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

1. import numpy as np

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

array = np.array(list1)

print("NumPy Array:", array)

1. import numpy as np

tuple1 = (10, 20, 30, 40)

array = np.array(tuple1)

print("NumPy Array:", array)

1. import numpy as np

array = np.arange(0, 30, 5)

print("NumPy Array:", array)

1. import numpy as np

array = np.linspace(0, 50, 6)

print("NumPy Array:", array)

1. import numpy as np

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

print("4x4 Matrix of Ones:\n", matrix)

1. import numpy as np

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

print("3x3 Matrix of Zeros:\n", matrix)

1. import numpy as np

array = np.empty((2, 2))

print("2x2 Uninitialized Array:\n", array)

1. import numpy as np

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

print("5x5 Array with All Elements Equal to 9:\n", array)

1. import numpy as np

identity\_matrix = np.eye(3)

print("3x3 Identity Matrix:\n", identity\_matrix)

1. import numpy as np

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

print("2x2 Matrix of Random Integers:\n", matrix)

1. import numpy as np

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

float\_array = array.astype(float)

print("Float Array:", float\_array)

1. import numpy as np

array = np.arange(9)

reshaped\_array = array.reshape(3, 3)

print("3x3 Reshaped Matrix:\n", reshaped\_array)

1. import numpy as np

diagonal\_elements = [10, 20, 30, 40]

matrix = np.diag(diagonal\_elements)

print("4x4 Matrix with Specified Diagonal:\n", matrix)

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

1. import numpy as np

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

third\_element = array[2]

print("Third Element:", third\_element)

-Output-

Third Element: 15

1. import numpy as np

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

element = array[1, 2]

print("Element at Row 2, Column 3:", element)

-Output-

Element at Row 2, Column 3: 6

1. import numpy as np

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

last\_element = array[-1]

print("Last Element:", last\_element)

-Output-

Last Element: 45

1. import numpy as np

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

sliced\_array = array[:4]

print("First Four Elements:", sliced\_array)

-Output-

First Four Elements: [10 20 30 40]

1. import numpy as np

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

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

print("Sliced Array:\n", sliced\_array)

-Output-

Sliced Array:

[[10 20]

[40 50]]

1. import numpy as np

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

reversed\_array = array[::-1]

print("Reversed Array:", reversed\_array)

-Output-

Reversed Array: [5 4 3 2 1]

1. import numpy as np

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

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

print("Selected Elements:", selected\_elements)

-Output-

Selected Elements: [10 30 40]

1. 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("Selected Elements:", elements)

-Output-

Selected Elements: [2 6 7]

1. import numpy as np

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

sliced\_array = array[::2]

print("Sliced Array (Every Second Element):", sliced\_array)

-Output-

Sliced Array (Every Second Element): [1 3 5 7]

1. import numpy as np

array = np.array([[10, 20, 30, 40], [50, 60, 70, 80], [90, 100, 110, 120]])

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

print("Sliced Array (Every Second Column):\n", sliced\_array)

-Output-

Sliced Array (Every Second Column):

[[ 10 30]

[ 50 70]

[ 90 110]]

1. import numpy as np

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

last\_row = array[-1]

print("Last Row:", last\_row)

-Output-

Last Row: [7 8]

1. import numpy as np

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

reversed\_rows = array[:, ::-1]

print("Array with Reversed Rows:\n", reversed\_rows)

-Output-

Array with Reversed Rows:

[[3 2 1]

[6 5 4]

[9 8 7]]

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

1. A shallow copy in NumPy refers to creating a new array object that shares

the same data as the original array. Modifications to the data in one array

will reflect in the other because they reference the same memory location.

However, changes to the shape or structure of one array (e.g., resizing)

won't affect the other.

import numpy as np

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

shallow\_copy = array.view()

shallow\_copy[1] = 99

print("Original Array:", array)

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

-Output-

Original Array: [10 99 30 40]

Shallow Copy: [10 99 30 40]

1. In a shallow copy, modifying an element in the copy directly affects the

original array because both share the same underlying data.

However, structural changes, such as reshaping, won't impact the other.

import numpy as np

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

shallow\_copy = original.view()

shallow\_copy[2] = 99

print("Original Array After Modification:", original)

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

-Output-

Original Array After Modification: [ 1 2 99 4]

Shallow Copy: [ 1 2 99 4]

1. A deep copy in NumPy creates a completely independent copy of an array.

This means the new array has its own memory allocation, and any changes

made to the copied array will not affect the original array and vice versa.

import numpy as np

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

deep\_copy = original.copy()

deep\_copy[2] = 99

print("Original Array After Modification:", original)

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

-Output-

Original Array After Modification: [1 2 3 4]

Deep Copy: [ 1 2 99 4]

1. No, modifying a deep copy does not affect the original array because

they are entirely independent of each other. Here's an example to illustrate this:

import numpy as np

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

deep\_copy = original.copy()

deep\_copy[1] = 99

print("Original Array After Modification:", original)

print("Deep Copy After Modification:", deep\_copy)

-Output-

Original Array After Modification: [10 20 30 40]

Deep Copy After Modification: [10 99 30 40]

1. **Key Difference**

In a **shallow copy**, both arrays share the same data, so changes to data reflect in both arrays.

In a **deep copy**, the arrays are completely independent, and changes in one do not affect the other.

**Shallow Copy**

A shallow copy creates a new array object but shares the same data as the original array.

Changes to the data in either array affect both, but changes to the shape or

structure (like resizing) affect only the specific array.

import numpy as np

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

shallow\_copy = original.view()

shallow\_copy[1] = 99

print("Original Array:", original)

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

-Output-

Original Array: [ 1 99 3 4]

Shallow Copy: [ 1 99 3 4]

**Deep Copy**

A deep copy creates a completely independent copy of the array.

Changes in one array do not affect the other because they are stored in

separate memory locations.

import numpy as np

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

deep\_copy = original.copy()

deep\_copy[2] = 99

print("Original Array:", original)

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

-Ouput-

Original Array: [10 20 30 40]

Deep Copy: [10 20 99 40]