Tömb létrehozása

- Python listából / tuple-ből NumPy tömböt hoz létre: **np.array(iterable)**
- Tartomány generálása (mint range, de array): **np.arange(start, stop, step)**
- Egyenletesen elosztott num darab érték adott intervallumban: **np.linspace(start, stop, num)**
- Nullákkal feltöltött tömb: **np.zeros(shape)**
- 1-esekkel feltöltött tömb: **np.ones(shape)**
- Inicializálatlan tömb (gyors, de szemétértékkel): **np.empty(shape)**
- Adott értékkel feltöltött tömb: **np.full(shape, value)**
- Egységmátrix: **np.eye(n)**
- 0–1 közötti egyenletes eloszlás: **np.random.rand(…)**
- Normál eloszlás: **np.random.randn(…)**
- Véletlen egész számok: **np.random.randint(low, high, size)**

Tömb tulajdonságok

- Dimenziók mérete: **arr.shape**
- Dimenziók száma: **arr.ndim**
- Elemek száma: **arr.size**
- Adattípus: **arr.dtype**
- Típuskonverzió: **arr.astype(type)**

Matematikai alapműveletek (vektorizált)

- Elemankénti műveletek.:
  - **np.add(a, b)**
  - **np.subtract(a, b)**
  - **np.multiply(a, b)**
  - **np.divide(a, b)**
- Hatványozás: **np.power(a, b)**
- Négyzetgyök: **np.sqrt(a)**
- Abszolútérték: **np.abs(a)**
- e^x: **np.exp(a)**
- Természetes logaritmus: **np.log(a)**

Aggregáló (redukciós) függvények

- Összegzés: **np.sum(a, axis=None)**
- Átlag: **np.mean(a, axis=None)**
- Medián: **np.median(a)**
- Szórás: **np.std(a)**
- Variancia: **np.var(a)**
- Minimum: **np.min(a)**
- Maximum: **np.max(a)**
- Minimum indexe: **np.argmin(a)**
- Maximum indexe: **np.argmax(a)**
- Szorzat: **np.prod(a)**

Mátrix- és lineáris algebra

- Skalárszorzat / mátrixszorzás (régebbi forma): **np.dot(a, b)**
- Modern mátrixszorzás operátor: **a @ b**
- Mátrixszorzás: **np.matmul(a, b)**
- Transzponálás: **np.transpose(a)** / **a.T**
- Inverz mátrix: **np.linalg.inv(a)**
- Determináns: **np.linalg.det(a)**
- Sajátértékek, sajátvektorok: **np.linalg.eig(a)**
- Lineáris egyenletrendszer megoldása: **np.linalg.solve(A, b)**

Indexelés és szűrés

- Boolean indexelés, feltételes szűrés: **a[a > 5]**
- Feltétel indexei: **np.where(condition)**
- Nem nulla elemek indexei: **np.nonzero(a)**
- Értékek levágása intervallumra: **np.clip(a, min, max)**

Tömb átalakítás

- Átalakítás: **np.reshape(a, shape)**
- 1D-re lapítás (másolat): **a.flatten()**
- 1D-re lapítás (nézet, ha lehet): **a.ravel()**
- Összefűzés: **np.concatenate([a, b], axis=0)**
- Függőleges összefűzés: **np.vstack([…])**
- Vízszintes összefűzés: **np.hstack([…])**
- Felosztás: **np.split(a, indices)**

Rendezés és keresés

- Rendezett másolat: **np.sort(a)**
- Helyben rendez: **a.sort()**
- Rendezési indexek: **np.argsort(a)**
- Egyedi elemek: **np.unique(a)**

Statisztikai / speciális

- Percentilis: **np.percentile(a, q)**
- Kumulatív összeg: **np.cumsum(a)**
- Kumulatív szorzat: **np.cumprod(a)**
- Korreláció: **np.corrcoef(a, b)**
- Konvolúció (pl. mozgóátlag): **np.convolve(a, kernel, mode)**

Broadcasting (nem függvény, hanem mechanizmus)

- Automatikus méretillesztés műveleteknél:

```
a + 5
a + np.array([1,2,3])
```

Fontos logikai függvények

- Minden elem igaz? **np.all(condition)**
- Van legalább egy igaz? **np.any(condition)**
- És: **np.logical_and(a, b)**
- Vagy: **np.logical_or(a, b)**

