

**NumPy Exam Paper (Total 30 Questions - 2 Marks Each)**



**Section A: NumPy Creation (13 Questions)**

1. **From a Python List**   
 Write a code to create a NumPy array from the following list:

list1 = [1, 2, 3, 4, 5]

**Answer: import numpy as np**

**# Given list**

**list1 = [1, 2, 3, 4, 5]**

**# Convert to NumPy array**

**array1 = np.array(list1)**

**# Print the NumPy array**

**print(array1)**

2. **From a Tuple**   
 Convert the tuple (10, 20, 30, 40) into a NumPy array.

**Answer: import numpy as np**

**# Given tuple**

**tuple1 = (10, 20, 30, 40)**

**# Convert to NumPy array**

**array1 = np.array(tuple1)**

**# Print the NumPy array**

**print(array1)**

3. **Using arange**   
 Create a NumPy array from 0 to 30 with a step of 5 using arange .

**Answer: import numpy as np**

**# Create array using arange**

**array1 = np.arange(0, 31, 5)**

**# Print the NumPy array**

**print(array1)**

4. **Using linspace**   
 Create an array of 6 evenly spaced values between 0 and 50 using linspace .

**Answer: import numpy as np**

**# Create an array with 6 evenly spaced values between 0 and 50**

**array1 = np.linspace(0, 50, 6)**

**# Print the NumPy array**

**print(array1)**

5. **Using ones**   
 Create a 4x4 matrix of ones using NumPy.

**Answer:**

import numpy as np

# Create a 4x4 matrix of ones

matrix = np.ones((4, 4))

# Print the matrix

print(matrix)

6. **Using zeros**   
 How do you create a 3x3 matrix of zeros in NumPy?

**Answer: import numpy as np**

**# Create a 3x3 matrix of zeros**

**matrix = np.zeros((3, 3))**

**# Print the matrix**

**print(matrix)**

7. **Using empty**   
 What is the purpose of empty in NumPy? Create a 2x2 uninitialized array.

**Answer:** The np.empty() function is used to create an **uninitialized array** of a given shape and data type. Unlike np.zeros() or np.ones(), it does **not initialize** the elements to any specific values. Instead, it allocates memory and leaves the array with whatever values happen to be in memory at that time (random garbage values).

8. **Using full**   
 Create a 5x5 array where all elements are equal to 9 using full .

**Answer: import numpy as np**

**# Create a 5x5 array filled with 9**

**array1 = np.full((5, 5), 9)**

**# Print the array**

**print(array1)**

9. **Using eye**   
 Create a 3x3 identity matrix using eye .

**Answer:** **import numpy as np**

**# Create a 3x3 identity matrix**

**identity\_matrix = np.eye(3)**

**# Print the matrix**

**print(identity\_matrix)**

10. **Using random**   
Generate a 2x2 matrix of random integers between 1 and 100 using NumPy’s random module.

**Answer: import numpy as np**

**# Generate a 2x2 matrix with random integers between 1 and 100**

**random\_matrix = np.random.randint(1, 101, (2, 2))**

**# Print the matrix**

**print(random\_matrix)**

11. **Using astype**   
 Convert the array np.array([10, 20, 30]) to a float array using astype .

**Answer:**

import numpy as np

# Create an integer array

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

# Convert to float

float\_arr = arr.astype(float)

# Print the float array

print(float\_arr)

12. **Using reshape**   
 Reshape the array np.arange(9) into a 3x3 matrix.

**Answer: import numpy as np**

**# Create a 1D array with values from 0 to 8**

**arr = np.arange(9)**

**# Reshape into a 3x3 matrix**

**matrix = arr.reshape(3, 3)**

**# Print the reshaped matrix**

**print(matrix)**

13. **Using diag**   
 Create a 4x4 matrix with the diagonal elements [10, 20, 30, 40] using diag .

**Answer: import numpy as np**

**# Create a 4x4 diagonal matrix**

**diag\_matrix = np.diag([10, 20, 30, 40])**

**# Print the matrix**

**print(diag\_matrix)**



**Section B: Indexing, Slicing, and Fancy Indexing (12 Questions)**

14. **Accessing Elements in 1D Array**   
 Access the third element of the array np.array([5, 10, 15, 20, 25]) .

