NumPy I – tömbök (array), lista vs. NumPy array, vektorizált műveletek

Mi az a NumPy?

NumPy = numerikus számításokra optimalizált Python könyvtár.

Telepítés:

```
pip install numpy
```

Importálás:

```
import numpy as np
```

Miért nem elég a lista?

- Lassabb nagy adatmennyiségnél
- Nincs valódi vektorművelet
- Matematikai műveletek körülményesek

Lista vs. NumPy array

```
import numpy as np

#lista
szamok = [1, 2, 3, 4]
print(szamok * 2)
#Ez nem elemenkénti szorzás!

#NumPy array
arr = np.array([1, 2, 3, 4])
print(arr * 2)
#Ez elemenként szoroz.
```

Feladat:

- Hozz létre egy listát: [10, 20, 30, 40]
- Alakítsd NumPy array-jé
- Szorozd meg 3-mal

```

```

```
import numpy as np

lista = [10, 20, 30, 40]
arr = np.array(lista)

print(arr * 3)
```

Tömb létrehozása különböző módokon

```
np.zeros(5) #csak nullák
np.ones(5) #csak egyesek
np.arange(0, 10, 2) #mint range, de array
np.linspace(0, 1, 5) #intervallum felosztása
```

Feladat: Hozz létre egy array-t 0-tól 20-ig 5-ös lépéssel. Hozz létre 6 darab 1-est tartalmazó tömböt.

```
import numpy as np

arr1 = np.arange(0, 21, 5)
arr2 = np.ones(6)

print(arr1)
print(arr2)

#Indexelés és szeletelés
arr = np.array([10, 20, 30, 40, 50])

print(arr[0])
print(arr[1:4])
```

Szeletelés: start:end

Feladat: Hozz létre egy 0–9-ig terjedő array-t. Írd ki az első 5 elemet. Írd ki az utolsó 3 elemet

```
import numpy as np

arr = np.arange(10)

print(arr[:5])
print(arr[-3:])
```

Vektorizált műveletek

NumPy műveletek elemenként történnek.

```
arr = np.array([1, 2, 3, 4])
print(arr + 5)
print(arr ** 2)
print(arr.mean())
```

Fontos:

- Nincs szükség for ciklusra
- Gyorsabb és tisztább kód

Feladat: Hozz létre egy array-t 1–5-ig. Számold ki a négyzetüket. Számold ki az átlagukat

```
import numpy as np

arr = np.arange(1, 6)

negyzet = arr ** 2
atlag = arr.mean()

print("Négyzetek:", negyzet)
print("Átlag:", atlag)
```

Összefoglaló feladatok

1. Feladat: Az adat.txt fájl számokat tartalmaz. Írj programot, ami:

- Beolvassa a számokat NumPy array-be
- Kiszámolja az átlagot és szórást
- Kiírja az eredményeket

```
import numpy as np

adat = np.loadtxt("adat.txt")

print("Átlag:", np.mean(adat))
print("Szórás:", np.std(adat))
```

2. Feladat – Lista vs NumPy teljesítmény és műveletek

Hozz létre 1 millió elemű számlistát:

- sima Python listával
- NumPy tömbbel
- Szorozd meg minden elemét 2-vel:
- listán ciklussal
- NumPy array-n vektorizáltan
- Mérd az időt.
- Hasonlítsd össze az eredményeket.

```
import numpy as np
import time

# Lista
lista = list(range(1_000_000))

start = time.time()
uj_lista = []
for x in lista:
    uj_lista.append(x * 2)
print("Lista idő:", time.time() - start)

# NumPy
array = np.arange(1_000_000)

start = time.time()
uj_array = array * 2
print("NumPy idő:", time.time() - start)
```

3. Feladat – Fájlbeolvasás és statisztikai elemzés

Adott egy adatok.txt fájl, amely soronként számokat tartalmaz.

- Olvasd be listába.
- Alakítsd NumPy tömbbé.
- Számold ki:
    - átlag
    - szórás
    - minimum
    - maximum
- Szűrd ki a 10 feletti értékeket (vektorizáltan).

```
import numpy as np

def beolvasas(fajlnev):
    lista = []
    try:
```

```
        with open(fajlnev, "r") as f:
            for sor in f:
                lista.append(float(sor.strip()))
    except FileNotFoundError:
        print("A fájl nem található!")
    return lista

lista = beolvasas("adatok.txt")
array = np.array(lista)

print("Átlag:", np.mean(array))
print("Szórás:", np.std(array))
print("Minimum:", np.min(array))
print("Maximum:", np.max(array))

szurt = array[array > 10]
print("10 felettiek:", szurt)
```

4. Feladat – Mátrixműveletek és vektorizált számítás

- Generálj egy 5×5-ös véletlen mátrixot.
- Számold ki:
  - soronkénti összeget
  - oszloponkénti átlagot
  - Normalizáld a mátrixot (0–1 közé skálázás).
  - Ne használj ciklust, csak NumPy műveleteket.

```
import numpy as np

matrix = np.random.randint(1, 100, (5, 5))

print("Mátrix:\n", matrix)

# Sorösszeg
sor_osszeg = np.sum(matrix, axis=1)
print("Sorösszegek:", sor_osszeg)

# Oszlopátlag
oszlop_atlag = np.mean(matrix, axis=0)
print("Oszlopátlag:", oszlop_atlag)

# Normalizálás
min_val = np.min(matrix)
max_val = np.max(matrix)

normalizalt = (matrix - min_val) / (max_val - min_val)
print("Normalizált mátrix:\n", normalizalt)
```

5. Feladat – Két dimenziós adatfeldolgozás (vizsgaeredmények)

- Egy 2D NumPy tömb sorai diákok, oszlopai tantárgyak pontszámai.
- Számold ki minden diák átlagát.
- Számold ki a tantárgyankénti átlagot.
- Szűrd ki azokat a diákokat, akiknek az átlaga 60 alatt van.
- Hasonlítsd össze listás és NumPy megoldással.

```
import numpy as np

pontok = np.array([
    [70, 80, 90],
    [50, 40, 60],
    [90, 95, 100],
    [30, 50, 45]
])

# Diák átlag
diak_atlag = np.mean(pontok, axis=1)
print("Diák átlagok:", diak_atlag)

# Tantárgy átlag
tantargy_atlag = np.mean(pontok, axis=0)
print("Tantárgy átlagok:", tantargy_atlag)

# Bukottak
bukott_index = np.where(diak_atlag < 60)
print("Bukott diák indexek:", bukott_index)
```

6. Feladat – Idősoros adatelemzés (komplex)

- Generálj 365 napos hőmérsékleti adatot.
- Számold ki:
    - éves átlag
    - legmelegebb 10 nap
    - hány nap volt 30°C felett
    - Mozgóátlag (7 napos).
- Oldd meg ciklus nélkül, vektorizáltan.

```
import numpy as np

# Generált adatok
homerseklet = np.random.normal(20, 10, 365)

# Éves átlag
print("Éves átlag:", np.mean(homerseklet))
```

```
# Legmelegebb 10 nap
legmelegebb = np.sort(homerseklet)[-10:]
print("Top 10 meleg nap:", legmelegebb)

# 30 fok feletti napok
print("30 fok feletti napok száma:", np.sum(homerseklet > 30))

# 7 napos mozgóátlag
kernel = np.ones(7) / 7
mozgo_atlag = np.convolve(homerseklet, kernel, mode='valid')

print("Mozgóátlag első 10 értéke:", mozgo_atlag[:10])
```