**#1. Creating NumPy Arrays**

## 1. Create a 1D NumPy array with integers from 1 to 10.

Use the np.array method to create the array.

Print the created array.

```
# importing library
import numpy as np

# Example: Creating a 1D array
example_array = np.array([5, 10, 15])
print(example_array)

# Start writing your code here

# importing library
import numpy as np

# Create a 1D NumPy array with integers from 1 to 10
array = np.array([1, 2, 3, 4, 5, 6, 7, 8, 9, 10])

# Print the created array
print("Created 1D array:", array)
```

```
[ 5 10 15]
Created 1D array: [ 1  2  3  4  5  6  7  8  9 10]
```

## 2. Create a 2D array using np.arange with 12 elements and reshape it into a 3x4 array.

Hint: Use reshape to change the dimensions.

Print the reshaped array.

```
# Example: Creating a 1D array
example_array = np.array([5, 10, 15])
print(example_array)

# Start writing your code here

# importing library
import numpy as np

# Create a 2D array using np.arange with 12 elements
```

```
array_2d = np.arange(12)

# Reshape the array into a 3x4 array
reshaped_array = array_2d.reshape(3, 4)

# Print the reshaped array
print("Reshaped 3x4 array:")
print(reshaped_array)
```

```
[ 5 10 15]
Reshaped 3x4 array:
[[ 0  1  2  3]
 [ 4  5  6  7]
 [ 8  9 10 11]]
```

## 3. Create a 1D NumPy array with integers from 1 to 20.

- Slice and print elements from index 5 to 15.

- Reverse the array using slicing.

```
#Example:
# Original array: [1, 2, 3, ..., 20]
# Sliced elements: [6, 7, 8, ..., 15]
# Reversed array: [20, 19, 18, ..., 1]

# Step 1: Create the array
# Start writing your code here


# Step 2: Now slice the array
# Start writing your code here


# Step 3: Reverse the array
# Start writing your code here

# importing library
import numpy as np

# Step 1: Create the array with integers from 1 to 20
array = np.arange(1, 21)

# Print the original array
print("Original array:", array)

# Step 2: Slice the array from index 5 to 15 (inclusive)
sliced_array = array[5:16]  # Slice from index 5 to 15
print("Sliced elements (index 5 to 15):", sliced_array)

# Step 3: Reverse the array using slicing
```

```
reversed_array = array[::-1]  # Reverse the array
print("Reversed array:", reversed_array)


Original array: [ 1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18
19 20]
Sliced elements (index 5 to 15): [ 6  7  8  9 10 11 12 13 14 15 16]
Reversed array: [20 19 18 17 16 15 14 13 12 11 10  9  8  7  6  5  4  3
2  1]
```

## 2. Array Attributes and Properties

### 4. Find the number of dimensions, shape, and size of the following array:

declare arrays different arrays : scalar, 1D, 2D, 3D and print shape, size, and diemnsions for each array type

Use ndim, shape, and size methods to print these properties

```
# Example:
array = np.array([1, 2, 3])
print(array.ndim)
print(array.shape)
print(array.size)


# Start writing your code here

# importing library
import numpy as np

# Scalar array (0D array)
scalar_array = np.array(42)
print("Scalar Array:")
print("ndim:", scalar_array.ndim)
print("shape:", scalar_array.shape)
print("size:", scalar_array.size)
print()

# 1D array
array_1d = np.array([1, 2, 3, 4, 5])
print("1D Array:")
print("ndim:", array_1d.ndim)
print("shape:", array_1d.shape)
print("size:", array_1d.size)
print()
```

```
# 2D array
array_2d = np.array([[1, 2, 3], [4, 5, 6], [7, 8, 9]])
print("2D Array:")
print("ndim:", array_2d.ndim)
print("shape:", array_2d.shape)
print("size:", array_2d.size)
print()

# 3D array
array_3d = np.array([[[1, 2], [3, 4]], [[5, 6], [7, 8]]])
print("3D Array:")
print("ndim:", array_3d.ndim)
print("shape:", array_3d.shape)
print("size:", array_3d.size)


1
(3,)
3
Scalar Array:
ndim: 0
shape: ()
size: 1

1D Array:
ndim: 1
shape: (5,)
size: 5

2D Array:
ndim: 2
shape: (3, 3)
size: 9

3D Array:
ndim: 3
shape: (2, 2, 2)
size: 8
```

