El?z? gyakorlat - Fel - Következ? gyakorlat

Tartalomjegyzék

Mátrixozás még

Emlékeztet? és még pár dolog

Mátrixot megadhatunk a következ? módon:

```
m = matrix([[1, 0], [0, 1]])
```

Ez a következ? mátrixot eredményezi:

```
1 0
0 1
```

Blokkmátrixot, csupa 1-es mátrixot, továbbá f?átlóval adott mátrixot is kényelmesen adhatunk meg:

```
A = diagonal_matrix([1, 5])
B = ones_matrix(2, 2)
block_matrix([[A, -1*A], [A^(-1), B]])
```

Ez a következ? mátrixot eredményezi:

```
 1   0| -1   0
 0   5|  0  -5
------+-------
 1   0|  1   1
 0 1/5|  1   1
```

Egy mátrix determinánsát kiszámolhatjuk a **det** metódussal:

```
m.det()
```

Feladatok

Blokkmátrix

Számoljuk ki a determinánsát a következ? blokkmátrixnak:

```
X I
O X
```

ahol I a 3x3-as egységmátrix és O a 3x3-as csupa 0 mátrix, X pedig a következ?:

```
 0 -1 -1
-1  0 -1
-1 -1  0
```

Egyenlet megoldás

Oldjuk meg az Ax = b alakú egyenletrendszert, ahol A és b rendre:

```
 1 -1  0  |  1
 3  1 -1  |  1
-2  0  1  |  2
```

Használjuk az el?adáson tanult **solve_right** metódust!

Ha megkaptuk az eredményt, akkor állítsuk át a mátrixot, hogy GF(3) felett legyen értelmezve (a **change_ring** metódussal) és nézzük meg így is a megoldást.

Összefügg?

Határozzuk meg, hogy az alábbi mátrix sorai (vagy oszlopai) milyen **x** értékekre lesznek összefügg?k. (Használjuk a **solve** parancsot a fentiekkel együtt.)

```
x  0  1
0  2  x
1  x -1
```

Listaértelmezések

Emlékeztet?

```
[kifejezés for elem in bejárható_objektum]
```

Egy olyan listát hoz létre melyben a **kifejezés** szerepel a **bejárható_objektum** minden elemére. Bejárható objektum például egy lista, az is amit a **range** függvény hoz létre.

```
[kifejezés if feltétel else kifejezés_alt for elem in bejárható_objektum]
```

Mint az elõzõ, de csak azok az elemek lesznek benne melyekre teljesül a **feltétel**.

```
[kifejezés if feltétel1 else kifejezés_alt for elem1 in bejárható_objektum1
           for elem2 in bejárható_objektum2
           for elemN in bejárható_objektumN]
```

Több feltétel és ciklus is írható akár.

Pl:

```
[n^2 for n in range(1, 5)]  # [1, 4, 9, 16]
[n for n in [-1, 2, -3, 4] if n > 0]  # [2, 4]
```

Feladatok

Mit csinál?

Futtassuk le az alábbi példákat és értelmezzük ?ket mi is történik bennük és hogyan érjük ezt el.

```
[n for n in range(1, 10)]

[(n, m) for n in range(1, 10) for m in range(1, 5)]

[n for n in range(1, 10) if is_prime(n)]

[n for n in range(1, 100) if n % 5 == 0 and n % 7 == 1]

[(n, m) for n in range(1, 5) for m in range(n, 5)]

[(m, n) for n in range(1, 10) for m in range(n, 10) if m % n == 0]

sorted([(m, n) for n in range(1, 10) for m in range(n, 10) if m % n == 0])

sum([n for n in range(1, 10) if is_prime(n)])
```

Az utolsóhoz egy kis spoiler, ha nem menne: spoiler

```
[n for n in range(1, 100) if n == sum([m for m in range(1, n) if n % m == 0])]
```

Oldjuk meg

1. Keressük meg az összes olyan 1000 alatti négyzetszámot, melynél eggyel nagyobb szám prím. Pl a 4 ilyen.
2. Keressük meg az összes olyan 100 alatti számpárt, melyekre igaz, hogy mindkett? prím és az egész osztással vett eredményük is prím. Pl (11, 2) ilyen.
3. Keressük meg az összes egy jegy? számhármast, mely egymás után írva megegyezik a köbeik összegével. Ilyen például az 1, 5, 3, mert 1^3 + 5^3 + 3^3 == 153
4. Keressük meg az összes olyan 1000 alatti számot, melynek négyzete megegyezik az nálánál kisebb osztói köbeinek az összegével. (Egy kis csavar a tökéletes számokon)
5. Keressük meg az összes olyan 10000 alatti számot, mely legalább kétféleképpen írható fel 2 darab szám köbének összegeként.

- Adjuk meg az **cos(x)sin(x)x^2** függvény els? 10 deriváltját (beleértve a 0. deriváltat) a 0-ban egy 11 hosszú listaként.

- Ábrázoljuk ezt a függvényt 4 deriváltjával együtt a **[-2pi, 2pi]** intervallumon!

El?z? gyakorlat - Fel - Következ? gyakorlat