

**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  
list1 = [1, 2, 3, 4, 5]  
array1 = np.array(list1)  
print(array1)

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

**Answer:**import numpy as np  
tuple = (10, 20, 30, 40)  
my\_array = np.array(tuple)  
print(my\_array)

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

**Answer:**import numpy as np  
arr = np.arange(0, 31, 5)  
print(arr)

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

**Answer:**import numpy as np  
arr = np.linspace(0, 50, 6)  
print(arr)

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

**Answer:**

import numpy as np  
 matrix = np.ones((4, 4))  
 print(matrix)

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

**Answer:**import numpy as np  
matrix\_0= np.zeros((3, 3))  
print(matrix\_0)

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

**Answer:**import numpy as np  
uninitialized\_array = np.empty((2, 2))  
print(uninitialized\_array)

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

**Answer:**import numpy as np

array\_9 = np.full((5, 5), 9)

print(array\_9)

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

**Answer:**import numpy as np  
Matrix\_id = np.eye(3)  
print(matrix\_id)

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

**Answer:**import numpy as np

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

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  
 arr = np.array([10, 20, 30])  
 float\_array = arr.astype(float)  
 print(float\_array)

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

**Answer:**import numpy as np  
arr = np.arange(9)  
reshaped\_matrix = arr.reshape(3, 3)  
print(reshaped\_matrix)

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

**Answer:**import numpy as np  
diagonal\_elements = [10, 20, 30, 40]  
matrix = np.diag(diagonal\_elements)  
print(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

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

third\_element = arr[2]

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  
arr = np.array([[1, 2, 3], [4, 5, 6], [7, 8, 9]])  
element = arr[1, 2]

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

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

last\_element = arr[-1]

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.

**Answer:**import numpy as np

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

first\_four\_elements = arr[:4]

print(first\_four\_elements)

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

arr = np.array([[10, 20, 30], [40, 50, 60], [70, 80, 90]])

sliced\_array = arr[:2, :2]

print(sliced\_array)

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

**Answer:**import numpy as np

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

reversed\_array = arr[::-1]

print(reversed\_array)

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

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

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

arr = np.array([[1, 2, 3], [4, 5, 6], [7, 8, 9]])

elements = arr[[0, 1, 2], [1, 2, 0]]

print(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])

sliced\_array = arr[::2]  
print(sliced\_array)

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]])

sliced\_array = arr[:, ::2]

print(sliced\_array)

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:**

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:**



**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:**

In a **shallow copy**, modifying an element in the copied array **affects the original array** because the shallow copy does not create a new copy of the data. Instead, it creates a new array object that **references the same underlying data** as the original array. Therefore, changes made to the shallow copy will also reflect in the original array since they share the same data.

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

**Answer:**

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

**Answer:**To create a deep copy of a NumPy array, you can use the .copy() method. This method returns a new array that is a **deep copy** of the original array.

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

**Answer:**No, modifying a deep copy does **not** affect the original array in NumPy.

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

**Answer:**1. Shallow Copy:

A shallow copy creates a new array, but it does not create a new copy of the data. Instead, it simply references the same data. So, changes made to the data in one array will also reflect in the other array.

import numpy as np

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

shallow\_copy = arr.view()

arr[0] = 99

print("Original array:", arr)

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

o/p - **Original array: [99 2 3]**

**Shallow copy: [99 2 3]**

2. Deep Copy:

A deep copy creates a completely new array and also duplicates the data. The new array and the original array are completely independent of each other, and changes to one will not affect the other.

import numpy as np

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

deep\_copy = arr.copy()

arr[0] = 99

print("Original array:", arr)

print("Deep copy:", deep\_copy)  
o/p:

**Original array: [99 2 3]**

**Deep copy: [1 2 3]**

