Tartalomjegyzék

Mátrixok

Blokkmátrix

Számoljuk ki a determinánsát a következõ blokkmátrixnak:

```
X I
O X
```

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

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

Megoldás

```
O = 0 * ones_matrix(3, 3)
I = diagonal_matrix([1, 1, 1])
X = -1 * ones_matrix(3, 3) + I
block_matrix([[X, I], [O, X]]).det()
```

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 elõadáson 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.

Megoldás

```
A = matrix([[1, -1, 0], [3, 1, -1], [-2, 0, 1]])
b = vector([1, 1, 2])
A.solve_right(b)
```

```
A = matrix([[1, -1, 0], [3, 1, -1], [-2, 0, 1]])
b = vector([1, 1, 2])
G = A.change_ring(GF(3))
G.solve_right(b)
```

Összefüggõ

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

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

Megoldás

```
m = matrix([[x, 0, 1], [0, 2, x], [1, x, -1]])
solve(m.det() == 0, x)
```

Listaértelmezések

Mit csinál?

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

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

Egész számok 1-tõl 9-ig.

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

Összes lehetséges számpár, ahol az elsõ szám 1-tõl 9-ig terjed, a második 1-tõl 4-ig.

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

Prímek 1 és 9 között.

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

1 és 99 között az 5-el osztható és 7-el 1 maradékot adó számok.

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

Számpárok ahol az elsõ szám 1-tõl 4-ig terjed, a második pedig az elsõ számtól 4-ig. (Így a második mindig legalább akkora lesz, mint az elsõ.)

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

Számpárok 1-tõl 9-ig, melyekben az elsõ elem a második osztója.

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

Az elõzõ, csak lexikografikusan rendezve.

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

1-tõl 9-ig a prímek összege.

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])]
```

Összetett számok...duh.

Oldjuk meg

- Keressük meg az összes olyan 1000 alatti négyzetszámot, melynél eggyel nagyobb szám prím. Pl a 4 ilyen.

```
[n^2 for n in range(1,sqrt(1000)) if is_prime(n^2 + 1)]
```

- Keressük meg az összes olyan 100 alatti számpárt, melyekre igaz, hogy mindkettõ prím és az egészosztással vett eredményük is prím. Pl (11, 2) ilyen.

```
[(n, m) for n in range(1, 100) for m in range(1, 100) if is_prime(n) and is_prime(m) and is_prime(
```

- Keressük meg az összes egy jegyû számhármast, mely egymás után írva megegyezik a köbeik összegével. Ilyen például az 1, 5, 3, mert 1^3 + 5^3 + 3^3 == 153

```
[(n, m, k) for n in range(1, 10) for m in range(1, 10) for k in range(1, 10) if 100*n + 10*m + k =
```

- Keressük meg az összes olyan 1000 alatti számot, melynek négyzete megegyezik az nálánál kisebb osztói köbeinek az összegével. (Egy kis csavar a tökéletes számokon)

```
[n for n in range(1, 1000) if n^2 == sum([m^3 for m in range(1, n) if n % m == 0])]
```

- Keressük meg az összes olyan 10000 alatti számot, mely kétféleképpen írható fel 2 darab szám köbének összegeként.

```
[((n, m, n^3 + m^3), (k, l, k^3 + l^3)) for n in range(1, 10000^(1/3)) for m in range(n + 1, 10000
```