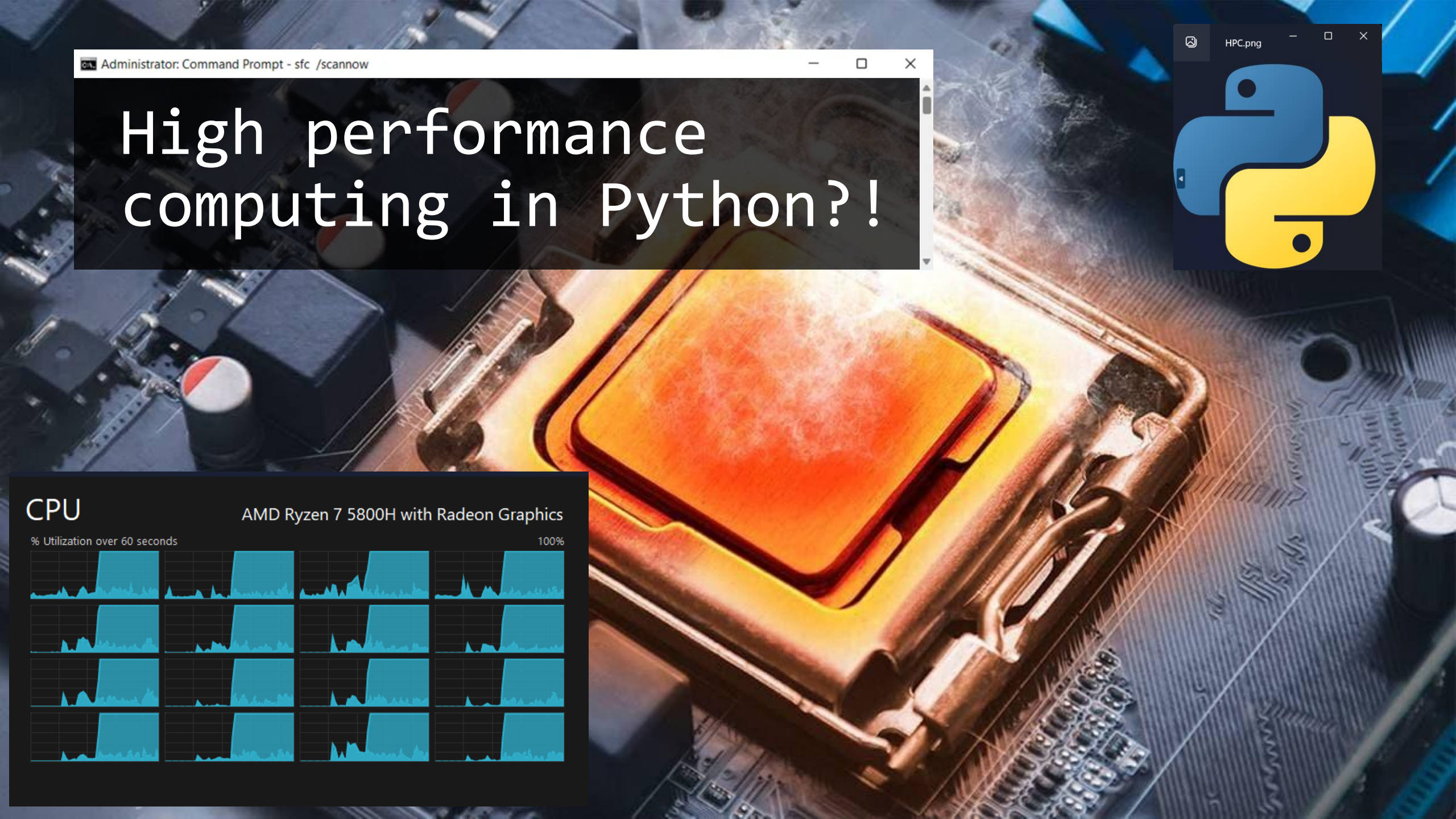

Administrator: Command Prompt - sfc /scannow
High performance computing in Python?!
HPC.png
CPU
AMD Ryzen 7 5800H with Radeon Graphics
% Utilization over 60 seconds
100%

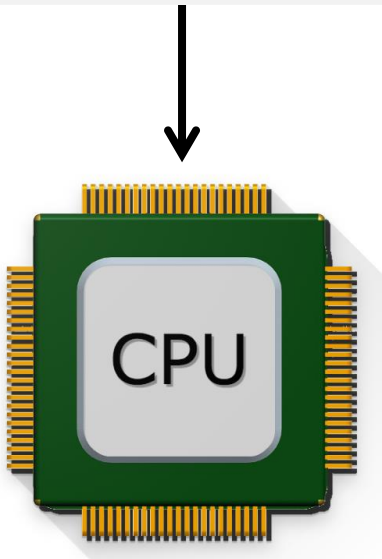
Miért lassú a Python?

Fordított nyelvek

Interpretált nyelvek

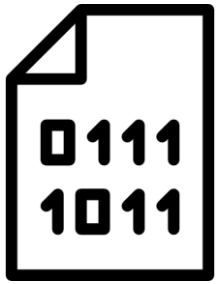
Fordított nyelvek

```
        .global  _start
_start:
        MOV R7, #4
        MOV R0, #1
        MOV R2, #14
        LDR R1,=string
        SWI 0
        MOV R7, #1
        SWI 0
        .data
string:
        .ascii "Hello, World!\n"
```

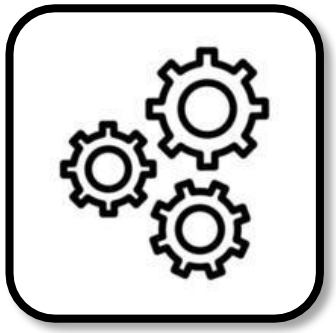


Interpretált nyelvek

Forráskód
example.py



Bytekód



Interpreter
Python.exe

Fordított nyelvek

```
double add(double a)
{
    return (a + a);
}
```

```
add(5);
```

```
10.0
```

```
add("five");
```

```
Error
```

Interpretált nyelvek

```
def add(a):
    return (a + a)
```

```
add(5)
```

```
10
```

```
add(5.0)
```

```
10.0
```

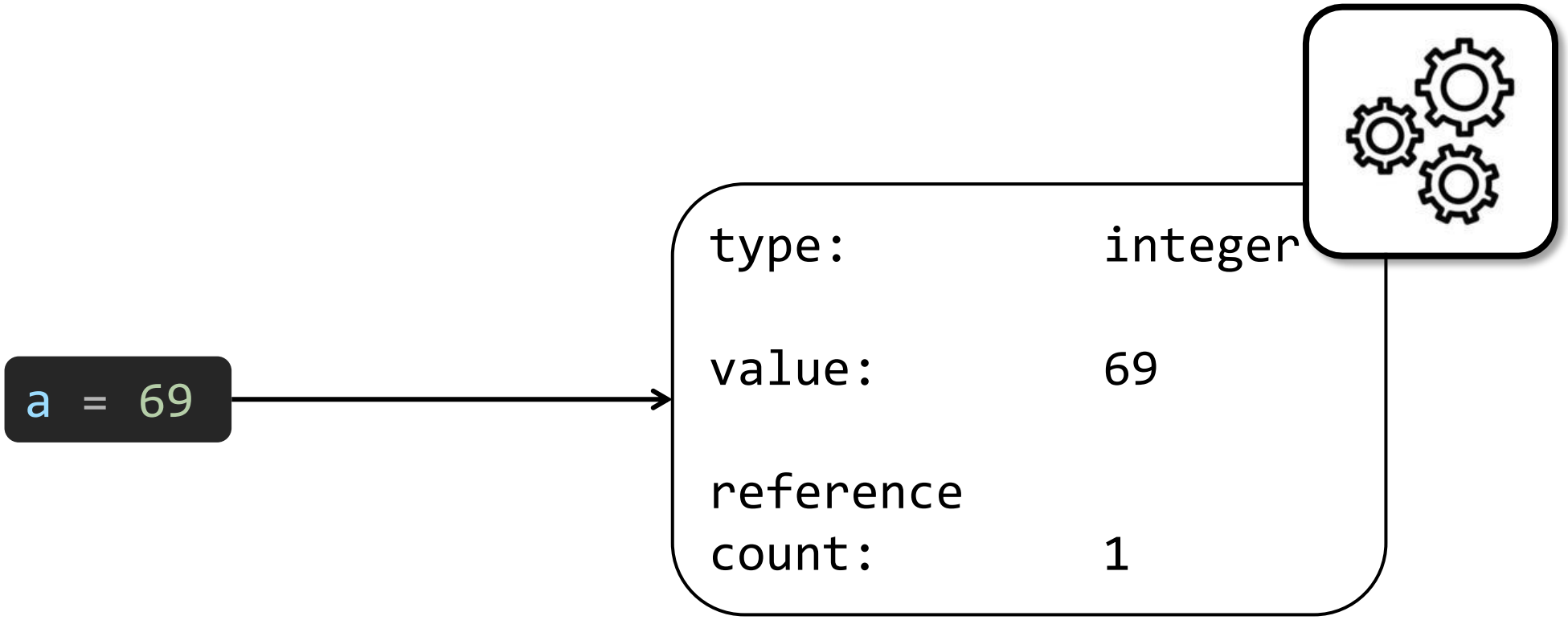
```
add('five')
```

```
’fivefive’
```

```
add([5])
```

```
[5, 5]
```