Fájlkezelés NumPy-val

- Szövegfájl betöltése: **np.loadtxt("file.txt")**
- Mentés szövegfájlba: **np.savetxt("file.txt", array)**
- Bináris mentés: **np.save("file.npy", array)**
- Bináris betöltés: **np.load("file.npy")**

A legfontosabb 20, amit tudni kell

```
array, arange, linspace
zeros, ones, full
shape, reshape
sum, mean, std, min, max
argmax, argmin
dot / @
where
sort, argsort
concatenate
unique
loadtxt, savetxt
```

Gyakorló feladatok

I. Tömb létrehozás: Alap generálás

1. Hozz létre egy 0–99 közötti egész számokat tartalmazó tömböt np.arange() segítségével.
2. Hozz létre 50 darab egyenletesen elosztott számot 0 és 1 között np.linspace()-szel.
3. Készíts egy 4×4-es nullmátrixot.
4. Készíts egy 3×5-ös, 7-esekkel feltöltött mátrixot.

```
import numpy as np

# 1
a = np.arange(100)
# 2
b = np.linspace(0, 1, 50)
# 3
c = np.zeros((4, 4))
# 4
d = np.full((3, 5), 7)
print(a, b, c, d)
```

II. Tömb tulajdonságok és átalakítás: Shape és reshape

1. Generálj egy 1D tömböt 0–23-ig.
2. Alakítsd át 4×6-os mátrixszá.
3. Transzponáld.
4. Lapítsd vissza 1D-be flatten() és ravel() segítségével.
5. Ellenőrizd shape, ndim, size, dtype.

```
arr = np.arange(24)

matrix = arr.reshape(4, 6)
transposed = matrix.T

flat1 = matrix.flatten()
flat2 = matrix.ravel()

print(matrix.shape)
print(matrix.ndim)
print(matrix.size)
print(matrix.dtype)
```

III. Vektorizált matematikai műveletek: Elemankénti műveletek

Hozz létre két 10 elemű tömböt.

1. Számold ki:
    1. összegüket
    2. különbségüket
    3. szorzatukat
    4. hányadosukat
2. Emeld az egyik tömb elemeit négyzetre.
3. Számold ki a négyzetgyöküket.
4. Alkalmazz logaritmust és exponenciális függvényt.

```
```

```
a = np.arange(1, 11)
b = np.arange(11, 21)

osszeg = a + b
kulonbseg = a - b
szorzat = a * b
hanyados = a / b

negyzet = a ** 2
gyok = np.sqrt(a)

log = np.log(a)
exp = np.exp(a)

print(osszeg, kulonbseg, szorzat, hanyados)
```

IV. Aggregáló függvények: Statisztikai elemzés

1. Generálj 1000 normál eloszlású számot.
2. Számold ki:
    1. átlag
    2. medián
    3. szórás
    4. variancia
    5. minimum
    6. maximum
3. Add meg a legnagyobb elem indexét.
4. Számold ki az első 100 elem összegét.

```
data = np.random.randn(1000)

print("Átlag:", np.mean(data))
print("Medián:", np.median(data))
print("Szórás:", np.std(data))
print("Variancia:", np.var(data))
print("Min:", np.min(data))
print("Max:", np.max(data))

print("Max index:", np.argmax(data))
print("Első 100 elem összege:", np.sum(data[:100]))
```

V. Mátrixműveletek: Lineáris algebra

1. Generálj két 3×3-as mátrixot.
2. Szorozd össze őket:
    1. np.dot()
    2. @ operátorral
3. Számold ki az egyik determinánsát.