## 5. Change the shape of a given array to a 2x5 array:

```
# importing library
import numpy as np

# Create an array using np.arange with 10 elements
arr = np.arange(10)
print("Original array:", arr)

# Change the shape of the array to 2x5
```

```
reshaped_arr = arr.reshape(2, 5)


Original array: [0 1 2 3 4 5 6 7 8 9]
```

Print the reshaped array and its new shape.

```
# Example: Changing shape
reshaped_array = np.arange(6).reshape(2, 3)
print(reshaped_array)

# Start writing your code here
# Print the reshaped array
print("Reshaped 2x5 array:")
print(reshaped_arr)



[[0 1 2]
 [3 4 5]]
Reshaped 2x5 array:
[[0 1 2 3 4]
 [5 6 7 8 9]]
```

## 3. Array Operations

### 6. Perform basic mathematical operations on two 2D arrays:

Create two arrays, A and B, using np.array.

Perform addition, subtraction, multiplication, and division operations.

Print the results for each operation.

```
# Example: Element-wise addition
A = np.array([1,2,3,4])
B = np.array([1,2,3,4])
Add=A+B
print("addition : ",Add)
# Start writing your code here

# importing library
import numpy as np

# Create two 2D arrays, A and B
A = np.array([[1, 2], [3, 4]])
```

```
B = np.array([[5, 6], [7, 8]])

# Perform element-wise addition
addition = A + B
print("Addition:\n", addition)

# Perform element-wise subtraction
subtraction = A - B
print("Subtraction:\n", subtraction)

# Perform element-wise multiplication
multiplication = A * B
print("Multiplication:\n", multiplication)

# Perform element-wise division
division = A / B
print("Division:\n", division)
```

```
addition :  [2 4 6 8]
Addition:
 [[ 6  8]
 [10 12]]
Subtraction:
 [[-4 -4]
 [-4 -4]]
Multiplication:
 [[ 5 12]
 [21 32]]
Division:
 [[0.2        0.33333333]
 [0.42857143 0.5       ]]
```

## 7. Calculate the square root of all elements in the following array:

```
# importing library
import numpy as np

# Create the array
arr = np.array([1, 4, 9, 16, 25])
```

Use the np.sqrt function to calculate square roots.

```
# Start writing your code here
# Calculate the square root of all elements using np.sqrt
sqrt_arr = np.sqrt(arr)

# Print the result
print("Square root of each element:", sqrt_arr)
```

```
Square root of each element: [1. 2. 3. 4. 5.]
```

## 4. Slicing and Indexing

### 8. Declare 1D array of range 1 to 10 and slice 3 elements from 3

```python
# importing library
import numpy as np

# Declare a 1D array with a range from 1 to 10
arr = np.arange(1, 10)
print("Original array:", arr)

# Slice 3 elements from index 3
arr1 = arr[3:6]
print("Sliced array:", arr1)
```

```
Original array: [1 2 3 4 5 6 7 8 9]
Sliced array: [4 5 6]
```

### 9. Perform similar opertations on 2D and 3D arrays

```python
# Start writing your code here
# importing library
import numpy as np

# Create a 2D array (3x3)
arr_2d = np.array([[1, 2, 3], [4, 5, 6], [7, 8, 9]])
print("Original 2D Array:")
print(arr_2d)

# Slice 2D array: Take rows 1 to 2 and columns 1 to 2
arr_2d_slice = arr_2d[1:3, 1:3]
print("\nSliced 2D Array:")
print(arr_2d_slice)


# Create a 3D array (2x3x3)
arr_3d = np.array([[[1, 2, 3], [4, 5, 6], [7, 8, 9]], [[10, 11, 12],
[13, 14, 15], [16, 17, 18]]])
print("\nOriginal 3D Array:")
print(arr_3d)
```

```
# Slice 3D array: Take 2nd matrix and first 2 rows and 2 columns
arr_3d_slice = arr_3d[1, :2, :2]
print("\nSliced 3D Array:")
print(arr_3d_slice)
```

```
Original 2D Array:
[[1 2 3]
 [4 5 6]
 [7 8 9]]

Sliced 2D Array:
[[5 6]
 [8 9]]

Original 3D Array:
[[[ 1  2  3]
  [ 4  5  6]
  [ 7  8  9]]

 [[10 11 12]
  [13 14 15]
  [16 17 18]]]

Sliced 3D Array:
[[10 11]
 [13 14]]
```

