FREE EDITION · NOTES + 3 SAMPLE MCQS

CUET · COMPUTER SCIENCE · CLASS XI · CODE 308

Introduction to NumPy

CUET unit: Introduction to NumPy

By **UniDrill** · NCERT-grounded study material

WWW.UNIDRILL.IN

📌 Snapshot

- NumPy (Numerical Python) is the foundational library for scientific computing and data analysis in Python, built on an n-dimensional array object called ndarray.
- A NumPy array is a homogeneous, contiguously stored data structure; unlike Python lists, arrays are faster and more memory-efficient.
- A NumPy array has a full lifecycle: creation (from lists, zeros, ones, arange), attribute inspection (ndim, shape, size, dtype, itemsize), indexing/slicing, arithmetic and statistical operations, reshaping, concatenation, splitting, and file I/O.
- NumPy underpins all data-handling topics in the syllabus; CUET questions probe array creation syntax, attribute values, slicing notation, element-wise vs. matrix operations, and statistical functions.
- The case study (Iris dataset) demonstrates real-world application of loading, splitting, and analysing 2-D arrays from CSV files, a pattern directly tested in applied MCQs.

📖 Detailed Notes

2.1 Core concepts

- **NumPy and ndarray:** NumPy stands for "Numerical Python". It is a package for data analysis and scientific computing that uses a multidimensional array object (officially called `ndarray`, commonly called array) as its core data structure. NumPy can be interfaced with C, C++ and other Python packages. Install using `pip install NumPy`; import with `import numpy as np`. (NCERT §6.1, p. 95)
- **Array definition and characteristics:** An array is a data type that stores multiple values of the same data type under a single identifier. Elements are stored contiguously in memory (making operations fast), each element has a unique integer index starting from 0 (zero-based indexing), and the entire array occupies a fixed memory block. (NCERT §6.2, p. 96)
- **List vs. Array:** A Python list can hold elements of different data types and is not stored contiguously; it also stores type information for each element separately, making it less memory-efficient. A NumPy array holds elements of the same data

type, supports element-wise arithmetic operations (e.g., `A1/3` divides every element by 3), and is part of the NumPy library — not core Python. (NCERT §6.3.1, p. 97)

- **Creating arrays from lists:** `np.array()` converts a Python list or nested list into an ndarray. A 1-D array holds a single row of elements; a 2-D array is created from nested lists. If a list contains mixed types (e.g., int and string), all values are promoted to the broadest type (e.g., all become strings with dtype `<U32`). Integers in a nested list passed alongside floats are promoted to floats. (NCERT §6.3.2, p. 97–98)
- **ndarray attributes — ndim:** `ndarray.ndim` returns the number of dimensions (axes/rank) as an integer. A 1-D array has ndim=1; a 2-D array has ndim=2. The row-axis of a 2-D array is axis-0 and the column-axis is axis-1. (NCERT §6.3.3 (i), p. 98)
- **ndarray attributes — shape:** `ndarray.shape` returns a tuple giving the size along each dimension. For a 1-D array of 3 elements, shape is `(3,)`; for a 2-D array of 3 rows and 2 columns, shape is `(3, 2)`. (NCERT §6.3.3 (ii), p. 98–99)
- **ndarray attributes — size:** `ndarray.size` returns the total number of elements, equal to the product of all values in the shape tuple. (NCERT §6.3.3 (iii), p. 99)
- **ndarray attributes — dtype:** `ndarray.dtype` gives the data type of all elements (e.g., `int32`, `float64`, `<U32`). All elements in an array share the same dtype. (NCERT §6.3.3 (iv), p. 99)
- **ndarray attributes — itemsize:** `ndarray.itemsize` specifies the size in bytes of each element. int32/float32 → 4 bytes; int64/float64 → 8 bytes; strings (U32) → 128 bytes. (NCERT §6.3.3 (v), p. 99)
- **Other creation methods:** (1) `dtype` argument in `np.array()` forces a specific type (e.g., `dtype=float` converts integers to floats). (2) `np.zeros((r,c))` creates an array of shape (r,c) with all elements 0.0 (default float). (3) `np.ones((r,c))` creates an array of all 1.0 (default float). (4) `np.arange(start, stop, step)` creates a 1-D array analogous to Python's `range()`. (NCERT §6.3.4, p. 99–100)
- **Indexing:** For 1-D arrays, zero-based integer indexing applies. For 2-D arrays, each element is referenced by two indices `[i, j]` where `i` is the row (0-based) and `j` is the column (0-based). Accessing an out-of-range index raises an `IndexError`. (NCERT §6.4.1, p. 100–101)
- **Slicing:** Extract a sub-array using `[start:end]` (end index is excluded). Omitting start/end defaults to first/last element. `[::-1]` reverses the array. For 2-D arrays, slicing on both axes uses `[row_start:row_end, col_start:col_end]`; omitting an axis means all elements along that axis are included. (NCERT §6.4.2, p. 101–102)
- **Arithmetic operations:** When the same arithmetic operator (+, -, , /, %, *) is applied to two arrays of the same shape, the operation is performed element-wise (on corresponding pairs). For element-wise operations, both arrays must have the same shape. The `@` operator performs matrix multiplication. (NCERT §6.5.1, p. 102–103)