Interpretált nyelvek



Interpretált nyelvek

```
a = [69]          # ref = 1
b = a             # ref = 2
b.append(42)
a
```

```
[69, 42]
```

```
a = [69]                    # ref = 1

def f(b):                   # ref = 2
    b.append(42)
    return 0                # ref = 1

f(a)
```

```
a
```

```
[69, 42]
```

```
b
```

```
NameError: name 'b' is not defined
```

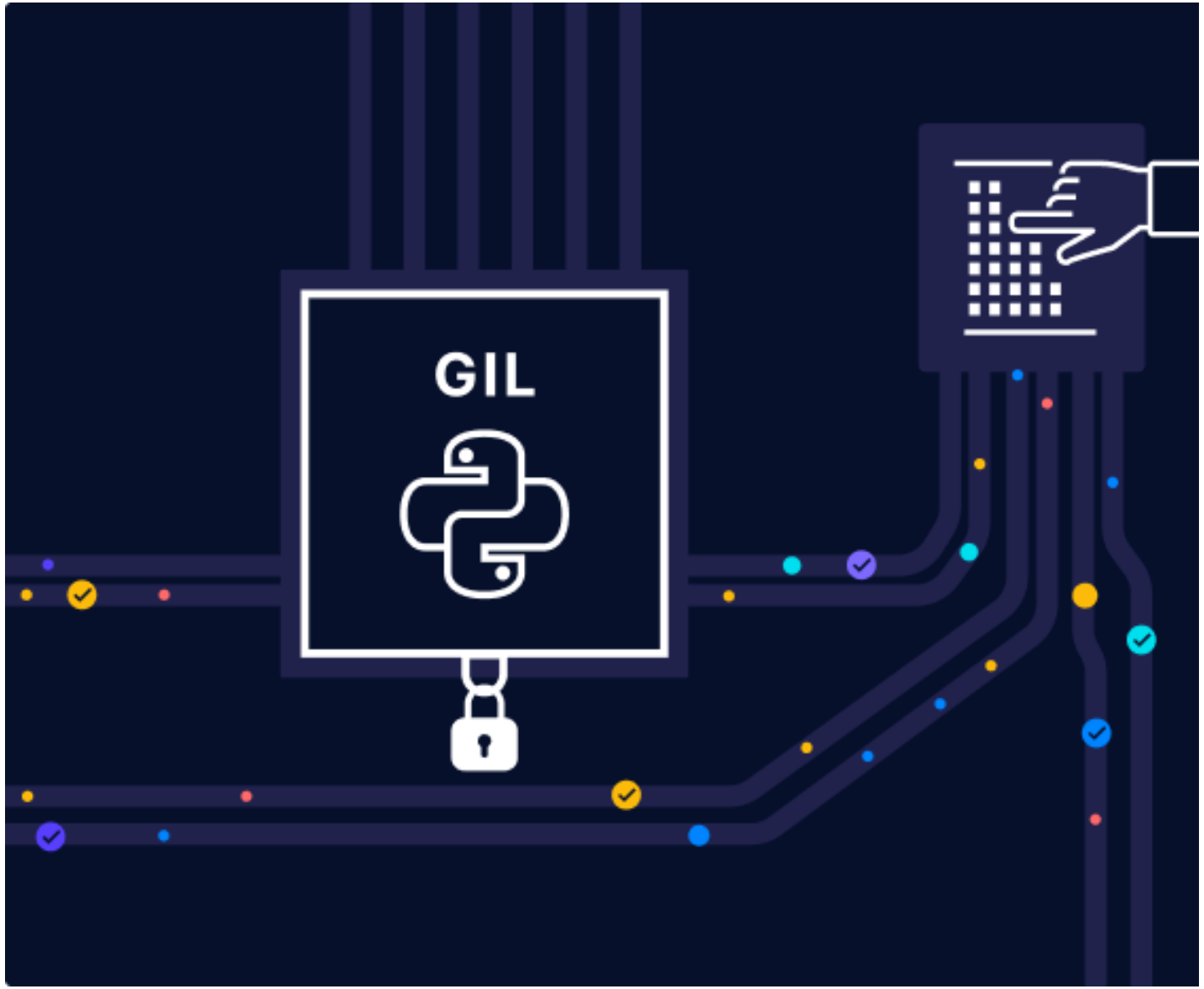
Garbage collector

```
reference_count = 0
```

```
int* arr = new int[42];

/* ... */

delete[] arr;
```

Global Interpreter Lock

- A Python alapvetően 1 szálon fut!



Mit lehet tenni?

Gépi kódra fordítás - Cython

- Python bővített változata
- C kódra fordít
- típusos

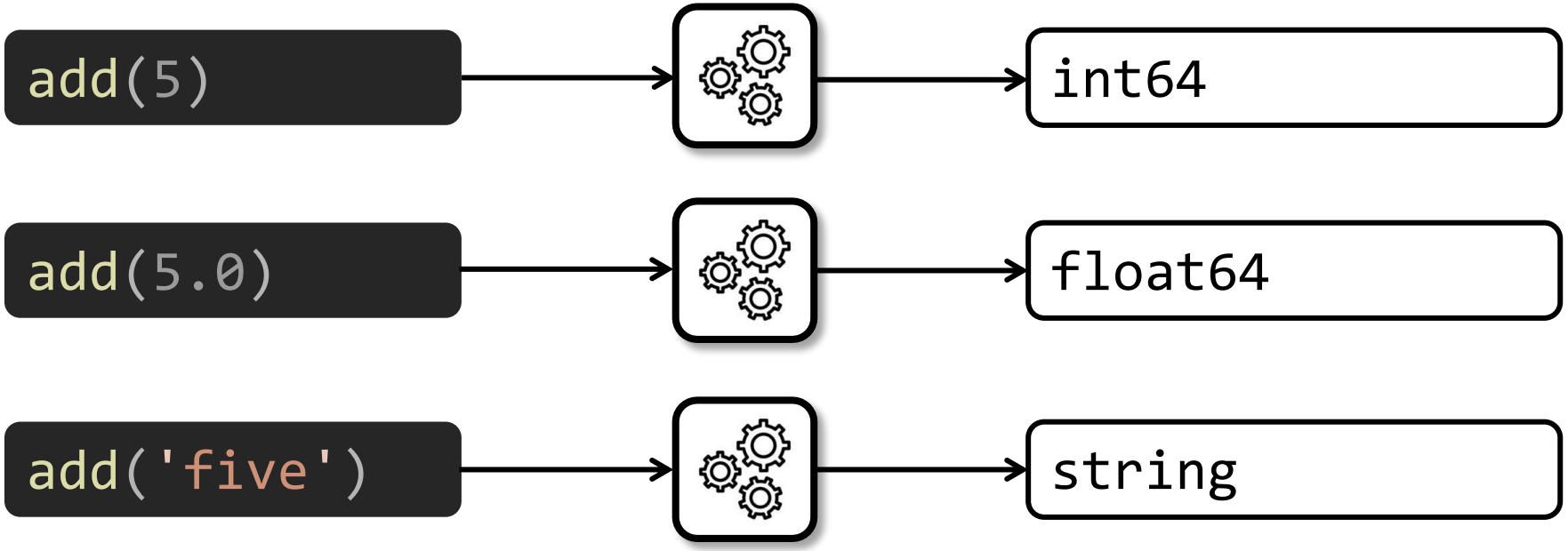
```
def f(double x):
    return x ** 2 - x

def integrate_f(double a, double b, int N):
    cdef int i
    cdef double s
    cdef double dx
    s = 0
    dx = (b - a) / N
    for i in range(N):
        s += f(a + i * dx)
    return s * dx
```