## 10. Flatten the following 2D array into a 1D array:

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

Use the ravel function to flatten the array.

```
# Start writing your code here
# importing library
import numpy as np

# Create a 2D array
arr = np.array([[1, 2, 3], [4, 5, 6]])
print("Original 2D Array:")
print(arr)

# Flatten the array using ravel
flattened_arr = np.ravel(arr)
print("\nFlattened 1D Array:")
print(flattened_arr)
```

```
Original 2D Array:
[[1 2 3]
```

```
 [4 5 6]]

Flattened 1D Array:
[1 2 3 4 5 6]
```

## 5. Statistical Operations

### 11. Generate a 4x4 NumPy array with random integers between 10 and 50.

- Find the maximum and minimum values along each row.

```
# Example:
# Array:
# [[12, 45, 23, 34],
#  [25, 16, 10, 40],
#  [33, 44, 12, 18],
#  [50, 29, 30, 19]]
# Max values per row: [45, 40, 44, 50]
# Min values per row: [12, 10, 12, 19]
#Use np.max, np.min, and np.sum with the axis parameter.


# Step 1: Generate the array
# Start writing your code here
# importing library
import numpy as np
arr = np.random.randint(10, 51, size=(4, 4))
print("Generated Array:")
print(arr)

# Step 2: Find max and min values along each row
# Start writing your code here

max_values = np.max(arr, axis=1)
min_values = np.min(arr, axis=1)

print("\nMax values per row:", max_values)
print("Min values per row:", min_values)


Generated Array:
[[35 18 10 18]
 [20 46 41 32]
 [12 22 28 44]
 [20 29 19 38]]
```

```
Max values per row: [35 46 44 38]
Min values per row: [10 20 12 19]
```

## 12. Write a program to create a 5x5 identity matrix using NumPy. Replace all diagonal elements with 10.

```python
# Example:
# Identity matrix:
# [[1, 0, 0, 0, 0],
#  [0, 1, 0, 0, 0],
#  [0, 0, 1, 0, 0],
#  [0, 0, 0, 1, 0],
#  [0, 0, 0, 0, 1]]
# After replacing diagonal elements:
# [[10, 0, 0, 0, 0],
#  [0, 10, 0, 0, 0],
#  [0, 0, 10, 0, 0],
#  [0, 0, 0, 10, 0],
#  [0, 0, 0, 0, 10]]

# Step 1: Create the identity matrix
# Start writing your code here
identity_matrix = np.eye(5)
print("Identity matrix:")
print(identity_matrix)

# Step 2: Replace diagonal elements
# Start writing your code here
np.fill_diagonal(identity_matrix, 10)
print("\nAfter replacing diagonal elements with 10:")
print(identity_matrix)
```

```
Identity matrix:
[[1. 0. 0. 0. 0.]
 [0. 1. 0. 0. 0.]
 [0. 0. 1. 0. 0.]
 [0. 0. 0. 1. 0.]
 [0. 0. 0. 0. 1.]]

After replacing diagonal elements with 10:
[[10.  0.  0.  0.  0.]
 [ 0. 10.  0.  0.  0.]
 [ 0.  0. 10.  0.  0.]
 [ 0.  0.  0. 10.  0.]
 [ 0.  0.  0.  0. 10.]]
```

