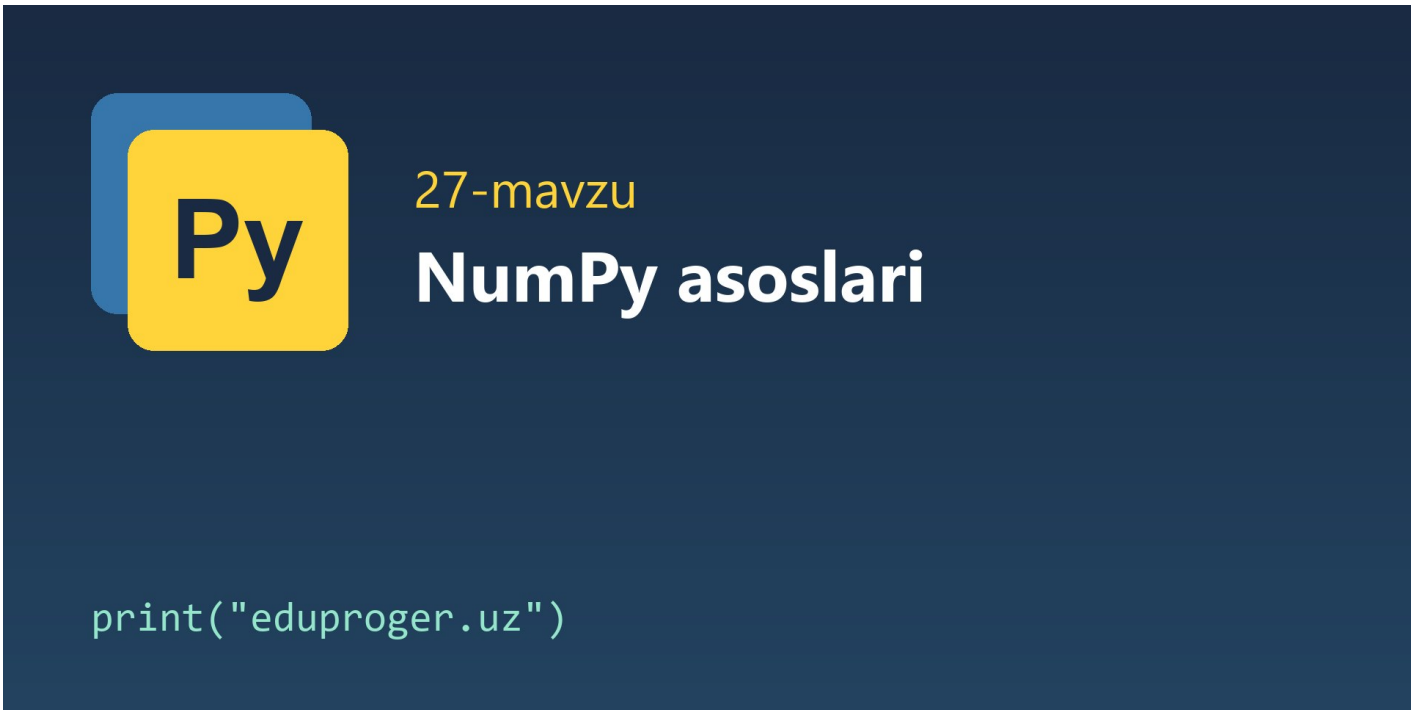
27. NumPy asoslari



NumPy — Pythonda sonli hisoblashlar uchun asosiy kutubxona: ma'lumotlar tahlili, sun'iy intellekt, ilmiy hisoblar shunga quriladi. Uning markazida **ndarray** — bir xil turdagi sonlar massivi. U oddiy ro'yxatdan o'nlab marta tez ishlaydi va sikllarsiz yoziladi.

> **O'rnatish**
>
> O'rnatish: `pip install numpy` (Anaconda'da oldindan bor). Saytdagi muharrir uni birinchi `import numpy` da o'zi yuklaydi (bir necha soniya).

1. Massiv yaratish

```
import numpy as np
a = np.array([1, 2, 3, 4, 5])
print(a, type(a).__name__, a.dtype)
print(np.arange(0, 10, 2))
print(np.linspace(0, 1, 5))
print(np.zeros(3), np.ones(3, dtype=int))
```

Natija:

```
[1 2 3 4 5] ndarray int64
[0 2 4 6 8]
[0.   0.25 0.5  0.75 1.  ]
[0. 0. 0.] [1 1 1]
```

`import numpy as np` — umumqabul qilingan qisqa nom. `arange` — range kabi, `linspace(a, b, n)` — a dan b gacha n ta teng oraliqli son.