Gépi kódra fordítás - Numba

- Just In Time (JIT) fordító
- Futási időben fordít
- Új típus >> újrafordítás

Gépi kódra fordítás - Numba

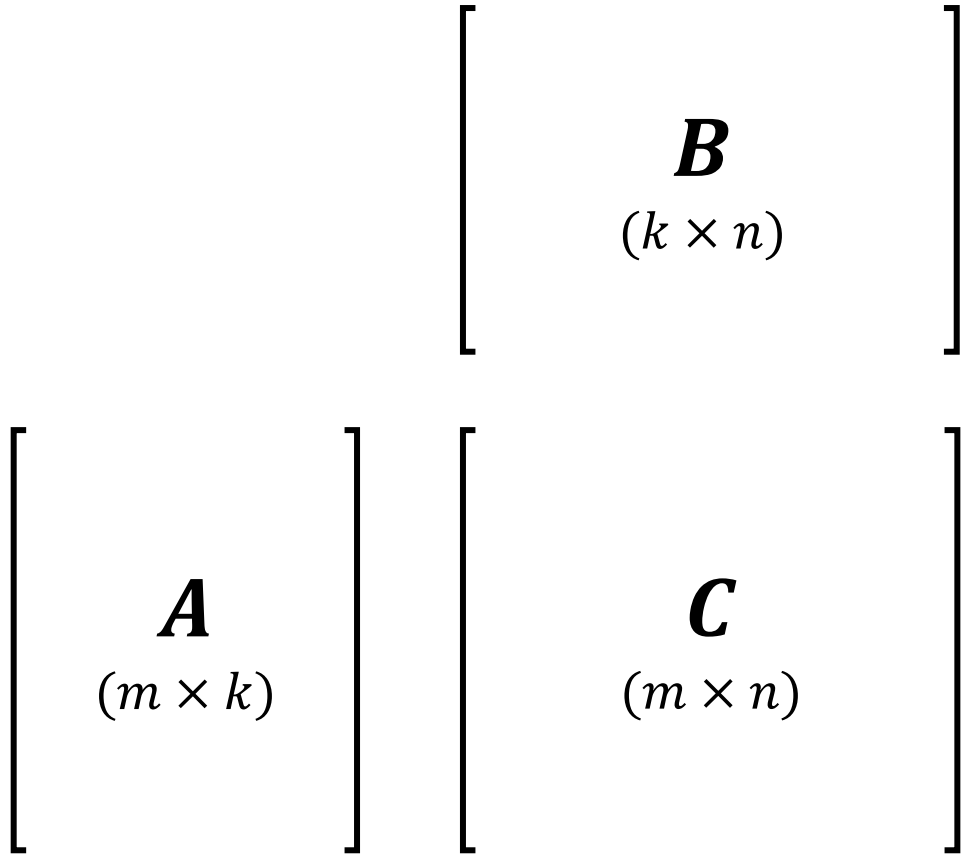
```
@jit
def add(a):
    return (a + a)
```



