Name: A T Abbilaash
Roll Number: 23N201
Course: Machine Learning Lab 1

Exercise 1:

1. Creating a 1-D array

```
import numpy as np
print(np.arange(10))
```

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

2. Reshaping a 1D array

```
import numpy as np
l = np.arange(10).reshape(5,-1)
print(l)
```

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

3. Multiply a (5,3) matrix with (3,2) matric

```
import numpy as np
mat1 = np.arange(0,15).reshape(5,3)
mat2 = np.arange(15,21).reshape(3,2)
result = np.dot(mat1,mat2)
print(result)
```

```
[[ 55  58]
 [208 220]
 [361 382]
 [514 544]
 [667 706]]
```

4. Print odd numbers in an array

```
import numpy as np
l = np.array([1,3,5,3,6,7,9,5])
odd_nums = l[l%2!=0]
print(odd_nums)
```

```
[1 3 5 3 7 9 5]
```

5. Replace all odd numbers in an array of 1-10 with -1

```
import numpy as np
arr = np.arange(1,11)
arr[arr%2!=0] = -1
print(arr)
```

```
[-1  2 -1  4 -1  6 -1  8 -1 10]
```

6. Convert a 1D array to a boolean array where all positive values become True

```
import numpy as np
arr = np.arange(-5,11)
arr = arr>0
print(arr)
```

```
[False False False False False False  True  True  True  True  True  True
  True  True  True  True]
```

7. Replace all even numbers in a 1D array with their negative

```
import numpy as np
arr = np.arange(1,11)
arr[arr%2==0] = -arr[arr%2==0]
print(arr)
```

```
[  1  -2   3  -4   5  -6   7  -8   9 -10]
```

8. Create a random 3x3 matrix and normalize it

```
import numpy as np
matrix = np.random.random((3,3))
print("Original Matrix\n",matrix)
normalized = (matrix - np.mean(matrix))/np.std(matrix)
#Z=mat-mean/std
print("Normalized array\n",normalized)
```

```
Original Matrix
 [[0.43099592 0.77792116 0.00261331]
 [0.70419841 0.97907588 0.63651128]
 [0.31923564 0.40647797 0.5843094 ]]
Normalized array
 [[-0.39642454  0.8897334  -1.98456989]
 [ 0.61642062  1.63547546  0.36548373]
 [-0.81075409 -0.48732008  0.17195539]]
```

9. Calculate the sum of the diagonal elements of a 3x3 matrix

```
import numpy as np
```

```
matrix = np.random.random((3,3))
dsum = sum([matrix[i][i] for i in range(0,3)])
print(dsum)
```

```
0.25508144382122533
```

10. Find the indices of non-zero elements from [1,2,0,0,4,0]

```
import numpy as np
arr = np.array([1,2,0,0,4,0])
print(np.nonzero(arr)[0].tolist())
```

```
[0, 1, 4]
```

11. Reverse a 1D array (first element becomes the last)

```
import numpy as np
arr = np.array([1,2,0,0,4,0])
reversed_arr = np.flip(arr)
print(reversed_arr)
```

```
[0 4 0 0 2 1]
```

12. Create a 3x3 identity matrix

```
import numpy as np
arr = np.eye(3)
print(arr)
```

```
[[1. 0. 0.]
 [0. 1. 0.]
 [0. 0. 1.]]
```

13. Reshape a 1D array to a 2D array with 5 rows and 2 columns

```
import numpy as np
arr = np.random.random(10)
arr = arr.reshape(5,2)
print(arr)
```

```
[[0.10372604 0.82736796]
 [0.10552635 0.78046966]
 [0.50624493 0.91376935]
 [0.28036857 0.21178538]
 [0.83640827 0.02489124]]
```

14. Stack two arrays vertically

```
import numpy as np
arr1 = np.random.random(5)
```

```
arr2 = np.random.random(5)
stack_arr = np.vstack((arr1,arr2))
print(stack_arr)
```

```
[[0.86356876 0.06073582 0.43482721 0.4227613  0.54976026]
 [0.40942236 0.65377918 0.03489892 0.58975791 0.57634788]]
```

15. Get the common items between two arrays

```
import numpy as np
arr1 = np.random.randint(0,50,size=10)
arr2 = np.random.randint(0,50,size=10)
intersect_arr = np.intersect1d(arr1,arr2)
print(intersect_arr)
```

```
[18]
```

