numpy all concept

Here is **NumPy — All Concepts (Beginner → Advanced)** explained in a clean, structured, exam + interview–friendly way.
If you want, I can also create a **PDF**, **notes**, or **full cheat-sheet**.

🔥 NumPy – Complete Concepts (A to Z)

NumPy = *Numerical Python*, the backbone of scientific computing, ML, data analysis, and linear algebra.

---

1. Introduction to NumPy

✔ What is NumPy?

- A Python library for fast numerical computing.
- Uses **ndarray** (N-dimensional array).
- Much faster than Python lists due to:
  - Fixed data types
  - Contiguous memory
  - Vectorization (no loop overhead)
  - Internal C-optimized implementation

---

2. NumPy Arrays (ndarray)

✔ Creating Arrays

python Copy code

```python
import numpy as np

arr = np.array([1, 2, 3])
arr2 = np.array([[1,2,3],[4,5,6]])
np.zeros(5)           # array of zeros
np.ones((2,3))        # 2x3 ones
np.full((3,3), 7)     # constant array
np.eye(4)             # identity matrix
```

```
np.arange(1, 10, 2)   # step array
np.linspace(0, 1, 5)  # 5 equal points
```

---

3. Array Attributes

python Copy code

```
arr.ndim      # dimension
arr.shape     # rows, cols
arr.size      # total elements
arr.dtype     # data type
arr.itemsize  # size of each element
arr.nbytes    # total bytes
```

---

4. Data Types in NumPy

- `int32` , `int64`
- `float32` , `float64`
- `complex`
- `bool`
- `string_`
- `object`

Convert dtype:

python Copy code

```
arr.astype('float')
```

---

5. Indexing & Slicing

✔ Basic Indexing

python Copy code

```
arr[0]
arr[1:4]
```

```
arr[:, 1]       # column
arr[0:2, 1:3]   # subarray
```

✔ Boolean Indexing

python Copy code

```
arr[arr > 10]
```

✔ Fancy Indexing

python Copy code

```
arr[[0,2,4]]
```

---

6. Array Operations (Vectorization)

✔ Arithmetic

python Copy code

```
arr + 5
arr * 2
arr1 + arr2
arr1 * arr2
```

✔ Mathematical Functions

python Copy code

```
np.sqrt(arr)
np.log(arr)
np.exp(arr)
np.sin(arr)
np.sum(arr)
np.mean(arr)
np.median(arr)
np.std(arr)
```

---

7. Reshaping & Transpose

python Copy code

```python
arr.reshape(3, 3)
arr.ravel()         # 1D flatten
arr.flatten()
arr.T               # transpose
arr.reshape(-1)     # auto calculate dimension
```

---

8. Stacking

✔ Horizontal (columns)

python Copy code

```python
np.hstack((a,b))
```

✔ Vertical (rows)

python Copy code

```python
np.vstack((a,b))
```

✔ Column Stack

python Copy code

```python
np.column_stack((a,b))
```

---

9. Splitting

python Copy code

```python
np.split(arr, 3)
np.hsplit(arr, 3)
np.vsplit(arr, 2)
```

---

10. Broadcasting

Rules:

1. If shapes differ, prepend 1
2. Compare dimensions from last
3. Dimensions must be:
   - equal
   - OR one of them must be 1

Example:

python Copy code

```
a = np.array([[1,2,3],[4,5,6]])
b = np.array([10,20,30])
a + b   # row-wise broadcasting
```

---

11. Copies vs Views

✔ View (Shallow Copy)

python Copy code

```
b = a.view()
```

(b reflects changes)

✔ Copy (Deep Copy)

python Copy code

```
b = a.copy()
```

(b independent)

---

12. Random Module

✔ Generate random values

python Copy code

```
np.random.rand(3,3)
np.random.randn(3,3)
np.random.randint(1, 10, 5)
```

✔ Important

python Copy code

```
np.random.seed(42)
```

---

13. Linear Algebra (np.linalg)

Used in ML: PCA, Regression, Neural Networks.

python Copy code

```
np.dot(a,b)
np.matmul(a,b)
np.linalg.inv(a)       # inverse
np.linalg.det(a)       # determinant
np.linalg.eig(a)       # eigenvalues/vectors
np.linalg.norm(a)      # vector magnitude
```

---

14. Statistics

python Copy code

```
np.mean(a)
np.var(a)
np.std(a)
np.min(a)
np.max(a)
np.percentile(a, 90)
```

