

**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]

numpy\_array = np.array(list1)

print(numpy\_array)

print(type(numpy\_array)) # verify the data type

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

**Answer:**

import numpy as np

my\_tuple = (10, 20, 30, 40)

numpy\_array = np.array(my\_tuple)

print(numpy\_array)

print(type(numpy\_array))

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

**Answer:**

import numpy as np

numpy\_array = np.arange(0, 31, 5)

print(numpy\_array)

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

**Answer:**

import numpy as np

numpy\_array = np.linspace(0, 50, 6)

print(numpy\_array)

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

**Answer:**

import numpy as np

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

print(matrix\_ones)

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

**Answer:**

import numpy as np

matrix\_zeros = np.zeros((3, 3))

print(matrix\_zeros)

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

**Answer:**

import numpy as np

u\_array = np.empty((2, 2))

print(u\_array)

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

**Answer:**

import numpy as np

arr\_n = np.full((5, 5), 9)

print(arr\_n)

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

**Answer:**

import numpy as np

i\_matrix = np.eye(3)

print(i\_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

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

print(r\_matrix)

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

**Answer:**

import numpy as np

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

float\_array = original\_array.astype(float)

print(float\_array)

print(float\_array.dtype)

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

**Answer:**

import numpy as np

original\_array = np.arange(9)

reshaped\_matrix = original\_array.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

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

print(d\_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

my\_array = np.array([5, 10, 15, 20, 25])

third\_ele = my\_array[2]

print(third\_ele)

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

my\_array = np.array([[1, 2, 3],

[4, 5, 6],

[7, 8, 9]])

element = my\_array[1, 2] # Row 2 (index 1), column 3 (index 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

my\_array = np.array([12, 23, 34, 45])

last\_element = my\_array[-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

my\_array = np.array([10, 20, 30, 40, 50, 60])

sliced\_array = my\_array[:4]

print(sliced\_array)

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

my\_array = np.array([[10, 20, 30],

[40, 50, 60],

[70, 80, 90]])

sliced\_array = my\_array[: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

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

reversed\_array = my\_array[::-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])

indices = [0, 2, 3] # Indices of the elements to select

selected\_elements = arr[indices]

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]])

rows = [0, 1, 2]

cols = [1, 2, 0]

selected\_elements = arr[rows, cols]

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])

sliced\_array = arr[1::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[:, 1::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:**

import numpy as np

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

[3, 4],

[5, 6],

[7, 8]])

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]])

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

a **=** geek.arange(10, dtype **=**'int16')

print("a is: \n", a)

v **=** a.view('int32')

print("\n After using view() with dtype = 'int32' a is : \n", a)

v += 1

print("\n After using view() with dtype = 'int32' and adding 1 a is : \n", a)

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

**Answer:**

import copy

a = [[1, 2, 3], [4, 5, 6]]

b = copy.copy(a)

b[0][0] = 99

print(b)

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 copy

a = [[1, 2, 3], [4, 5, 6]]

b = copy.deepcopy(a)

b[0][0] = 99

print(b)

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. This is the fundamental difference between deep and shallow copies.

import numpy as np

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

deep\_copy = original\_array.copy()

print("Original Array before modification:", original\_array)

print("Deep Copy before modification:", deep\_copy)

deep\_copy[2] = 300 # Change the 3rd element to 300

print("\nOriginal Array after modification of deep copy:", original\_array)

print("Deep Copy after modification:", deep\_copy)

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

**Answer:**

**Shallow Copy:**

* Creates a new array object that *points to the same data* as the original.
* Changes in the copy affect the original, and vice versa.
* Created using view().

import numpy as np

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

shallow = original.view()

shallow[0] = 10

print(original)