- **Transpose:** `ndarray.transpose()` (or `.T`) swaps rows and columns, turning a (3, 4) array into a (4, 3) array. The original array is not modified. (NCERT §6.5.2, p. 103–104)
- **Sorting:** `ndarray.sort()` sorts elements in ascending order by default. For 2-D arrays, `sort()` with default axis=1 sorts row-wise (each row sorted independently); `sort(axis=0)` sorts column-wise (each column sorted independently). (NCERT §6.5.3, p. 104)
- **Concatenation:** `np.concatenate((arr1, arr2), axis)` joins arrays. By default axis=0 (row-wise). For concatenation along axis=1 (column-wise), the number of rows must match; for axis=0, the number of columns must match. A dimension mismatch raises a `ValueError`. (NCERT §6.6, p. 104–105)
- **Reshaping:** `ndarray.reshape(r, c)` changes the shape without changing the total number of elements. Attempting to change the element count via reshape raises an error. (NCERT §6.7, p. 105–106)
- **Splitting:** `numpy.split(array, indices_or_sections, axis)` splits an array into sub-arrays. The parameter can be a list of split indices or an integer N (split into N equal parts). Default axis=0 splits row-wise. (NCERT §6.8, p. 106–107)
- **Statistical operations:** NumPy provides `max()`, `min()`, `sum()`, `mean()`, `std()` on arrays. Each can be applied to the whole array or along a specific axis. `axis=0` operates column-wise; `axis=1` operates row-wise. (NCERT §6.9, p. 107–109)
- **Loading from files — loadtxt():** `np.loadtxt(filename, skiprows, delimiter, dtype, unpack)` loads data from a text/CSV file. `skiprows=1` skips the header row. Default delimiter is space; default dtype is float. When `unpack=True`, the returned array is transposed so columns become separate arrays. (NCERT §6.10.1, p. 109–110)
- **Loading from files — genfromtxt():** `np.genfromtxt(filename, skip_header, delimiter, dtype, filling_values)` also handles missing values and non-numeric strings, converting them to `nan` (or `-1` if dtype is int, or a custom value via `filling_values`). (NCERT §6.10.2, p. 111)
- **Saving to files:** `np.savetxt(filename, array, delimiter, fmt)` saves a NumPy array to a text file. The `fmt` parameter specifies the format (e.g., `'%i'` for integers). Default format is float. (NCERT §6.11, p. 112)

2.2 Definitions to memorise

Term	Definition	Page
NumPy	Numerical Python; a Python library for scientific computing using n-dimensional array objects	95
ndarray	The official name of the NumPy array class; an n-dimensional homogeneous array stored contiguously in memory	97
Array		96

Term	Definition	Page
	A data type storing multiple values of the same data type using a single identifier, with zero-based indexing	
ndim	Attribute giving the number of dimensions (rank/axes) of an ndarray	98
shape	Attribute returning a tuple of integers indicating the size of the array along each dimension	98
size	Attribute giving the total number of elements in the array (product of shape values)	99
dtype	Attribute giving the data type of the array's elements (e.g., int32, float64, U32)	99
itemsize	Attribute giving the size in bytes of each element of the array	99
axis-0	The row-axis of a 2-D NumPy array; operations along axis=0 work column-wise	98
axis-1	The column-axis of a 2-D NumPy array; operations along axis=1 work row-wise	98
Zero-based indexing	The first element of an array has index 0, the second index 1, and so on	96
Contiguous memory	Array elements stored in adjacent memory locations, enabling fast operations	96
CSV	Comma Separated Values; a common text file format for tabular data that can be loaded into NumPy arrays	109
nan	Not a Number; the value genfromtxt() assigns to missing or non-numeric entries in float arrays	111
reshape()	Function that changes the shape of an array without altering the total number of elements	105
transpose()	Operation that swaps rows and columns of a 2-D array; does not modify the original array	103
`np.array()`	Function that creates an ndarray from a Python list or nested list	97
`np.zeros(shape)`	Creates an array of given shape filled with 0.0 (default float)	100
`np.ones(shape)`	Creates an array of given shape filled with 1.0 (default float)	100
`np.arange(a, b, s)`	Creates a 1-D array from a to b (exclusive) with step s	100
`np.concatenate`	Joins two or more arrays along the specified axis	104
`np.split`	Splits an array into multiple sub-arrays	106
`np.loadtxt`	Loads numeric data from a text/CSV file into an ndarray	109
`np.genfromtxt`	Like loadtxt but tolerates missing values (fills with nan/-1)	111

Term	Definition	Page
`np.savetxt`	Saves an ndarray to a text/CSV file with optional format string	112
Element-wise operation	Arithmetic operation applied to corresponding pairs of two same-shape arrays	103
Matrix multiplication	Linear-algebra multiplication using the `@` operator	103
Pip	Python package installer used to install NumPy via `pip install numpy`	95
`ndarray.sum()`	Computes the sum across the whole array or along an axis	107