Java™

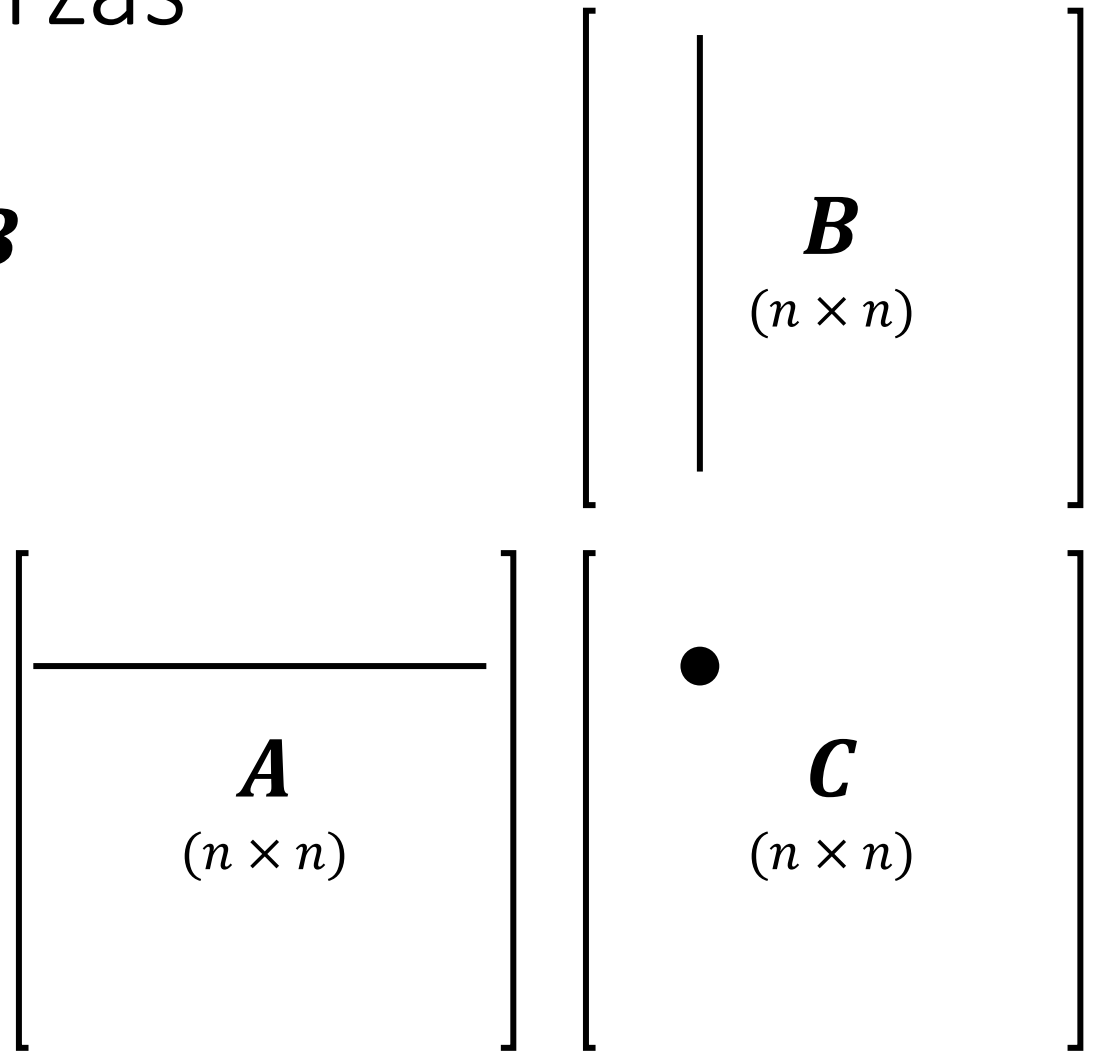
Mátrix szorzás

$$\boldsymbol{C = AB}$$



Mátrix szorzás

$$\boldsymbol{C = AB}$$



Összes számítás

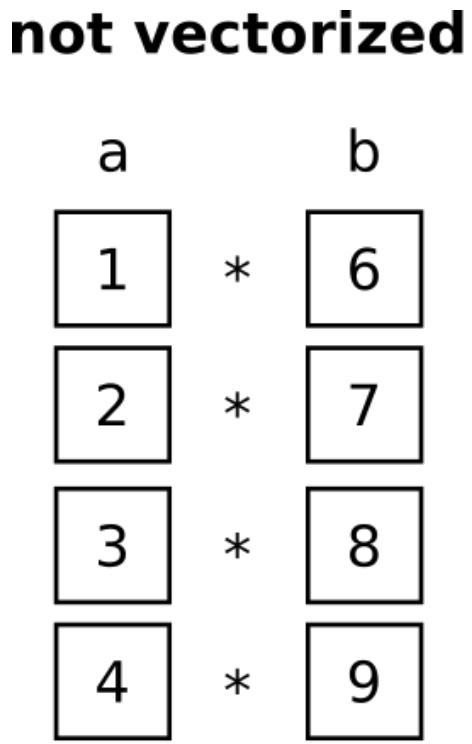
$n \cdot n \cdot (\dots)$

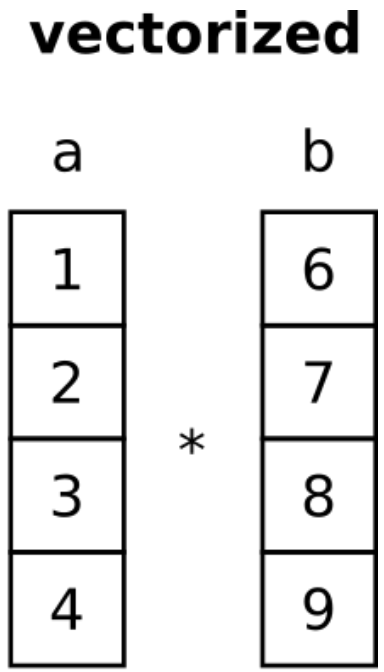
$n \cdot n \cdot (n + n - 1)$

$\mathcal{O}(n^3)$

Vektorizáció (Numpy)

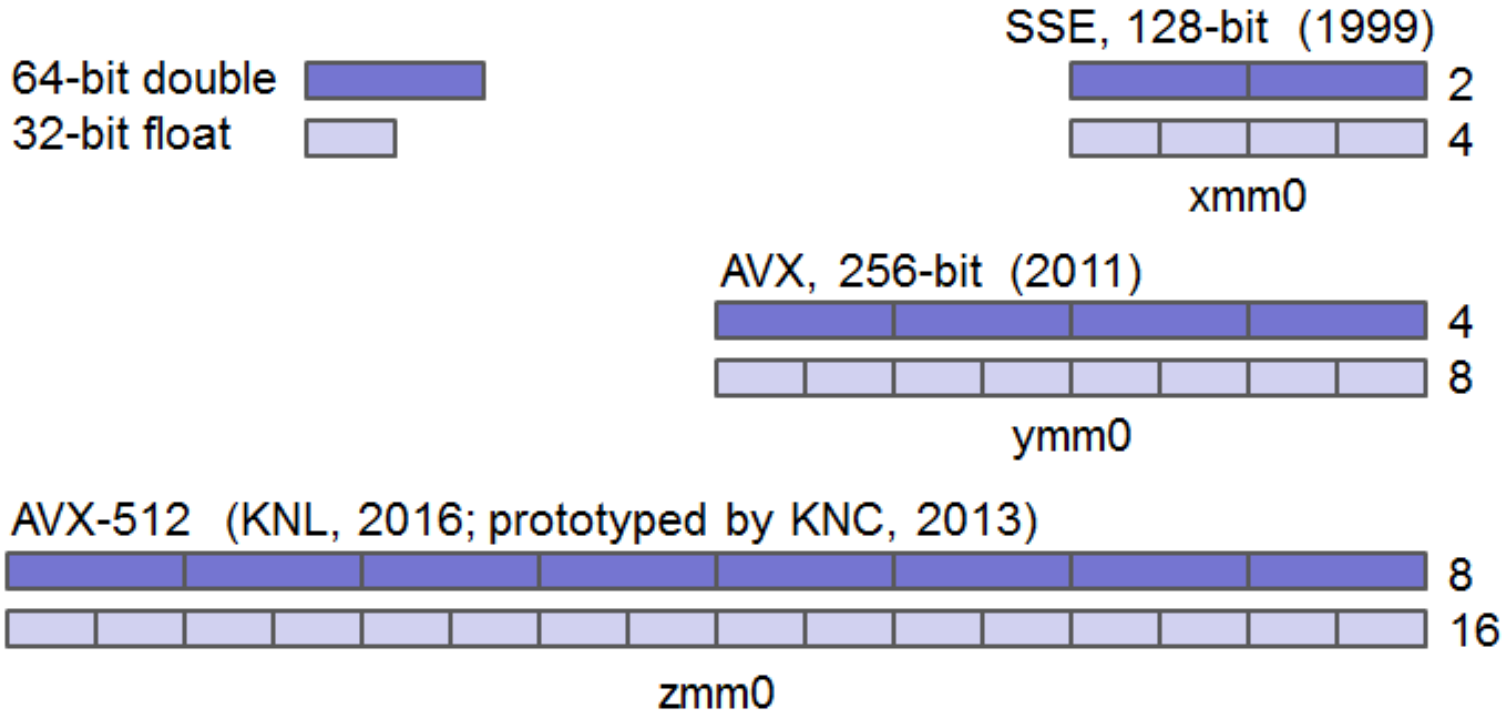
- Pytfon for ciklus: interpretált
- Numpy tömbök: gépi kód (C)





