NumPy Python array computing

CONCEPTS

An ndarray holds one dtype in a fixed-shape block. Whole-array expressions run in C, so replace Python loops with them.

Create array, arange, zeros
↓ `shape`
Reshape reshape, T, stack
↓ `compute`
Compute ufuncs, broadcasting
↓ `reduce`
Reduce sum, mean, argmax
↓ `store`
Save save, savez, savetxt

Views vs copies
Basic slices are views: writing to a[1:3] changes a. Fancy (list or mask) indexing returns a copy. Use a.copy() to detach.

CREATE

`np.array([[1, 2], [3, 4]])`
Array from nested lists

`np.zeros((<r>, <c>))`
Array of zeros

`np.arange(<start>, <stop>, <step>)`
Evenly spaced by step

INSPECT

`a.shape`
Tuple of axis lengths

INDEX AND SLICE

`a[<i>, <j>]`
One element

`a[<a>:<b>:<step>]`
Slice (end excluded)

`a[:, <j>]`
One column

`a[a > <x>]`
Elements matching a mask

SHAPE AND JOIN

`a.reshape(<r>, -1)`
Reshape, -1 infers one length

`a.T`
Transpose

`np.concatenate([<a>, <b>], axis=0)`
Join along an existing axis

MATH AND BROADCASTING

`a + b, a * 2, a ** 2`
Elementwise arithmetic

Broadcasting rules

Shapes	Result
`(3, 4) + (4,)`	(3, 4)
`(3, 1) + (1, 4)`	(3, 4)
`(3, 4) + (3,)`	error

Broadcasting
Shapes line up from the right. Each pair of lengths must be equal or one of them 1, which is stretched. Use a[:, None] to fix a mismatch.

REDUCTIONS

`a.sum() / a.mean()`
Total / average of everything

`a.sum(axis=0)`
Collapse rows: one value per column

LINEAR ALGEBRA

`a @ b`
Matrix product

RANDOM

`rng = np.random.default_rng(<seed>)`
Create a seeded generator

SAVE AND LOAD

`np.save("<file>.npy", <a>)`
Write one array (binary)

`np.load("<file>.npy")`
Read it back