2.3 Diagrams / processes to remember

- **Contiguous vs. non-contiguous memory:** The sidebar on p. 96 contrasts the two allocation strategies. In non-contiguous (block) allocation, data is divided into fixed-size blocks placed anywhere in memory. In contiguous allocation, data occupies one unbroken memory region — the reason NumPy operations are faster than list operations.
- **Table 6.1 — List vs. Array comparison:** A five-row table on p. 97 systematically contrasts lists and arrays across: data type flexibility, memory storage, element-wise operations, memory efficiency, and library membership. This table is a frequent source of MCQ distractors.
- **Table 6.1 (Marks of students) — 2-D array indexing:** The 4×3 marks matrix on p. 101 illustrates that `marks[i,j]` refers to the element at row (i+1), column (j+1). The example `marks[3,1]` = 72 (Prasad's English marks) is a direct illustration of zero-based 2-D indexing.
- **Figure: axis-0 and axis-1 in 2-D sorting (p. 104):** `sort()` default (axis=1) sorts each row; `sort(axis=0)` sorts each column. Students should be able to predict the output of both calls on a given 2-D array.
- **Concatenation error scenario (p. 105):** Concatenating a (2,2) array with a (2,3) array along axis=0 raises a `ValueError` because the column counts differ. Concatenating along axis=1 succeeds and produces a (2,5) result. This exact scenario appears as a trap question.

2.4 Common confusions / NTA trap points

- **axis=0 means column-wise, NOT row-wise:** When statistical functions (`sum` , `mean` , `max` , `std`) are called with `axis=0` , they collapse along rows and produce one result per column. Students often reverse this. `axis=1` collapses along columns, giving one result per row. (NCERT §6.9, p. 108)
- **sort() default axis in 2-D is axis=1 (row-wise), NOT axis=0:** Unlike statistical functions where the default is the whole array, `sort()` on a 2-D array defaults to axis=1, sorting elements within each row. NTA may present code without an explicit axis and ask for output. (NCERT §6.5.3, p. 104)

- **reshape() cannot change element count:** A common distractor asks what happens when `np.arange(12).reshape(4,4)` is called — this raises an error because 12 elements cannot fill a 4×4 (16-element) grid. (NCERT §6.7, p. 105)
- **genfromtxt() vs. loadtxt() for missing data:** `loadtxt()` crashes on missing values; `genfromtxt()` handles them by substituting `nan` (float) or `-1` (int) or a custom `filling_values`. Confusing the two functions is a trap. (NCERT §6.10.2, p. 111)
- **Type promotion in mixed-type lists (NCERT §6.3.2, p. 97-98).** Passing `[5, -7.4, 'a', 7.2]` to `np.array()` promotes all elements to string (U32), not float.
- **`@` ≠ `*` for matrices (NCERT §6.5.1, p. 103).** `A * B` is element-wise; `A @ B` is matrix multiplication. NTA distractor swaps them.
- **np.arange stop is exclusive (NCERT §6.3.4, p. 100).** Same as Python's `range`.
- **Transpose does not modify original (NCERT §6.5.2, p. 103-104).** It returns a new view. NTA may suggest in-place mutation.
- **Concatenation axis must match (NCERT §6.6, p. 104).** Axis=0 requires equal column counts; axis=1 requires equal row counts. Mismatch → ValueError.
- **`ndarray.size` ≠ `ndarray.shape` (NCERT §6.3.3, p. 99).** size is total elements (int); shape is the tuple.

🎯 Practice MCQs

Q1. Which of the following statements correctly describes the default data type of an array created by `np.zeros((3, 4))`?

A. int32

B. int64

C. float64

D. bool

Q2. Consider the following code: ```python import numpy as np arr = np.array([[1, 2, 3], [4, 5, 6], [7, 8, 9]]) print(arr[1:3, 0:2]) ``` What will be the output?

A. `array([[1, 2], [4, 5]])`

B. `array([[4, 5], [7, 8]])`

C. `array([[2, 3], [5, 6]])`

D. `array([[4, 5, 6], [7, 8, 9]])`

Q3. Assertion (A): For element-wise arithmetic operations on two NumPy arrays, the shape of both arrays must be the same. Reason (R): NumPy performs arithmetic operations on each corresponding pair of elements.

A. Both A and R are true, and R is the correct explanation of A.

B. Both A and R are true, but R is NOT the correct explanation of A.

C. A is true but R is false.

D. A is false but R is true.

📊 PYQ Alignment

NumPy appears consistently in CUET Computer Science papers as part of the data-handling and Python programming units; questions typically focus on NumPy array creation syntax (zeros, ones, arange), attribute values (ndim, shape, size, dtype, itemsize), slicing notation output prediction, the distinction between element-wise (`*`) and matrix (`@`) multiplication, and the difference between `loadtxt()` and `genfromtxt()` for handling missing data. See PYQ archive for Computer Science.