

16. TÖBBDIMENZIÓS LISTA – MÁTRIX, TÖMB

Ahogy az előző fejeztben is említettük a listánk egyik eleme lehet egy másik lista. Sőt egy lista minden eleme is lehet (belső) lista. Ilyenkor beszélünk többdimenziós listásól. Tehát a „mátrix” lista első (0.) eleme: =['A','B','C'] és annak az első (0.) eleme: 'A'

```
matrix=[['A','B','C'],['D','E','F'],['G','H','I']]
matrix=[['A','B','C',
         ['D','E','F'],
         ['G','H','I']]
```

Mindig a nagyobb (külső) listából indulunk a kisebb (belső) lista felé!
A többdimenziós listákat mindig téglalapként képzeljük el, írjuk fel! (második példa)



Az elemek téglalapszerű elrendezését kétdimenziós listának, vagy mátrixnak nevezzük.

Az első index a sort (sorindex), a második az oszlopot (oszlopindex) jelöli.

Azt nem szabad elfelejteni, hogy itt is az index értéke a nullától indul.

0,0	0,1	0,2
1,0	1,1	1,2
2,0	2,1	2,2

A	B	C
D	E	F
G	H	I

Tehát a `matrix[1][2]` → F

Nem kell azonos sor és oszlopszámúnak lennie a tömböknek!
pl. n=5, m=6 (5*6)

[0][0]	[0][1]	[0][2]	[0][3]	[0][4]	[0][5]
[1][0]	[1][1]	[1][2]	[1][3]	[1][4]	[1][5]
[2][0]	[2][1]	[2][2]	[2][3]	[2][4]	[2][5]
[3][0]	[3][1]	[3][2]	[3][3]	[3][4]	[3][5]
[4][0]	[4][1]	[4][2]	[4][3]	[4][4]	[4][5]

Mondatszerű leírásban a tömbök megadása pl.: `matrix(2,4)`

Pythonban az előbb említett módon pl.: `matrix[2][4]`

Többdimenziós tömbök készítésénél fokozottan figyeljünk a négyzetes zárójelek „[]” és a vesszők”,” helyes használatára!

(16a.py)

Készítsünk programot, melyben megadunk és kiíratunk egy 3*3-as tömböt! A feladatban egy „amőba” játék tábláját jelenítsük meg! A táblában az üres helyet „_” karakter, a másik kettőt pedig a „X” és „O” karakterek jelenítsék meg!

```
1  tabla=[['_','X','O'],
2         ['X','_','O'],
3         ['_','O','X']]
4  for i in range(0,3):
5      for j in range(0,3):
6          print(tabla[i][j],end=" ")
7      print()
```

- Az első három sorban megadjuk a tábla elemeit. Sorokba tördelve megfelelő struktúrával! Az elemek megfelelő elválasztására figyeljünk!
- Majd indítunk egy külső ciklust, melyben a sorokat fogjuk váltani. (0,1,2 sorok)
- Aztán egy belső ciklust indítunk, amelyben a sorokban lévő elemeit írjuk ki a minta szerint. A karaktereket szóközzel választjuk el egymástól!
- Majd az utolsó sorban a külső ciklus sortörése miatt egy `print()` utasítást gépelünk be. Új sort kezdünk.
- Futtassuk és teszteljük a programunkat!



```
Run: 16a x
E:\00_MM\12_Python_pr
_ X O
X _ O
_ O X
Process finished with
```

(16b.py)

Készítsünk programot, amely az előző példához hasonlóan egy fix tömböt kiírat a képernyőre! Ebben a példában egy órarendet szeretnénk kiíratni a képernyőre, soronként soron belül tabulátorokkal elválasztva.

```
1  orarend=[['Óra/Nap','Hét.','Ked.','Szer.','Csüt.','Pént.','Szom.','Vas.'],
2           ['1. ','-----'],
3           ['2. ','-----'],
4           ['3. ','-----'],
5           ['4. ','-----'],
6           ['5. ','-----'],
7           ['6. ','-----'],
8           ['7. ','-----']]
9  for i in range(0,8):
10     for j in range(0,8):
11         print(orarend[i][j],end="\t")
12     print()
```