4. Számold ki az inverzét.
5. Oldj meg egy Ax = b lineáris egyenletrendszert.

```
A = np.random.randint(1, 10, (3, 3))
B = np.random.randint(1, 10, (3, 3))

dot1 = np.dot(A, B)
dot2 = A @ B

det = np.linalg.det(A)

# Inverz csak ha invertálható
if det != 0:
    inv = np.linalg.inv(A)

b = np.array([1, 2, 3])
if det != 0:
    x = np.linalg.solve(A, b)

print(dot1, det)
```

VI. Boolean indexelés és szűrés: Feltételes műveletek

1. Generálj 100 elemből álló véletlen tömböt.
2. Szűrd ki az 50-nél nagyobb elemeket.
3. Cseréld az 50 alatti elemeket nullára np.where() segítségével.
4. Számold meg, hány elem nagyobb 75-nél.
5. Használd np.clip()-et az értékek 10–90 közé szorítására.

```
arr = np.random.randint(0, 100, 100)
nagyobb_50 = arr[arr > 50]
uj = np.where(arr < 50, 0, arr)
db_75 = np.sum(arr > 75)
clipped = np.clip(arr, 10, 90)
print(nagyobb_50, db_75)
```

VII. Összefűzés és darabolás: Tömbmanipuláció

1. Hozz létre két 2×3-as mátrixot.
2. Fűzd össze:
    1. függőlegesen (vstack)
    2. vízszintesen (hstack)
    3. Használd concatenate()-et.
3. Oszd fel a mátrixot 3 részre split() segítségével.

```
A = np.random.randint(1, 10, (2, 3))
B = np.random.randint(1, 10, (2, 3))

v = np.vstack((A, B))
h = np.hstack((A, B))

c = np.concatenate((A, B), axis=0)
split = np.split(v, 2)
print(v, h)
```

VIII. Rendezés és keresés:mRendezési feladatok

1. Generálj 20 véletlen számot.
2. Rendezd növekvő sorrendbe.
3. Add meg a rendezési indexeket (argsort).
4. Találd meg az egyedi elemeket (unique).
5. Számold ki a 90. percentilist.

```
arr = np.random.randint(0, 50, 20)

sorted_arr = np.sort(arr)
indices = np.argsort(arr)
unique_vals = np.unique(arr)

p90 = np.percentile(arr, 90)

print(sorted_arr, indices, unique_vals, p90)
```

IX. Haladó – Idősor feldolgozás:mMozgóátlag és kumuláció

1. Generálj 365 napi adatot.
2. Számold ki:
    1. kumulatív összeget (cumsum)
    2. kumulatív szorzatot (cumprod)
3. Számolj 7 napos mozgóátlagot np.convolve() segítségével.
4. Határozd meg azokat a napokat, ahol az érték nagyobb az éves átlagnál.

```
data = np.random.randn(365)

cumsum = np.cumsum(data)
cumprod = np.cumprod(data)

kernel = np.ones(7) / 7
moving_avg = np.convolve(data, kernel, mode="valid")

atlag = np.mean(data)
nagyobb = data[data > atlag]
```

```
print(cumsum[:5], moving_avg[:5])
```

X. NumPy fájlkezelés: Fájlműveletek

1. Ments el egy tömböt .txt formátumban (savetxt).
2. Olvasd vissza (loadtxt).
3. Mentsd el .npy formátumban.
4. Töltsd vissza.
5. Ellenőrizd, hogy a két tömb megegyezik-e (np.array_equal).

```
arr = np.random.randint(1, 100, (5, 5))

np.savetxt("adat.txt", arr)
loaded_txt = np.loadtxt("adat.txt")

np.save("adat.npy", arr)
loaded_npy = np.load("adat.npy")

print(np.array_equal(arr, loaded_txt))
print(np.array_equal(arr, loaded_npy))
```

XI. Broadcasting gyakorlás: Méretillesztés

1. Adj hozzá egy 1D tömböt egy 2D mátrix minden sorához.
2. Normalizálj soronként.
3. Vond ki minden oszlopból az oszlopátlagot.
4. Szorozd meg az egész mátrixot egy skalárral.
5. Ellenőrizd np.all() és np.any() használatával bizonyos feltételeket.

```
matrix = np.random.randint(1, 10, (4, 3))
vector = np.array([1, 2, 3])

# Soronkénti hozzáadás
result = matrix + vector

# Soronkénti normalizálás
row_mean = np.mean(matrix, axis=1, keepdims=True)
norm = matrix - row_mean

# Oszlopátlag kivonása
col_mean = np.mean(matrix, axis=0)
centered = matrix - col_mean

# Skalárszorzás
scaled = matrix * 10
```

```
print(np.all(matrix > 0))
print(np.any(matrix > 8))
```