PYTHON

NumPy

FULL COURSE

Arrays • Broadcasting • Linear Algebr



LONG VIDEO • 15 MIN

PYTHON • NUMPY FULL COURSE

Python NumPy Full Course

Arrays, broadcasting, statistics, linear algebra and real data

FREE FULL COURSE • MyWebUniversity.com



Wahidullah Lutfy
Founder, MyWebUniversity • CS Teacher

NUMPY

What You Will Learn

Arrays
Create arrays, and understand shape, dtype and dimensions.

Indexing
Slicing, boolean masks and fancy indexing.

Fast math
Vectorized operations, ufuncs and broadcasting.

Statistics
Aggregations along axes, and random data.

Linear algebra
Matrix multiplication, solving equations.

Real data
Load, analyze and save data files.

Python NumPy • Full Course

PART 1 · ARRAYS

NUMPY

Why NumPy? Lists vs Arrays

Python 3.13

```
$ pip install numpy
>>> import numpy as np
>>> np.__version__
'2.5.3'

>>> prices = [10, 20, 30]
>>> prices * 2
[10, 20, 30, 10, 20, 30]

>>> arr = np.array([10, 20, 30])
>>> arr * 2
array([20, 40, 60])
```

One type, packed tightly

An array stores numbers of one type, side by side in memory.

Vectorized

Math applies to every element at once, in fast compiled C code.

The standard nickname

Everyone writes import numpy as np. Use it too.

NUMPY

Creating Arrays

Python 3.13

```
>>> np.zeros(3), np.ones((2, 2))
(array([0., 0., 0.]), array([[1., 1.], [1., 1.]]))

>>> np.arange(0, 10, 2)
array([0, 2, 4, 6, 8])
>>> np.linspace(0, 1, 5)
array([0.  , 0.25, 0.5 , 0.75, 1.  ])

>>> m = np.array([[1, 2, 3], [4, 5, 6]])
>>> m.shape, m.ndim, m.size, m.dtype
((2, 3), 2, 6, dtype('int64'))
```

zeros, ones, full

Arrays filled with one value. Great starting points.

arange vs linspace

arange uses a step size; linspace uses a number of points.

shape, ndim, size, dtype

Rows and columns, number of dimensions, total elements, and the type.

NUMPY

Indexing and Slicing

Python 3.13

```
>>> a = np.array([5, 10, 15, 20, 25])
>>> a[0], a[-1], a[1:4]
(5, 25, array([10, 15, 20]))

>>> m = np.arange(1, 10).reshape(3, 3)
>>> m[1, 2], m[:, 0], m[0, :]
(6, array([1, 4, 7]), array([1, 2, 3]))

>>> a[a > 12]
array([15, 20, 25])
>>> a[[0, 2, 4]]
array([ 5, 15, 25])
```

Like lists, plus more

a[start:stop] works as usual; m[row, col] reaches into 2D arrays.

The colon means all

m[:, 0] is the whole first column; m[0, :] is the whole first row.

Boolean and fancy indexing

a[a > 12] filters by a condition; a[[0, 2, 4]] picks positions.

NUMPY

Reshaping, Stacking and Copies

Python 3.13

```
>>> a = np.arange(6)
>>> a.reshape(2, 3).T
array([[0, 3], [1, 4], [2, 5]])
>>> a.reshape(-1, 2).shape
(3, 2)


>>> np.vstack([[1, 2], [3, 4]]), np.concatenate([[1, 2], [3, 4]])

>>> b = a[:3]; b[0] = 99
>>> a
array([99,  1,  2,  3,  4,  5])
>>> c = a[:3].copy()
```

reshape and -1

Same data, new shape. -1 means: work this number out for me.

.T transposes

Rows become columns. vstack, hstack and concatenate join arrays.

Views vs copies

A slice is a view of the same memory. Use .copy() when you need independence.

Python NumPy • Full Course

PART 2 · FAST MATH

NUMPY

Vectorized Math and Universal Functions

Python 3.13

```
>>> a = np.array([1, 4, 9, 16])
>>> a + 10, a * a, a / 2
>>> np.sqrt(a)
array([1., 2., 3., 4.])

>>> np.round(np.sin(np.pi / 2), 3), np.exp(1)
(1.0, 2.718281828459045)

>>> np.where(a > 5, "big", "small")
array(['small', 'small', 'big', 'big'], dtype='<U5')
```

Element by element

+, -, *, /, ** and comparisons all work on whole arrays.