---

15. Sorting

python Copy code

```python
np.sort(arr)
arr.sort()        # in place
np.argsort(arr)
```

---

16. File Handling

✔ Save NumPy arrays

python Copy code

```python
np.save('file.npy', arr)
np.savez('file.npz', a=arr1, b=arr2)
```

✔ Load

python Copy code

```python
np.load('file.npy')
```

✔ CSV

python Copy code

```python
np.savetxt('data.csv', arr, delimiter=',')
np.loadtxt('data.csv', delimiter=',')
```

---

17. Aggregation with Axis

python Copy code

```python
np.sum(arr, axis=0)   # column sum
np.sum(arr, axis=1)   # row sum
```

---

18. Where, Unique, Argmax

python Copy code

```
np.where(arr > 10)
np.unique(arr)
arr.argmax()          # max index
arr.argmin()
```

---

19. Masked Arrays

python Copy code

```
import numpy.ma as ma
masked = ma.masked_array(arr, mask=[0,1,0,1])
```

---

20. Matrix Operations

python Copy code

```
A @ B           # matrix multiplication
A.T             # transpose
np.trace(A)
```

---

21. Important Concepts in ML with NumPy

- Vectorization
- Broadcasting
- Dot product
- Normalization
- Distance calculation
- Covariance matrix
- Eigen decomposition

  Used in: KMeans, PCA, Linear Regression, Neural Nets.

---

22. Performance Tips

- Use vectorized operations instead of loops

- Use `dtype=float32` to save memory
- Use slicing, avoid copying
- Use boolean indexing instead of loops

---

23. Interview-Level Questions

1. Difference: Python list vs NumPy array
2. What is Broadcasting?
3. Deep copy vs View
4. Why NumPy is faster?
5. Explain vectorization
6. ndArray vs Matrix
7. How does stride work?

⭐ Want More?

I can give you:

📘 **NumPy Notes (PDF)**
💯 **NumPy Interview Question Bank (100 Q)**
🖥️ **NumPy Cheat Sheet**
📦 **NumPy practice problems + solutions**
🧪 **NumPy for Machine Learning (full guide)**

Just tell me **which one you want**.

numpy arrays all concept

Below is **NumPy Arrays – ALL Concepts (Beginner → Advanced)** explained in a clean, structured way.
This is **only about Arrays (ndarray) —** their creation, shape, indexing, slicing, operations, broadcasting, views, memory, stride, advanced indexing, etc.

If you want, I can make **PDF Notes**, **Cheat Sheet**, or **Practice Questions**.

⭐ NUMPY ARRAYS – COMPLETE CONCEPTS

(Everything about `ndarray`)

1️⃣ What is `ndarray`?

- Core data structure of NumPy.
- Multi-dimensional, homogeneous array.
- Fast because:
  - Fixed data type
  - Stored in contiguous memory
  - C/C++ optimized
  - Vectorized operations

2️⃣ Creating Arrays

✔ From Python list/tuple

python Copy code

```
np.array([1,2,3])
np.array([[1,2],[3,4]])
```

✔ Predefined arrays

python Copy code

```
np.zeros((2,3))
np.ones(5)
np.full((3,3), 7)
np.empty((3,3))   # uninitialized
np.arange(1, 10, 2)
np.linspace(0, 1, 5)
np.eye(4)         # identity
```

✔ Random arrays

python Copy code

```
np.random.rand(3,3)
np.random.randint(1,10,(2,3))
np.random.randn(5)
```

---

3 Array Attributes

python Copy code

```
arr.ndim      # number of dimensions
arr.shape     # (rows, columns)
arr.size      # total elements
arr.dtype     # data type
arr.itemsize  # bytes per element
arr.nbytes    # total bytes
arr.strides   # step to move to next element
```

---

4 Dimensions (ndim)

- **0D** → Scalar
- **1D** → Vector
- **2D** → Matrix
- **3D** → Tensor
- **ND** → Higher-dimensional tensor

Example:

python Copy code

```
a = np.array(10)            # 0D
b = np.array([1,2,3])       # 1D
c = np.array([[1,2],[3,4]]) # 2D
d = np.arange(24).reshape(2,3,4)  # 3D
```

---

5 Indexing

✔ 1D indexing

python

```
arr[0], arr[-1], arr[2]
```

✔ 2D indexing

python Copy code

```
arr[0,1]     # row 0, column 1
arr[1, :]    # entire row
arr[:, 2]    # entire column
```