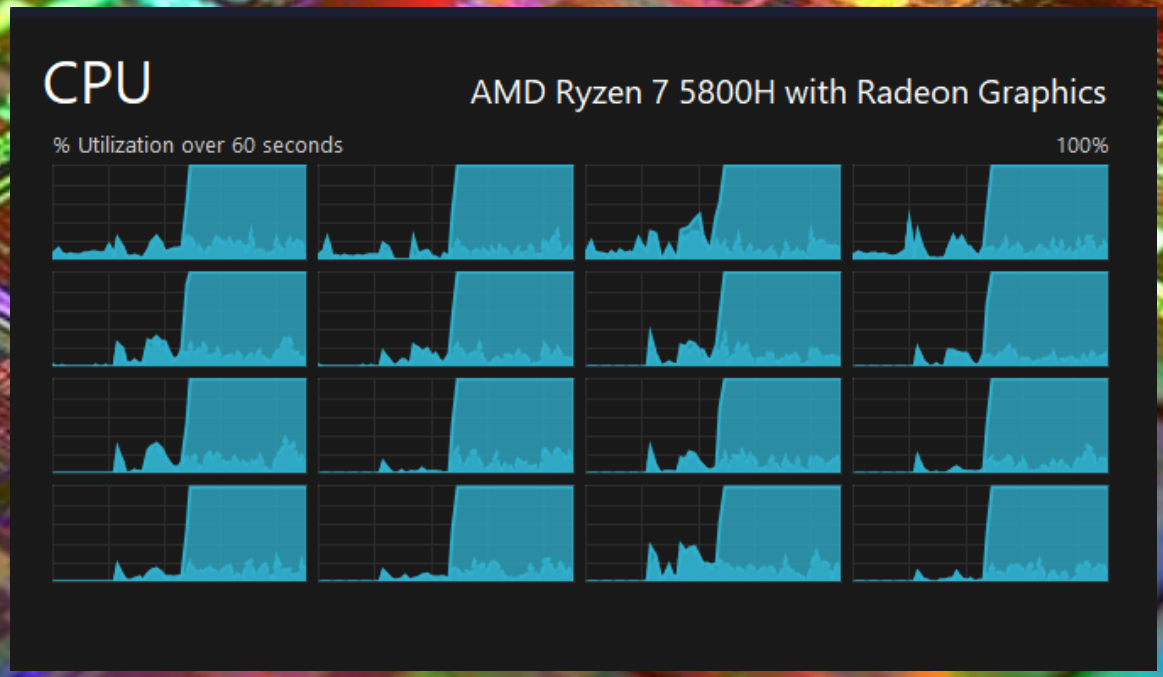
Vektor regiszterek

- C++ vector class library (Agner Fog)
- SIMD – Single Instruction Multiple Data

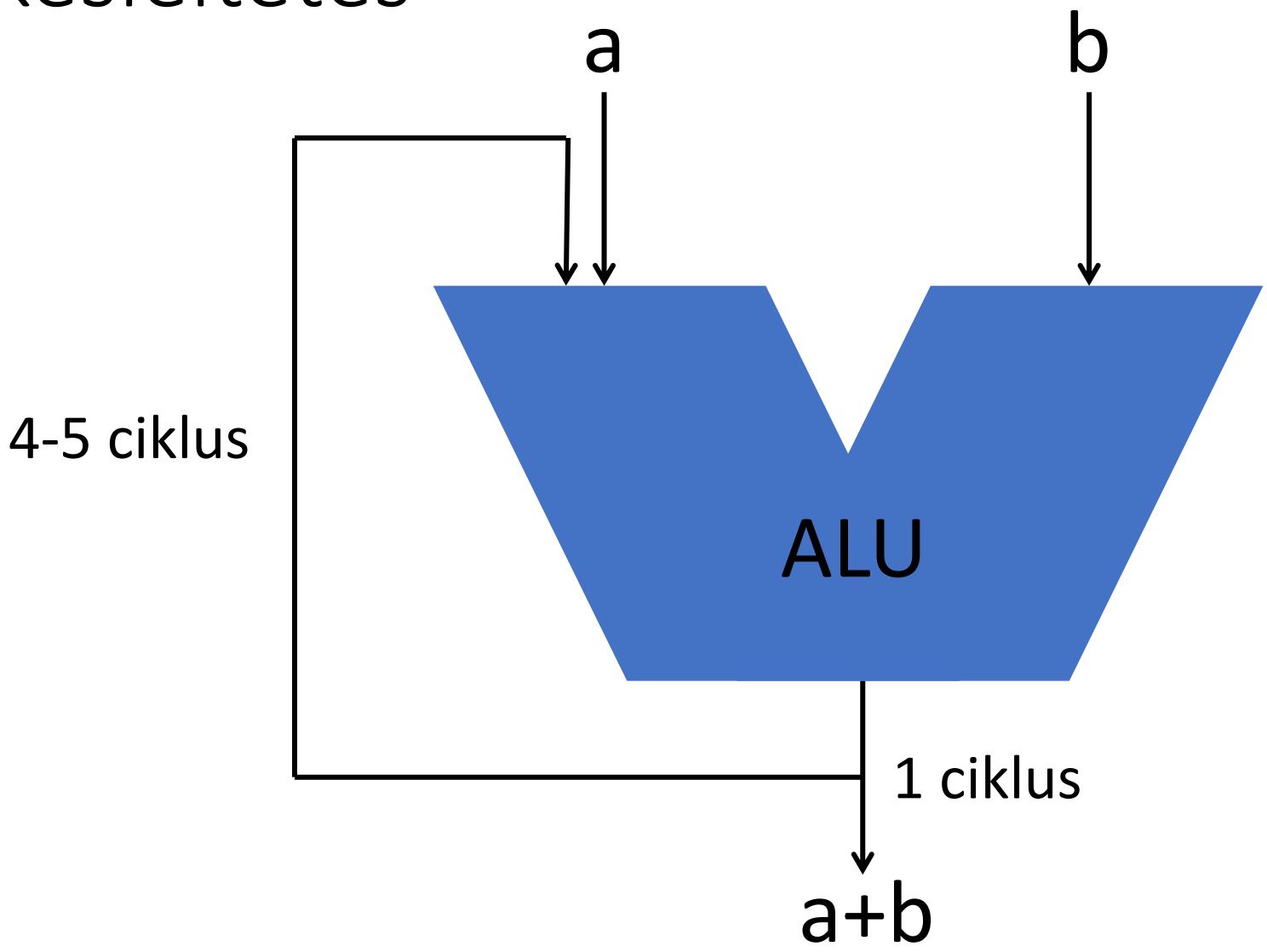


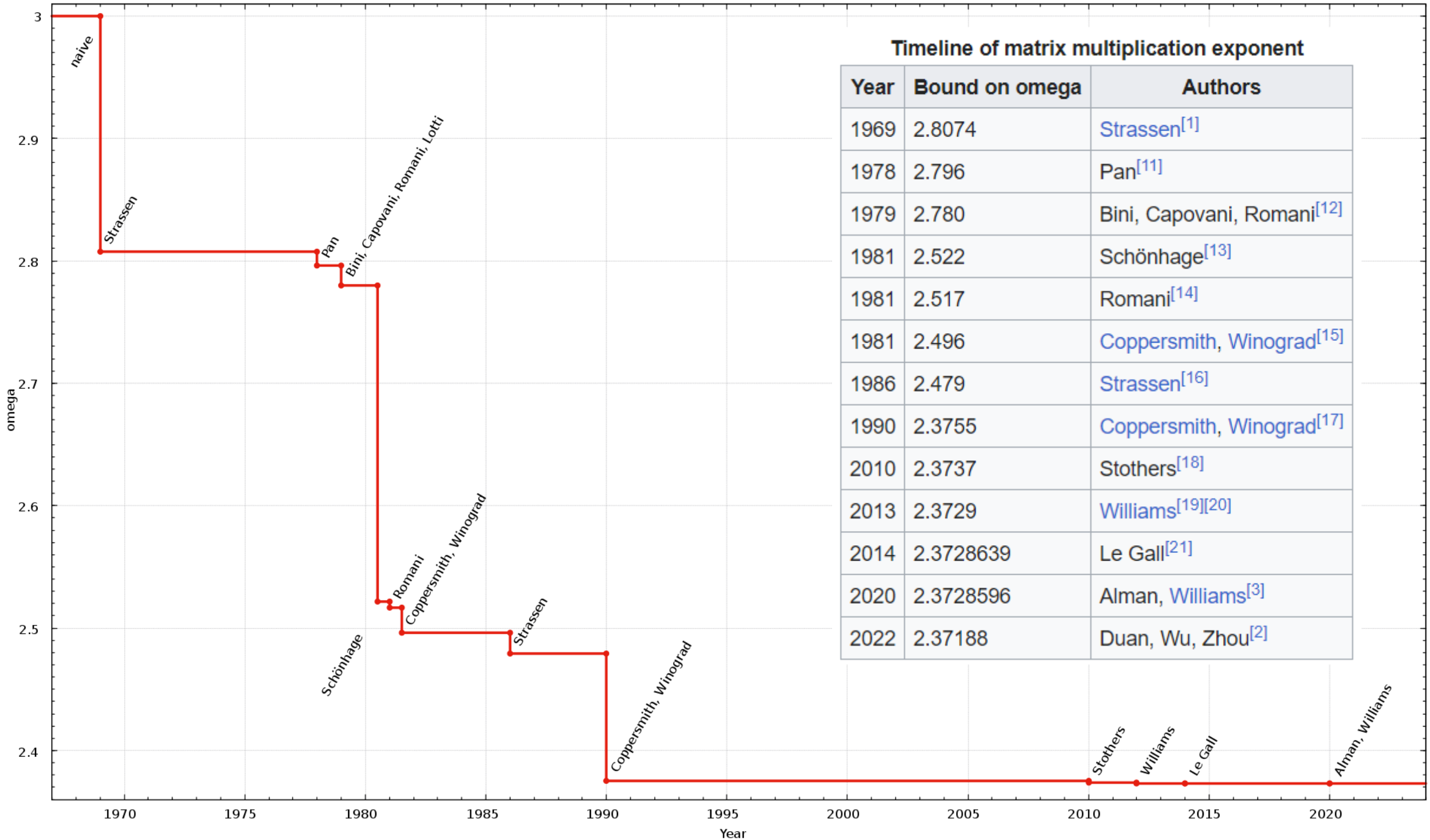
Párhuzamosítás (CPU)