**Answer: import numpy as np**

**# Create the array**

**arr = np.array([5, 10, 15, 20, 25])**

**# Access the third element (index 2)**

**third\_element = arr[2]**

**# Print the result**

**print(third\_element)**

15. **Accessing Elements in 2D Array**   
Retrieve the element at row 2, column 3 from the 2D array np.array([[1, 2, 3], [4, 5, 6], [7, 8, 9]]) .

**Answer: import numpy as np**

**# Create the 2D array**

**arr = np.array([[1, 2, 3],**

**[4, 5, 6],**

**[7, 8, 9]])**

**# Access the element at row 2, column 3 (indexing starts from 0)**

**element = arr[1, 2]**

**# Print the result**

**print(element)**

16. **Negative Indexing in 1D Array**   
Use negative indexing to access the last element of the array np.array([12, 23, 34, 45]) .

**Answer: import numpy as np**

**# Create the array**

**arr = np.array([12, 23, 34, 45])**

**# Access the last element using negative indexing**

**last\_element = arr[-1]**

**# Print the result**

**print(last\_element)**

17. **Slicing a 1D Array**   
 Slice the array np.array([10, 20, 30, 40, 50, 60]) to get the first four elements.

import numpy as np

# Create the array

arr = np.array([10, 20, 30, 40, 50, 60])

# Slice to get the first four elements

first\_four = arr[:4]

# Print the result

print(first\_four)

**Answer:**

18. **Slicing a 2D Array**   
From the array np.array([[10, 20, 30], [40, 50, 60], [70, 80, 90]]) , slice out the first two rows and the first two columns.

**Answer: import numpy as np**

**# Create the 2D array**

**arr = np.array([[10, 20, 30],**

**[40, 50, 60],**

**[70, 80, 90]])**

**# Slice the first two rows and first two columns**

**sliced\_arr = arr[:2, :2]**

**# Print the result**

**print(sliced\_arr)**

19. **Reverse a 1D Array Using Slicing**   
 Reverse the array np.array([1, 2, 3, 4, 5]) using slicing.

**Answer: import numpy as np**

**# Create the array**

**arr = np.array([1, 2, 3, 4, 5])**

**# Reverse the array using slicing**

**reversed\_arr = arr[::-1]**

**# Print the result**

**print(reversed\_arr)**

20. **Fancy Indexing in 1D Array**   
Using fancy indexing, select the 1st, 3rd, and 4th elements from the array arr = np.array([10, 20, 30, 40, 50]) .

**Answer: import numpy as np**

**# Create the array**

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

**# Select the 1st, 3rd, and 4th elements (indexing starts from 0)**

**selected\_elements = arr[[0, 2, 3]]**

**# Print the result**

**print(selected\_elements)**

21. **Fancy Indexing in 2D Array**   
 Use fancy indexing to retrieve elements at positions (0, 1), (1, 2), and (2, 0) from the array arr = np.array([[1, 2, 3], [4, 5, 6], [7, 8, 9]]) .

**Answer:** **import numpy as np**

**# Define the 2D array**

**arr = np.array([[1, 2, 3],**

**[4, 5, 6],**

**[7, 8, 9]])**

**# Use fancy indexing to select (0,1), (1,2), and (2,0)**

**selected\_elements = arr[[0, 1, 2], [1, 2, 0]]**

**# Print the result**

**print(selected\_elements)**

22. **Slice every second element in 1D Array**   
 From the array np.array([1, 2, 3, 4, 5, 6, 7, 8]) , slice every second element.

**Answer:**

import numpy as np

arr = np.array([1, 2, 3, 4, 5, 6, 7, 8])

# Slice every second element

result = arr[::2]

print(result)

23. **Slice every second column in a 2D Array**   
For the array np.array([[10, 20, 30, 40], [50, 60, 70, 80], [90, 100, 110, 120]]) , slice every second column.