- A program elején megadjuk egy 8*8-as tömbben az első sorban a napokat, majd a sorszámokat és az órák helyét 5db „-„ karakterrel.
- Majd egymásba ágyazott ciklussal 0-7-ig kiíratjuk a tömb elemeit tabulátorokkal elválasztva, majd sort váltunk.

```
Run: 16b x
C:\Users\eloado\PycharmProjects\pythonProject\venv\Scripts\pyt
Óra/Nap Hét. Ked. Szer. Csüt. Pént. Szom. Vas.
1. -----
2. -----
3. -----
4. -----
5. -----
6. -----
7. -----
```

(16c.py)

Készítsünk programot, melyben egy kétdimenziós tömbben szorzótáblát készítünk, majd kiíratjuk a képernyőre!

- A program első sorában létrehozunk egy üres tömböt, szorzótábla néven.
- Majd (2-6. sor) feltöltjük tömbünket. Egymásbaágyazott külső for ciklussal indulunk. Az „i” változóval a sorokban lépkedünk, ezért minden sor elején kell egy „belső” tömböt létrehozni, melyben a „j” változó „belső” for ciklussal az „oszlopok” elemeit töltjük fel az ötödik sorban. Mivel a lista leme nullával indul, ezért hozzá kell adni egyet a változók aktuális értékéhez.
- Aztán minden feltöltött sor[] tömböt hozzáadunk a „szorzotabla” tömbhöz (6. sor).
- Végül tabulátorokkal elválasztva, egymásbaágyazott for ciklussal kiíratjuk a képernyőre a szorzótáblát.

```
16c.py x
1 szorzotabla=[]
2 for i in range(0,10):
3     sor=[]
4     for j in range(0,10):
5         sor.append((i+1)*(j+1))
6     szorzotabla.append(sor)
7 for i in range(0,10):
8     for j in range(0,10):
9         print(szorzotabla[i][j], end="\t")
10    print()
```



```
Run: 16c x
C:\Users\kolmank\AppData\Local\Programs\
1 2 3 4 5 6 7 8 9 10
2 4 6 8 10 12 14 16 18 20
3 6 9 12 15 18 21 24 27 30
4 8 12 16 20 24 28 32 36 40
5 10 15 20 25 30 35 40 45 50
6 12 18 24 30 36 42 48 54 60
7 14 21 28 35 42 49 56 63 70
8 16 24 32 40 48 56 64 72 80
9 18 27 36 45 54 63 72 81 90
10 20 30 40 50 60 70 80 90 100
```

(16d.py)

Készítsünk programot, melyben feltöltünk egy 5*12-es tömböt 1-99 közötti véletlen számokkal, majd kiíratjuk a tömb legkisebb és legnagyobb elemét a minta alapján!

Tulajdonképpen az előző programot kell átírni és kibővíteni.

```
16d.py x
1 from random import *
2 tabla = []
3 max=0
4 min=100
5 for i in range(0,5):
6     sor = []
7     for j in range(0,12):
8         sor.append(randrange(1,100))
9     tabla.append(sor)
10 for i in range(0,5):
11     for j in range(0,12):
12         if tabla[i][j] > max:
13             max = tabla[i][j]
14         if tabla[i][j] < min:
15             min = tabla[i][j]
16     print(tabla[i][j], end="\t")
17     print()
18 print("A tömb legkisebb eleme: ",min)
19 print("A tömb legnagyobb eleme: ",max)
```

```
Run: 16d x
72 35 76 32 37 27 95 96 91 27 67 98
78 50 22 68 25 3 69 92 30 58 2 43
11 17 65 10 52 49 7 87 4 43 65 18
68 90 12 45 90 69 22 58 5 43 47 91
40 31 81 45 17 55 59 54 44 53 14 88
A tömb legkisebb eleme: 2
A tömb legnagyobb eleme: 98
```