- 8 mag – 16 szál
- Szálak együttműködése! megosztott memória



Késleltetés





Timeline of matrix multiplication exponent

Year	Bound on omega	Authors
1969	2.8074	Strassen[1]
1978	2.796	Pan[11]
1979	2.780	Bini, Capovani, Romani[12]
1981	2.522	Schönhage[13]
1981	2.517	Romani[14]
1981	2.496	Coppersmith, Winograd[15]
1986	2.479	Strassen[16]
1990	2.3755	Coppersmith, Winograd[17]
2010	2.3737	Stothers[18]
2013	2.3729	Williams[19][20]
2014	2.3728639	Le Gall[21]
2020	2.3728596	Alman, Williams[3]
2022	2.37188	Duan, Wu, Zhou[2]

CPU

Regiszterek (8-16 darab/mag/típus)

4-5 ciklus

L1 cache (64 kB/mag)

8-12 ciklus

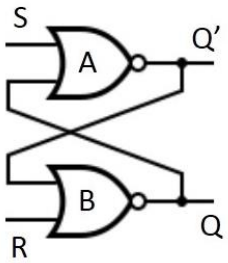
L2 cache (512 kB/mag)

40-60 ciklus

L3 cache (16 MB)

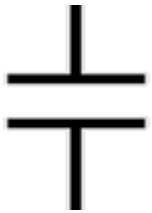
300-500 ciklus

SRAM



DRAM

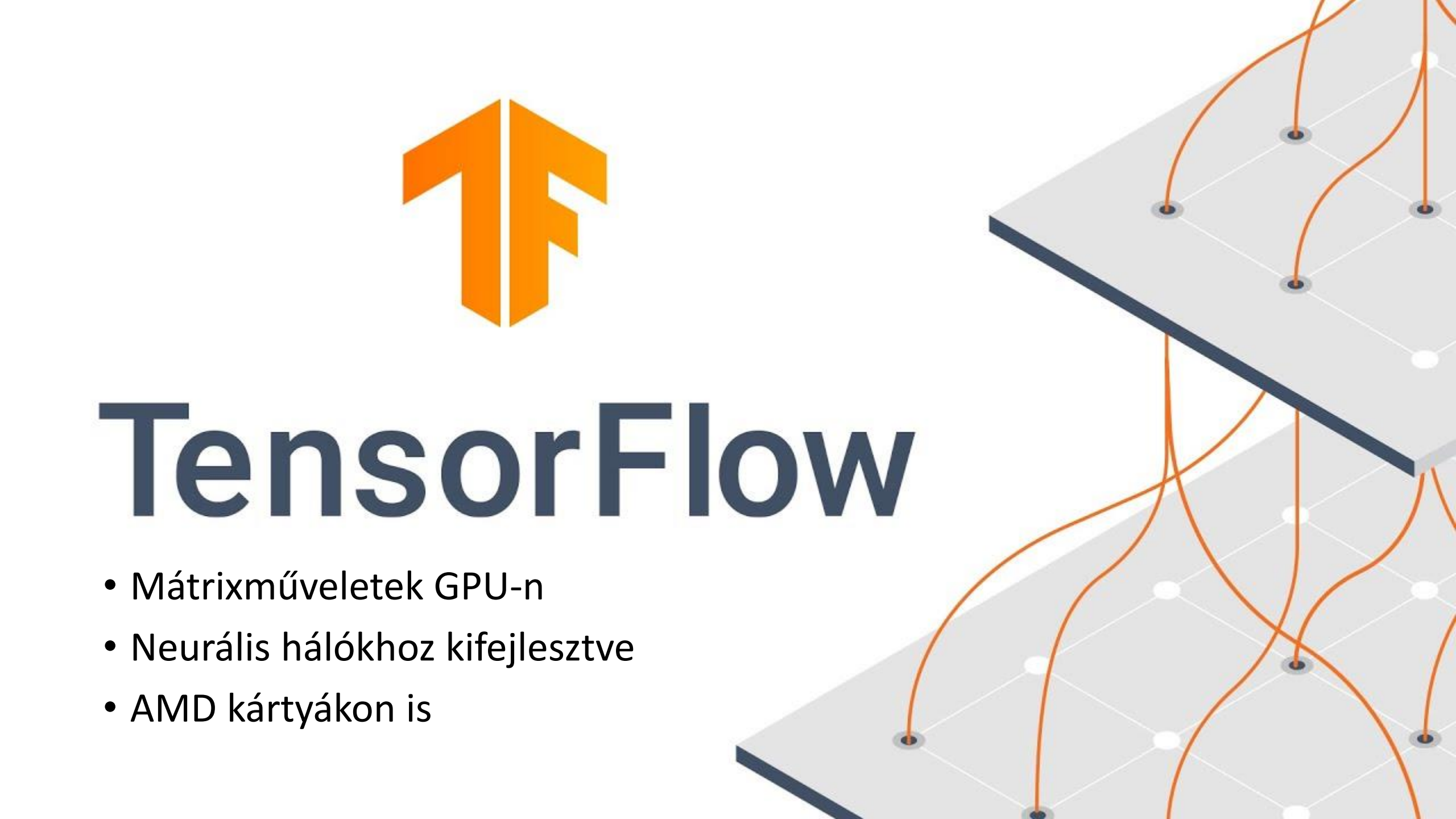
RAM (32 GB)



~50000 ciklus

SSD C (512 GB)

SSD D (512 GB)