2. Vektorli amallar — siklsiz

Amallar butun massivga BIR YO'LA qo'llanadi — `for` kerak emas:

```
import numpy as np
a = np.array([1, 2, 3, 4])
b = np.array([10, 20, 30, 40])
print(a + b)
print(a * 2)
print(a ** 2)
print(b / a)
print(np.sqrt(a))
```

Natija:

```
[11 22 33 44]
[2 4 6 8]
[ 1  4  9 16]
[10. 10. 10. 10.]
[1.         1.41421356 1.73205081 2.        ]
```

Oddiy ro'yxatda `[1, 2] * 2` → `[1, 2, 1, 2]` (takrorlash), NumPy da esa → `[2, 4]` (har elementni ko'paytirish). Farqni unutmang!

```
import numpy as np
print([1, 2] * 2)
print(np.array([1, 2]) * 2)
```

Natija:

```
[1, 2, 1, 2]
[2 4]
```

3. Statistika

```
import numpy as np
ballar = np.array([78, 95, 88, 64, 52, 91, 70])
print("yig'indi:", ballar.sum())
print("o'rtacha:", ballar.mean().round(2))
print("max/min:", ballar.max(), ballar.min())
print("eng kattaning indeksi:", ballar.argmax())
print("std:", ballar.std().round(2))
print("median:", np.median(ballar))
```

Natija:

```
yig'indi: 538
o'rtacha: 76.86
max/min: 95 52
eng kattaning indeksi: 1
std: 14.56
```

```
median: 78.0
```

4. Indekslash va kesish

```
import numpy as np
a = np.arange(10, 20)
print(a)
print(a[0], a[-1], a[2:5], a[::3])
a[0] = 99
print(a)
```

Natija:

```
[10 11 12 13 14 15 16 17 18 19]
10 19 [12 13 14] [10 13 16 19]
[99 11 12 13 14 15 16 17 18 19]
```

5. Mantiqiy filtrlash

Shart har bir element uchun True/False massivini beradi; uni indeks sifatida ishlatib, shartga mos elementlarni olamiz:

```
import numpy as np
ballar = np.array([78, 95, 88, 64, 52, 91, 70])
print(ballar >= 70)
print(ballar[ballar >= 70])
print("o'tganlar soni:", (ballar >= 70).sum())
print(ballar[(ballar > 60) & (ballar < 90)])
print(np.where(ballar >= 70, "o'tdi", "yiqildi"))
```

Natija:

```
[ True  True  True False False  True  True]
[78 95 88 91 70]
o'tganlar soni: 5
[78 88 64 70]
["o'tdi" "o'tdi" "o'tdi" 'yiqildi' 'yiqildi' "o'tdi" "o'tdi"]
```

6. Ikki o'lchamli massivlar (matritsalar)

```
import numpy as np
m = np.array([[1, 2, 3], [4, 5, 6]])
print(m)
print("shakl:", m.shape, "elementlar:", m.size)
print(m[1, 2], m[0])
print(m[:, 1])
print(m.sum(), m.sum(axis=0), m.sum(axis=1))
print(m.T)
```

Natija:

```
[[1 2 3]
 [4 5 6]]
shakl: (2, 3) elementlar: 6
```

```
6 [1 2 3]
[2 5]
21 [5 7 9] [ 6 15]
[[1 4]
 [2 5]
 [3 6]]
```

```
import numpy as np
print(np.arange(12).reshape(3, 4))
a = np.array([[1, 2], [3, 4]])
b = np.array([[5, 6], [7, 8]])
print(a @ b)        # matritsalar ko'paytmasi
```

Natija:

```
[[ 0  1  2  3]
 [ 4  5  6  7]
 [ 8  9 10 11]]
[[19 22]
 [43 50]]
```

`axis=0` — ustunlar bo'yicha (pastga), `axis=1` — qatorlar bo'yicha (yonga). `@` — matritsa ko'paytmasi, `*` esa elementma-element.

7. Tasodifiy sonlar

```
import numpy as np
rng = np.random.default_rng(2026)
print(rng.integers(1, 7, size=10))
print(rng.normal(170, 8, size=4).round(1))
```

Natija:

```
[6 2 1 4 3 3 1 3 4 3]
[167.7 167.5 172.4 167.9]
```

8. Nega NumPy tez?

```
import time
import numpy as np
n = 1_000_000
royxat = list(range(n))
t = time.perf_counter(); s1 = sum(x * x for x in royxat); t1 = time.perf_count
massiv = np.arange(n, dtype=np.int64)
t = time.perf_counter(); s2 = int((massiv * massiv).sum()); t2 = time.perf_cou
print(s1 == s2, "NumPy tezroq:", t2 < t1)
```

Natija:

```
True NumPy tezroq: True
```

NumPy sikllarni C tilida, xotirada ketma-ket joylashgan sonlar ustida bajaradi — shuning uchun katta ma'lumotda o'nlab marta tez.

9. Xulosa

- `np.array`, `arange`, `linspace`, `zeros/ones` — massiv yaratish.
- Amallar siklsiz: `a + b`, `a * 2`, `np.sqrt(a)`; statistika: `sum, mean, max, argmax, std, median`.
- Filtrlash: `a[a > 70]`, `(shart1) & (shart2)`, `np.where`.
- 2D: `shape`, `m[i, j]`, `m[:, j]`, `sum(axis=…)`, `reshape`, `.T`, `@`.