & 3 & 1 \\ 4 & 3 & 2 & 1 \\ 1 & 1 & 1 & 1 \end{pmatrix}$$

2.

a) Scrieți o funcție generică care să furnizeze o listă cu pozițiile pe care apar într-o colecție iterabilă elementele care au o anumită proprietate, implementată sub forma unei funcții cu un singur parametru care returnează `True` dacă acesta are proprietatea cerută sau `False` în caz contrar.

b) Scrieți câte o funcție care să implementeze proprietatea necesară pentru ca funcția generică să furnizeze:

- pozițiile valorilor strict pozitive dintr-un tuplu de numere întregi;
- pozițiile semnelor de punctuație dintr-un șir de caractere;
- pozițiile cuvintelor care sunt anagrame ale unui cuvânt s dintr-o listă de cuvinte;
- pozițiile numerelor naturale x dintr-o listă care sunt formate din n cifre și au suma cifrelor egală cu s .

3.

a) Pentru o carte se cunosc următoarele informații: titlul, autorii, anul apariției și prețul. Considerând o listă de cărți, scrieți o funcție care să ieftinească cu 20% cărțile apărute înainte de anul 2000.

b) Scrieți câte o funcție care să furnizeze cheia necesară sortării listei de cărți în raport de fiecare dintre următoarele criterii:

- descrescător după anul apariției și în fiecare an în ordinea alfabetică a titlurilor;
- în ordinea crescătoare a numărului de autori și în fiecare categorie în ordinea descrescătoare a prețurilor;
- în ordinea alfabetică a numelui primului autor (se consideră faptul că orice autor este dat printr-un șir de forma "Prenume Nume"), apoi în ordinea alfabetică a prenumelui primului autor, apoi în ordinea alfabetică a titlului și, respectiv, în ordinea crescătoare a anului apariției.

4. Scrieți o funcție și un generator care primesc ca parametrii un număr variabil de șiruri de caractere și un număr natural nenul n și furnizează toate șirurile care au lungimea n .