TensorFlow

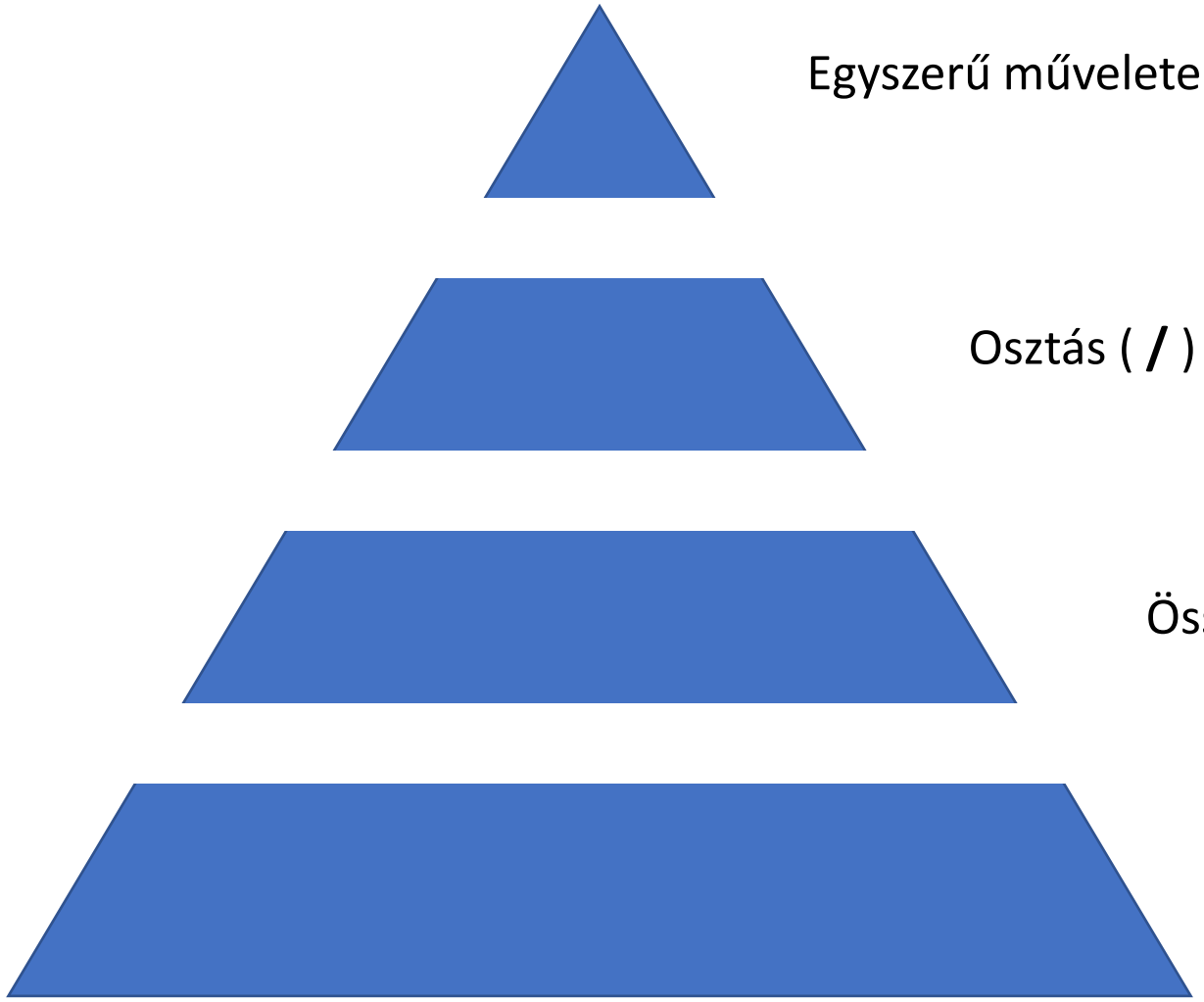
- Mátrixműveletek GPU-n
- Neurális hálókhoz kifejlesztve
- AMD kártyákon is

GPU

- 3584 CUDA mag

  NVIDIA GeForce RTX 3060
- block_size: min 32 thread
- grid_size: max 112

Műveletigény

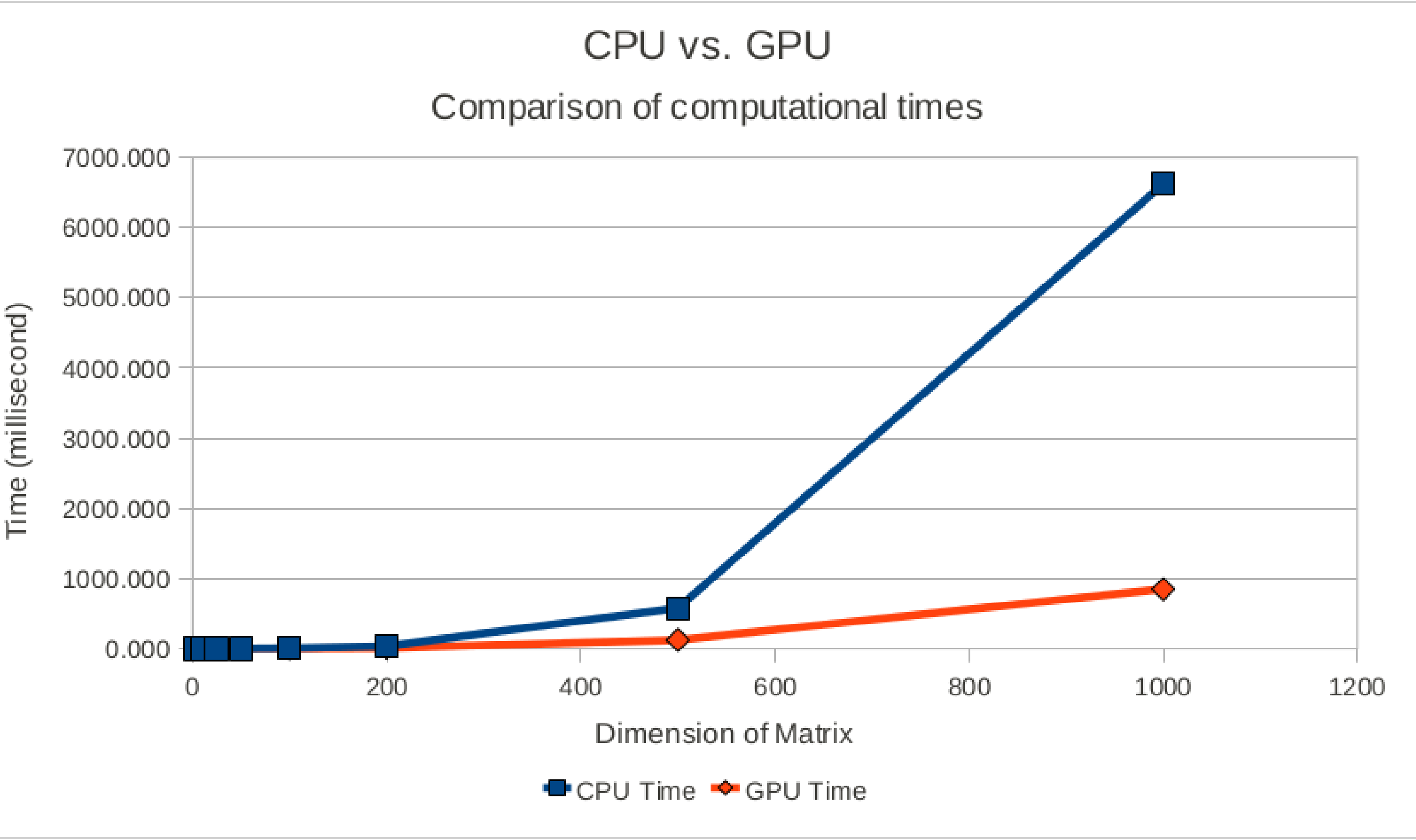


Egyszerű műveletek (**+**, *****, **or**, …)

Osztás (**/**)

Összetett műveletek (**sqrt**, **sin**, **exp**)

OP rendszer hívások (**File_IO**, **Cosole_IO**)


