Python Notes

Ahmed Yasser

2025

Contents

Core Concepts

- **ndarray**: NumPy's main data structure for multi-dimensional arrays
- **dtype**: Data type of array elements (e.g., np.int64, np.float32)
- **axes**: Dimensions of an array
- **shape**: Tuple indicating size along each dimension
- **views vs. copies**: Views share memory with original array; copies are independent

Array Creation

np.array(sequence)	Create array from Python sequences, Copy
np.fromiter(generator, dtype)	Create array from generators/iterables, Copy
np.ones(shape)	Create array filled with ones, Copy
np.zeros(shape)	Create array filled with zeros, Copy
np.empty(shape)	Create uninitialized array, Copy
np.eye(n)	Create identity matrix, Copy
np.diag(v)	Create diagonal array from sequence, Copy
np.arange(start, stop, step)	Create array with evenly spaced values, Copy
np.linspace(start, stop, num)	Create array with num evenly spaced values, Copy

Array Attributes

arr.ndim	Number of dimensions
arr.shape	Tuple of array dimensions
arr.size	Total number of elements
arr.dtype	Data type of elements
arr.itemsize	Size of each element in bytes

Type Conversion

arr.astype(dtype)	Convert array to different data type, Copy

Indexing and Slicing

- Basic indexing with integers and slices (**returns views**)
  - arr[row, col]: Access specific element
  - arr[start:stop:step]: Create slice
- Fancy indexing with lists or arrays of indices (**returns copy**)
- Boolean masking: arr[mask] where mask is boolean array (**returns copy**)

Array Manipulation

arr.reshape(shape) or np.reshape(arr, shape)	Change array shape, View
arr.ravel() or np.ravel(arr)	Flatten array, View
arr.flatten()	Flatten array, Copy
np.concatenate((arr1, arr2), axis=0)	Join arrays, Copy

arr.T or arr.transpose() or np.transpose(arr)	Transpose array, View
arr[:, np.newaxis] or arr[:, None]	Add new axis, View

Mathematical Operations

- Universal functions (ufuncs): Element-wise operations

np.add(arr1, arr2) or arr1 + arr2	Element-wise addition, Copy
np.subtract(arr1, arr2) or arr1 - arr2	Element-wise subtraction, Copy
np.multiply(arr1, arr2) or arr1 * arr2	Element-wise multiplication, Copy
np.divide(arr1, arr2) or arr1 / arr2	Element-wise division, Copy
np.exp(arr)	Element-wise exponential, Copy
np.log(arr), np.log10(arr), np.sqrt(arr)	Math functions, Copy

Reduction Operations

np.add.reduce(arr, axis)	Reduce using addition along axis, Copy
arr.sum(axis) or np.sum(arr, axis)	Sum of elements, Copy
arr.mean(axis) or np.mean(arr, axis)	Mean of elements, Copy
arr.std(axis) or np.std(arr, axis)	Standard deviation, Copy
arr.var(axis) or np.var(arr, axis)	Variance, Copy
arr.min() or np.min(arr)	Minimum value, Copy
arr.max() or np.max(arr)	Maximum value, Copy
arr.argmin() or np.argmin(arr)	Index of minimum value, Copy
arr.argmax() or np.argmax(arr)	Index of maximum value, Copy

Broadcasting

Broadcasting allows NumPy to perform element-wise operations between arrays of different shapes, **without copying data**.

NumPy Broadcasting Rules

To determine whether two shapes can be broadcast together:

1. Align shapes from the right (last dimension) to the left (first dimension). If one shape is shorter, pad it with 1s on the left.
   - For example, if you have shapes (3, 4) and (4,), they can be aligned as (3, 4) and (1, 4).
2. For each dimension, they are compatible if:
   - The dimensions are equal, or
   - One of them is 1

If all dimensions meet one of these conditions, the shapes can be broadcast.

Comparison and Boolean Operations

arr > value, arr == value, etc.	Create boolean mask, Copy
(condition1) & (condition2)	Logical AND, Copy
(condition1) | (condition2)	Logical OR, Copy
~(condition)	Logical NOT, Copy
mask.nonzero() or np.nonzero(mask)	Get indices of True elements, Copy
np.where(condition)	Get indices where condition is True, Copy
np.where(condition, x, y)	Return x where True, y where False, Copy

Random Number Generation

np.random.seed(seed)	Set random seed for reproducibility, NorA
np.random.RandomState(seed)	Create separate random generator, NorA
np.random.rand(shape) or rng.rand(shape)	Uniform distribution [0,1), Copy
np.random.randn(shape) or rng.randn(shape)	Standard normal distribution, Copy

Linear Algebra

np.matmul(a, b) or a @ b	Matrix multiplication, Copy
np.dot(a, b)	Dot productormatrix multiplication, Copy

Set Operations

np.unique(arr)	Get unique elements, Copy
np.intersect1d(arr1, arr2)	Set intersection, Copy
np.union1d(arr1, arr2)	Set union, Copy
np.setdiff1d(arr1, arr2)	Set difference, Copy

Serialization

np.save('file.npy', arr)	Save single array, NorA
np.savez('file.npz', arr1, arr2)	Save multiple arrays, NorA
np.savez('file.npz', key1=arr1, key2=arr2)	Save with custom keys, NorA
np.load('file.npy') or np.load('file.npz')	Load saved arrays, Copy

Memory Management

- np.may_share_memory(arr1, arr2): Check if arrays share memory
- Basic slicing and indexing create views (memory efficient)
- Advancedorfancy indexing creates copies
- Mathematical operations generally create copies