
🧠 10 Theoretical NumPy Questions and Answers

1. What is NumPy?

Answer:
NumPy (Numerical Python) is a Python library used for numerical computations. It provides a powerful **n-dimensional array object** and tools for performing operations like linear algebra, statistics, and Fourier transforms efficiently.

---

2. What is the difference between a Python list and a NumPy array?

Answer:

- **List**: Can store elements of different data types, slower, and doesn't support element-wise operations.
- **NumPy Array**: Stores elements of the **same data type**, supports **vectorized operations**, and is **much faster** due to C-based implementation.

---

3. What is the shape and size of a NumPy array?

Answer:

- **shape** → returns a tuple showing the dimensions of the array (e.g., (3, 4)).
- **size** → returns the total number of elements in the array.

---

4. What are NumPy data types (dtypes)?

Answer:
NumPy supports fixed-size numeric data types like:
int32, int64, float32, float64, bool_, complex_, etc.

---

5. What is broadcasting in NumPy?

Answer:
Broadcasting allows NumPy to perform arithmetic operations on arrays of **different shapes** by expanding smaller arrays automatically to match the larger array's shape.

---

6. What is the difference between view and copy in NumPy?

Answer:

- **View**: Shares the same data buffer — changes affect the original array.
- **Copy**: Creates a new independent array — changes don't affect the original.

---

7. What is vectorization in NumPy?

Answer:
Vectorization means performing operations directly on arrays without explicit loops, leading to **faster computation** by using optimized C code internally.

---

8. Explain slicing in NumPy.

Answer:
Slicing allows you to extract a portion of an array using [start:stop:step].
Example:
arr[1:5:2] extracts elements from index 1 to 4 with a step of 2.

---

9. What are universal functions (ufuncs) in NumPy?

Answer:
Ufuncs are functions that operate **element-wise** on arrays.
Examples: np.add(), np.sqrt(), np.exp(), np.log().

---

10. How does NumPy handle missing or NaN values?

Answer:
NumPy provides functions like np.isnan(), np.nanmean(), np.nan_to_num() to detect and handle NaN (Not a Number) values.

💻 **10 Practical NumPy Questions and Answers**

1. Create a 1D NumPy array of numbers from 1 to 10.

```
import numpy as np

arr = np.arange(1, 11)

print(arr)
```

Output: [1 2 3 4 5 6 7 8 9 10]

---

2. Create a 3×3 matrix of zeros.

```
np.zeros((3, 3))
```

Output:

```
[[0. 0. 0.]
 [0. 0. 0.]
 [0. 0. 0.]]
```

---

3. Create a 2×4 matrix filled with random integers between 10 and 50.

```
np.random.randint(10, 50, (2, 4))
```

---

4. Find the mean, median, and standard deviation of an array.

```
arr = np.array([10, 20, 30, 40, 50])

print(np.mean(arr), np.median(arr), np.std(arr))
```

Output: 30.0 30.0 14.1421356237

---

5. Reshape a 1D array of 9 elements into a 3×3 matrix.

```
arr = np.arange(1, 10)

print(arr.reshape(3, 3))
```

---

6. Find the maximum and minimum element in an array along with their indices.

```
arr = np.array([4, 9, 2, 7, 5])

print(arr.max(), arr.min())

print(arr.argmax(), arr.argmin())
```

Output:
9 2
1 2

---

7. Perform element-wise addition of two arrays.

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

b = np.array([4, 5, 6])

print(np.add(a, b))
```

Output: [5 7 9]

---

8. Flatten a 2D array into a 1D array.

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

print(arr.flatten())
```

Output: [1 2 3 4]

---

9. Extract all even numbers from a NumPy array.

```
arr = np.arange(1, 11)

print(arr[arr % 2 == 0])
```

Output: [2 4 6 8 10]

---

10. Stack two arrays vertically and horizontally.

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

b = np.array([[5, 6], [7, 8]])

print(np.vstack((a, b)))  # Vertical

print(np.hstack((a, b)))  # Horizontal
```

Output:

```
[[1 2]
 [3 4]
 [5 6]
 [7 8]]


[[1 2 5 6]
 [3 4 7 8]]
```