CPU vs. GPU
Comparison of computational times
7000.000
6000.000
5000.000
4000.000
3000.000
2000.000
1000.000
0.000
Time (millisecond)
0
200
400
600
800
1000
1200
Dimension of Matrix
CPU Time
GPU Time

$$z_{n+1} = z_n^2 + c$$

c can be changed

$$z_0 = 0$$

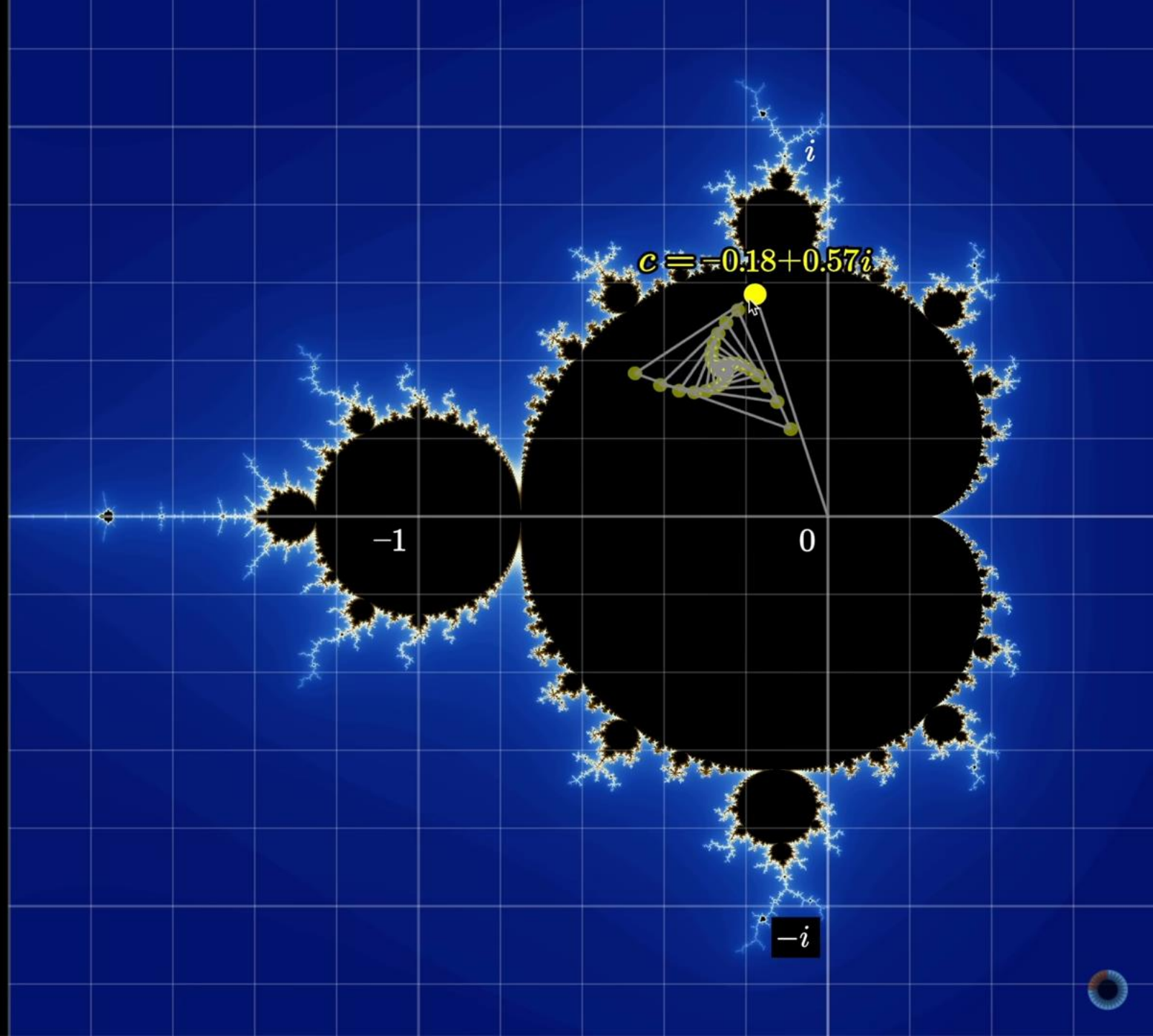
$$z_1 = 0^2 + c = c$$

$$z_2 = c^2 + c$$

$$z_3 = (c^2 + c)^2 + c$$

$$z_4 = ((c^2 + c)^2 + c)^2 + c$$

$$\vdots$$



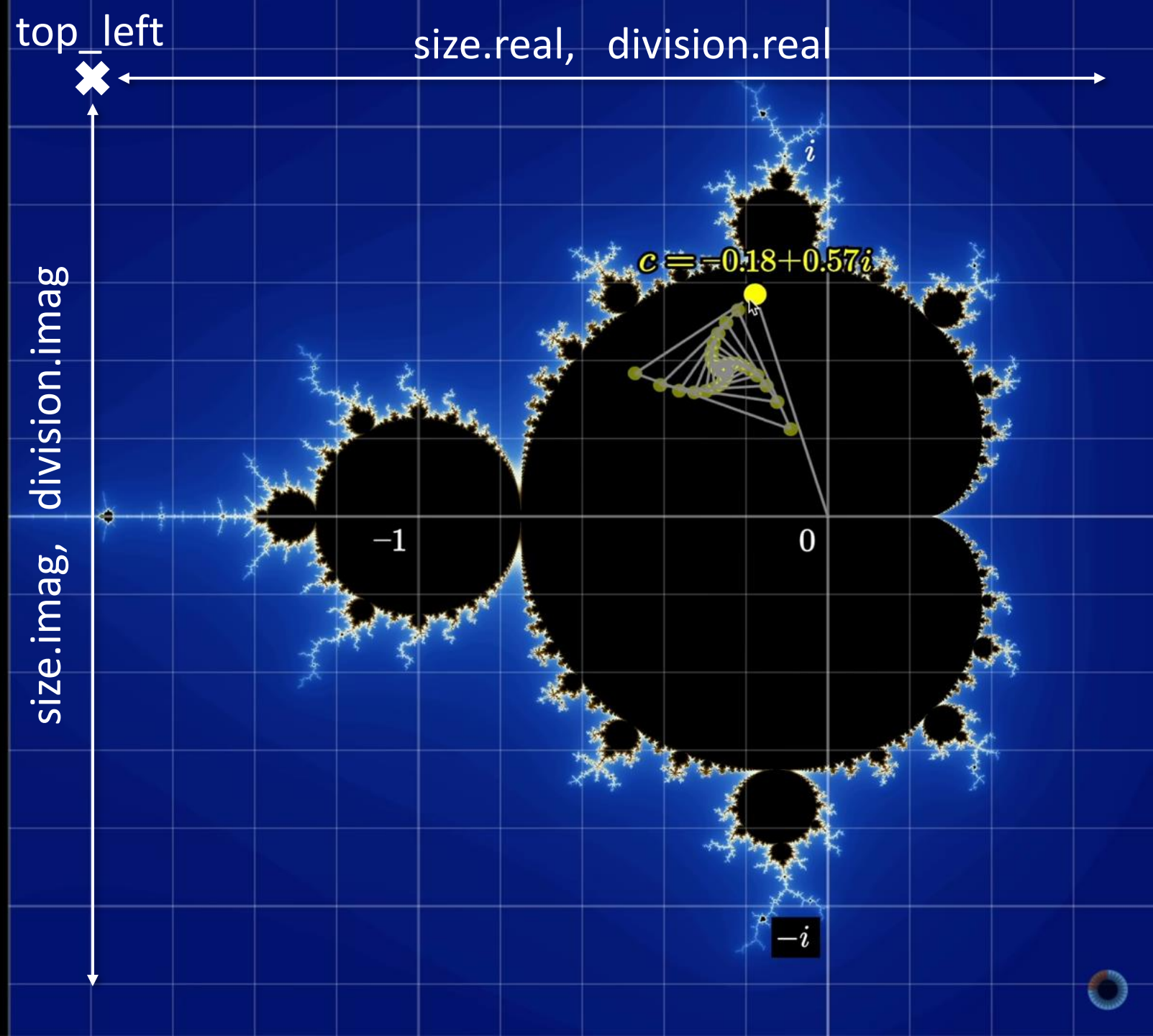
$$z_{n+1} = z_n^2 + c$$

c can be changed

1. $z_0 = 0$
2. $z_1 = 0^2 + c = c$
3. $z_2 = c^2 + c$
4. $z_3 = (c^2 + c)^2 + c$
5. $z_4 = ((c^2 + c)^2 + c)^2 + c$

$\vdots$

iteration_limit



Multiprocessing

```
import multiprocessing

def fun(x):
    return x**2

with multiprocessing.Pool() as pool:
    results = pool.imap_unordered(fun, range(10))

    for result in results:
        print(result, end=' ')
```

```
49 81 16 4 36 1 25 64 0 9
```


Köszönöm a figyelmet