✔ ND indexing

python Copy code

```
arr[1, 2, 3]   # element at (1,2,3)
```

---

6 Slicing

✔ Basic slicing

python Copy code

```
arr[1:5]
arr[:, 1:3]
arr[0:2, 1:3]
```

✔ Step slicing

python Copy code

```
arr[::2]
arr[::-1]   # reverse
```

---

7 Advanced Indexing

✔ Fancy Indexing

python Copy code

```
arr[[0,2,4]]     # pick rows
arr[:, [0,2]]    # pick columns
```

✔ Boolean Indexing

python Copy code

```
arr[arr > 10]
arr[(arr > 5) & (arr < 20)]
```

---

8 Copy vs View

✔ View (Shallow copy)

python Copy code

```
b = a.view()
```

- Shares data
- Changes reflect in both

✔ Copy (Deep copy)

python Copy code

```
b = a.copy()
```

- Independent array
- No shared memory

---

9 Broadcasting (Very Important)

✔ Broadcast rules

1. Compare shapes from **right to left**
2. Dimensions must be:
   - equal, or
   - one of them is 1

Example:

python Copy code

```
a = np.array([[1,2,3],[4,5,6]])
b = np.array([10,20,30])

a + b   # works → row broadcasting
```

Another:

python Copy code

```
scalar + array
```

Always works.

🔟 Shape Manipulation

✔ Reshape

python Copy code

```
arr.reshape(3, 4)
arr.reshape(-1)      # auto dimension
```

✔ Flattening

python Copy code

```
arr.ravel()          # view (memory-safe)
arr.flatten()        # copy
```

✔ Transpose

python Copy code

```
arr.T
np.transpose(arr)
```

✔ Swap axes

python Copy code

```
np.swapaxes(arr, 0, 1)
```

1️⃣1️⃣ Stacking Arrays

✔️ Vertical Stack (row-wise)

python Copy code

```
np.vstack((a,b))
```

✔️ Horizontal Stack (column-wise)

python Copy code

```
np.hstack((a,b))
```

✔️ Depth Stack

python Copy code

```
np.dstack((a,b))
```

1️⃣2️⃣ Splitting Arrays

python Copy code

```
np.split(arr, 3)
np.hsplit(arr, 2)
np.vsplit(arr, 4)
```

1️⃣3️⃣ Array Math Operations

✔️ Element-wise

python Copy code

```
arr1 + arr2
arr1 * arr2
arr ** 2
arr / 10
```

✔ Reduction operations

python Copy code

```
np.sum(arr)
np.mean(arr)
np.min(arr)
np.max(arr)
np.std(arr)
np.var(arr)
```

✔ Axis-based

python Copy code

```
np.sum(arr, axis=0)   # column-wise
np.sum(arr, axis=1)   # row-wise
```

1️⃣4️⃣ Logical Operations

python Copy code

```
arr == 10
arr > 5
np.any(arr > 10)
np.all(arr < 100)
```

1️⃣5️⃣ Sorting

python Copy code

```
np.sort(arr)
arr.sort()          # in-place
```

```
np.argsort(arr)
```

1️⃣6️⃣ Linear Algebra with Arrays

python Copy code

```
np.dot(a, b)
a @ b    # matrix multiply
np.linalg.inv(a)
np.linalg.det(a)
np.trace(a)
np.linalg.eig(a)
np.linalg.norm(a)
```

1️⃣7️⃣ Array Iteration

✔ Single iterator

python Copy code

```
for x in np.nditer(arr):
    print(x)
```

✔ With indices

python Copy code

```
for index, value in np.ndenumerate(arr):
    print(index, value)
```

1️⃣8️⃣ Memory Layout (Advanced)

✔ Row-major (C-order) vs Column-major (F-order)

python Copy code

```
arr = np.array([[1,2],[3,4]], order='C')
```

```
arr_f = np.array([[1,2],[3,4]], order='F')
```

✔ Strides

Distance in bytes to move each step.

python Copy code

```
arr.strides
```

---

1 9 Missing Data – Masked Arrays

python Copy code

```
import numpy.ma as ma
ma.masked_array(data, mask=[0,1,0])
```

---

2 0 Array File Operations

✔ Save

python Copy code

```
np.save('arr.npy', arr)
np.savetxt('file.csv', arr, delimiter=',')
```

✔ Load

python Copy code

```
np.load('arr.npy')
np.loadtxt('file.csv', delimiter=',')
```

⭐ Want More?

I can give you any of these:

📘 Numpy Arrays Full Notes (PDF)