**Answer: import numpy as np**

**arr = np.array([**

**[10, 20, 30, 40],**

**[50, 60, 70, 80],**

**[90, 100, 110, 120]**

**])**

**# Extract every second column**

**result = arr[:, ::2]**

**print(result)**

24. **Access last row using negative indexing**   
Retrieve the last row from the array np.array([[1, 2], [3, 4], [5, 6], [7, 8]]) using negative indexing.

**Answer: import numpy as np**

**arr = np.array([[1, 2], [3, 4], [5, 6], [7, 8]])**

**# Access the last row using negative indexing**

**last\_row = arr[-1]**

**print(last\_row)**

25. **Reverse each row in a 2D Array**   
Reverse the order of elements in each row of the array np.array([[1, 2, 3], [4, 5, 6], [7, 8, 9]]) .

**Answer: import numpy as np**

**arr = np.array([[1, 2, 3],**

**[4, 5, 6],**

**[7, 8, 9]])**

**# Reverse elements in each row**

**reversed\_rows = arr[:, ::-1]**

**print(reversed\_rows)**



**Section C: NumPy Copying (5 Questions)**

26. **Shallow Copy Using view()**   
 What is a shallow copy in NumPy? Demonstrate with a code example using view() .

**Answer: import numpy as np**

**# Original array**

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

**# Creating a shallow copy using view()**

**shallow\_copy = arr.view()**

**# Modifying the shallow copy**

**shallow\_copy[1] = 99**

**print("Original Array:", arr) # Changes reflect in the original array**

**print("Shallow Copy:", shallow\_copy)**

27. **Shallow Copy Modification**   
In a shallow copy, how does modifying an element affect the original array? Provide a code example.

**Answer:**

import numpy as np

# Create an original NumPy array

original\_array = np.array([1, 2, 3, 4, 5])

# Create a shallow copy using view()

shallow\_copy = original\_array.view()

# Modify an element in the shallow copy

shallow\_copy[2] = 99 # Changing the third element

# Print both arrays

print("Original Array:", original\_array) # Original array is also modified

print("Shallow Copy:", shallow\_copy)

28. **Deep Copy Using copy()**   
What is a deep copy in NumPy? Show how to create a deep copy using the copy() method.

**Answer: import numpy as np**

**# Create an original NumPy array**

**original\_array = np.array([1, 2, 3, 4, 5])**

**# Create a deep copy using copy()**

**deep\_copy = original\_array.copy()**

**# Modify an element in the deep copy**

**deep\_copy[2] = 99 # Changing the third element**

**# Print both arrays**

**print("Original Array:", original\_array) # Original remains unchanged**

**print("Deep Copy:", deep\_copy)**

29. **Effect of Modifying Deep Copy**   
 Does modifying a deep copy affect the original array? Illustrate with an example.

**Answer: import numpy as np**

**# Create an original NumPy array**

**original\_array = np.array([10, 20, 30, 40, 50])**

**# Create a deep copy using copy()**

**deep\_copy = original\_array.copy()**

**# Modify an element in the deep copy**

**deep\_copy[1] = 99 # Changing the second element**

**# Print both arrays**

**print("Original Array:", original\_array) # Original remains unchanged**

**print("Deep Copy:", deep\_copy) # Only deep\_copy is modified**

30. **Difference Between Shallow and Deep Copy**   
 Briefly explain the difference between shallow and deep copy in NumPy with examples.

**Answer:** Shallow Copy Deep copy

|  |  |  |
| --- | --- | --- |
| **Definition** | Creates a new array that shares data with the original array. | Creates a completely independent copy of the array. |

|  |  |  |
| --- | --- | --- |
| **Memory** | Both arrays share the same memory location. | Stored in a separate memory location. |

|  |  |  |
| --- | --- | --- |
| **Modification Effect** | Changes in the copied array reflect in the original array and vice versa. | Changes in the copied array do **not** affect the original array. |

|  |  |  |
| --- | --- | --- |
| **Use Case** | Used when you want to save memory but still reference the original data. | Used when you need a completely independent copy of the data |

