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:

   arr = np.array(list1)

   print(arr)

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

   Answer:

   arr = np.array(tuple)

   print(arr)

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

   Answer:

   arr = np.arange(0:30:5)

   print(arr)

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

   Answer:

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

print(arr)

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

   Answer:

```
arr  = np.ones([4,4])
```

print(arr)

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

   Answer:

```
arr  = np.zeros([3,3])
```

print(arr)

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

   Answer:

   Empty makes the matrix with empty values and return [ ]

   Import numpy as np

```
arr = np.empty((2, 2))

Print(arr)
```

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

   Answer:

   arr = np.full([5,5],9)

print(arr)

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

   Answer:

   arr = np.eye([3,3])

print(arr)

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

    Answer:

    import numpy as np

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

    Print(arr)

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

    Answer:

arr= np.astype(float)

print(arr)

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

    Answer:

    ```
    import numpy as np

    arr = np.arange(9).reshape(3, 3)

    Print(arr)
    ```

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

    Answer:

    ```
    import numpy as np

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

    Print(arr)
    ```

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])

     arr[2]
    ```

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:

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

arr[1,2]

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

    Answer:

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

    arr[-1]

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

Answer:
arr = np.array ([10, 20, 30, 40, 50, 60])
arr[:4]

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]])

    arr[:2, :2]

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

    Answer:

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

    arr[::-1]

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:

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

    arr[0,2,3]

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:

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

    arr[[0,1,2],[1,2,0]]

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:

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

arr[::2]

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:

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

    arr[:,:2]

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:

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

    arr[::-1,:]

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]])


    arr[:, ::-1]
    ```

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:

    Shallow copy is a copy while which if the change occur then the original dataframe also changes

    import numpy as np

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

    shallow_copy = arr.view()

    shallow_copy[0] = 100

    print("Original Array:", arr)

    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:

In numpy there are two types of copy deep copy and shallow copy in deep copy original dataset won't get change and remains original but in shallow copy the original dataset changes if there is Slight change in copy this happens due to the python backend way from import numpy as np
arr = np.array([1, 2, 3, 4, 5])

Shallow_copy = arr.view()

Shallow_copy[0] = 100

Print("Original Array:", arr)

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:

Deep copy is a copy by which original won’t Change even if the copy changes

```
import numpy as np

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

Deep_copy = arr.copy()

Deep_copy[0] = 100

Print(“Original Array:”, arr)

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:

No modifying a deep copy don’t have any affect on the original array

import numpy as np

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

Deep_copy = arr.copy()

Deep_copy[0] = 100

Print(“Original Array:”, arr)

Print(“Deep Copy:”, Deep_copy)
```

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

Answer:

In numpy there are two types of copy deep copy and shallow copy in deep copy original dataset won't get change and remains original but in shallow copy the original dataset changes if there is Slight change in copy this happens due to the python backend way from import numpy as np

Example for shallow copy:

```
import numpy as np

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

Shallow_copy = arr.view()

Shallow_copy[0] = 100

Print("Original Array:", arr)

Print("Shallow Copy:", shallow_copy)
```

Example for deep copy:

```
import numpy as np

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

Deep_copy = arr.copy()

Deep_copy[0] = 100

Print("Original Array:", arr)

Print("Deep Copy:", Deep_copy)
```