Universal functions

np.sqrt, np.exp, np.log, np.sin, np.abs... fast and vectorized.

np.where

A vectorized if-else: choose values based on a condition.

NUMPY

Broadcasting

Python 3.13

```
>>> scores = np.array([[80, 90], [70, 60], [95, 85]])   # 3 students, 2 exams
>>> curve = np.array([5, 10])
>>> scores + curve
array([[ 85, 100], [ 75,  70], [100,  95]])

>>> scores - scores.mean(axis=0)
array([[ -1.67,  11.67], [-11.67, -18.33], [ 13.33,   6.67]])
```

Stretching shapes

A smaller array is stretched to match a bigger one, without copying data.

The rule

Compare shapes from the right: sizes must be equal, or one of them must be 1.

Everyday use

Curving scores, normalizing columns, converting units for a whole table.

NUMPY

Statistics and Axes

Python 3.13

```
>>> sales = np.array([[120, 80, 95], [130, 90, 100]])   # 2 stores x 3 days
>>> sales.sum(), sales.mean().round(1)
(615, 102.5)

>>> sales.sum(axis=0)    # per day
array([250, 170, 195])
>>> sales.sum(axis=1)    # per store
array([295, 320])

>>> sales.max(), sales.argmax(), np.median(sales), sales.std().round(2)
```

Aggregations

sum, mean, median, min, max, std, var, argmin, argmax, cumsum.

axis=0

Collapses the rows: one result per column (down).

axis=1

Collapses the columns: one result per row (across).

Python NumPy • Full Course

PART 3 · RANDOM, LINEAR ALGEBRA AND REAL DATA

NUMPY

Random Numbers

Python 3.13

```
>>> rng = np.random.default_rng(42)
>>> rng.integers(1, 7, size=5)        # dice
array([1, 5, 4, 3, 3])

>>> heights = rng.normal(170, 8, size=1000)
>>> heights.mean().round(1), (heights > 185).mean()
(169.8, 0.026)

>>> rng.choice(["red", "green", "blue"], size=3), rng.permutation(5)
```

default_rng(seed)

The modern random generator. A seed makes results repeatable.

Distributions

integers, random, normal, uniform, choice, permutation, shuffle.

Simulations

The mean of a boolean array is the fraction of True values.

NUMPY

Linear Algebra

Python 3.13

```
>>> A = np.array([[2, 1], [1, 3]])
>>> B = np.array([[1, 0], [0, 1]])
>>> A @ B
array([[2, 1], [1, 3]])

# Solve:  2x + y = 5   and   x + 3y = 10
>>> np.linalg.solve(A, np.array([5, 10]))
array([1., 3.])

>>> np.linalg.det(A).round(2), np.linalg.inv(A).round(2)
```

@ means matrix multiply

A * B multiplies element by element; A @ B is true matrix multiplication.

np.linalg

solve, inv, det, eig, norm: the math behind machine learning.

Prefer solve over inv

np.linalg.solve is faster and more accurate than inverting a matrix.

NUMPY

Mini Project: Analyze Temperature Data

weather.py — Python 3.13

```
import numpy as np
data = np.loadtxt("temps.csv", delimiter=",", skiprows=1)   # day, city1, city2
temps = data[:, 1:]

print("Average per city:", temps.mean(axis=0).round(1))
hot_days = (temps > 30).sum(axis=0)
print("Days above 30C:", hot_days)
temps_f = temps * 9 / 5 + 32

np.savetxt("temps_f.csv", temps_f, delimiter=",", fmt="%.1f")
np.save("temps.npy", temps)   # fast binary format
```

Load

np.loadtxt and np.genfromtxt read text files; genfromtxt handles missing values.

Analyze

Slicing, axis statistics and boolean counts, all without a single loop.

Save

savetxt for CSV; np.save and np.load for fast binary .npy files.

Recap and Practice

Remember

Arrays are fast, typed and vectorized. Think in whole arrays, not loops.

Try it yourself

Simulate 10,000 dice rolls and find the percentage of sixes.

Watch out

Slices are views. Check shapes before you broadcast.