## 13. Generate a 3D NumPy array (3x3x3) with random integers between 1 and 100.

- Flatten the array into a 1D array.
- Reshape it back into its original dimensions.

```
# Example:
# Original 3D array:
# [[[11, 22, 33],
#   [44, 55, 66],
#   [77, 88, 99]],
#  [[15, 25, 35],
#   [45, 55, 65],
#   [75, 85, 95]],
#  [[12, 24, 36],
#   [48, 60, 72],
#   [84, 96, 18]]]
# Flattened array: [11, 22, 33, ..., 18]
# Reshaped array:
# [[[11, 22, 33],
#   [44, 55, 66],
#   [77, 88, 99]],
#  [[15, 25, 35],
#   [45, 55, 65],
#   [75, 85, 95]],
#  [[12, 24, 36],
#   [48, 60, 72],
#   [84, 96, 18]]]

# Step 1: Generate the 3D array
# Start writing your code here
arr_3d = np.random.randint(1, 101, size=(3, 3, 3))
print("Original 3D array:")
print(arr_3d)

# Step 2: Flatten the array
# Start writing your code here
flattened_arr = arr_3d.ravel()
print("\nFlattened array:")
print(flattened_arr)

# Step 3: Reshape it back
# Start writing your code here
reshaped_arr = flattened_arr.reshape(3, 3, 3)
print("\nReshaped array:")
print(reshaped_arr)
```

```
Original 3D array:
[[[85 78 71]
  [59 94 60]
  [65 66 70]]

 [[32 79  4]
  [85 79 46]
  [74 99 95]]

 [[79 18 79]
  [42 44 67]
  [65 27 22]]]

Flattened array:
[85 78 71 59 94 60 65 66 70 32 79  4 85 79 46 74 99 95 79 18 79 42 44
67
 65 27 22]

Reshaped array:
[[[85 78 71]
  [59 94 60]
  [65 66 70]]

 [[32 79  4]
  [85 79 46]
  [74 99 95]]

 [[79 18 79]
  [42 44 67]
  [65 27 22]]]
```

## 14. Create a NumPy array of 20 evenly spaced numbers between 0 and 10.

- Compute and display the sine and cosine values for each number.

```
 # Example:
# Array: [0.0, 0.526, 1.053, ..., 10.0]
# Sine values: [0.0, 0.502, 0.868, ..., -0.544]
# Cosine values: [1.0, 0.864, 0.496, ..., -0.839]

import numpy as np

# Step 1: Create the array of 20 evenly spaced numbers between 0 and
10
arr = np.linspace(0, 10, 20)
print("Array:", arr)

# Step 2: Compute sine and cosine values for each number
sine_values = np.sin(arr)
```

```
cosine_values = np.cos(arr)

print("\nSine values:", sine_values)
print("Cosine values:", cosine_values)
```

```
Array: [ 0.          0.52631579  1.05263158  1.57894737  2.10526316
2.63157895
  3.15789474  3.68421053  4.21052632  4.73684211  5.26315789
5.78947368
  6.31578947  6.84210526  7.36842105  7.89473684  8.42105263
8.94736842
  9.47368421 10.        ]

Sine values: [ 0.          0.50235115  0.86872962  0.99996678
0.86054034  0.48818921
 -0.01630136 -0.5163796  -0.87668803 -0.99970104 -0.85212237 -
0.47389753
  0.03259839  0.53027082  0.88441346  0.99916962  0.84347795
0.4594799
 -0.04888676 -0.54402111]
Cosine values: [ 1.          0.8646637   0.49528663 -0.00815095 -
0.5093823  -0.87273782
 -0.99986712 -0.8563598  -0.48105935  0.02445069  0.52334259  0.88058
  0.99946853  0.84782832  0.46670422 -0.04074393 -0.53716381 -
0.88818817
 -0.99880433 -0.83907153]
```

## 15. Calculate the standard deviation of the following array:

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

Use the np.std function to find the standard deviation.

```
# Start writing your code here
# importing library
import numpy as np

# Step 1: Define the array
arr = np.array([10, 20, 30, 40, 50])

# Step 2: Calculate the standard deviation
std_deviation = np.std(arr)

print("Standard Deviation:", std_deviation)
```

```
Standard Deviation: 14.142135623730951
```