16. Create a 5x5 matrix with row values ranging from 0 to 4

```
import numpy as np
matrix = np.zeros((5,5))
matrix += np.arange(5)
print(matrix)
```

```
[[0. 1. 2. 3. 4.]
 [0. 1. 2. 3. 4.]
 [0. 1. 2. 3. 4.]
 [0. 1. 2. 3. 4.]
 [0. 1. 2. 3. 4.]]
```

17. Find the index of the maximum value in a 1D array

```
import numpy as np
arr = np.random.random(10)
print(arr)
print("Index:",np.argmax(arr)," Value:",arr[np.argmax(arr)])
```

```
[0.77700343 0.72131738 0.61861682 0.60341715 0.46070049 0.83874788
 0.3243651  0.13555495 0.43148994 0.89256358]
Index: 9  Value: 0.8925635771821737
```

18. Normalize the values in a 1D array between 0 and 1.

```
import numpy as np
arr = np.array([2,3,5,6,17,20])
normalized_arr = (arr - np.min(arr))/(np.max(arr) - np.min(arr))
print(normalized_arr)
```

```
[0.         0.05555556 0.16666667 0.22222222 0.83333333 1.        ]
```

19. Calculate the dot product of two arrays

```
import numpy as np
arr1 = np.random.randint(10,20,size=5)
arr2 = np.random.randint(0,10,size=5)
print(np.dot(arr1,arr2))
```

```
335
```

20. Count the number of elements in an array within a specific range (20 to 30)

```
import numpy as np
arr = np.random.randint(10,50,size=20)
count = np.sum((arr>=20) & (arr<=30))
print(arr)
print(count)
```

```
[21 36 12 17 44 28 13 32 11 29 18 19 20 39 28 45 19 39 29 20]
7
```

21. Find the mean of each row in a 2D array

```
import numpy as np
arr = np.random.randint(10,50,(3,3))
print(arr)
print(np.mean(arr,axis=1))
```

```
[[12 19 48]
 [44 18 47]
 [37 25 46]]
[26.33333333 36.33333333 36.        ]
```

22. Create a random 4x4 matrix and extract the diagonal elements

```
import numpy as np
matrix = np.random.randint(10,50,(4,4))
print(matrix)
print(np.diag(matrix))
```

```
[[35 22 37 25]
 [37 32 20 19]
 [34 16 47 17]
 [15 40 44 44]]
[35 32 47 44]
```

23. Count the number of occurrences of a specific value in an array (n=15)

```
import numpy as np
arr = np.random.randint(10,20,size=20)
count = np.count_nonzero(arr==15)
print(arr)
```

```
print(count)
```

```
[15 18 18 19 13 12 12 15 19 18 10 11 10 11 10 16 17 14 16 18]
2
```

24. Replace all values in a 1D array with the mean of the array

```
import numpy as np
arr = np.random.randint(10,50,size=10)
print(arr)
mean = np.mean(arr)
arr[:] = mean
print(arr)
```

```
[14 39 45 33 33 44 41 44 44 28]
[36 36 36 36 36 36 36 36 36 36]
```

25. Find the indices of the maximum and minimum values in a 1D array

```
import numpy as np
arr = np.random.randint(10,50,size=10)
print(arr)
print("Minimum ind: ",np.argmin(arr))
print("Maximum ind: ",np.argmax(arr))
```

```
Minimum ind:  2
Maximum ind:  4
```

26. Create a 2D array with 1 on the border and 0 inside

```
import numpy as np
matrix = np.ones((5,5))
matrix[1:-1,1:-1] = 0
print(matrix)
```

```
[[1. 1. 1. 1. 1.]
 [1. 0. 0. 0. 1.]
 [1. 0. 0. 0. 1.]
 [1. 0. 0. 0. 1.]
 [1. 1. 1. 1. 1.]]
```

27. Find the unique values and their counts in a 1D array

```
import numpy as np
arr = np.random.randint(0,15,size=20)
values, counts = np.unique(arr, return_counts=True)
print("Unique:", values)
print("Counts:", counts)
```

```
Unique: [ 0  1  2  5  6  8 10 11 13 14]
Counts: [1 1 2 3 3 4 2 2 1 1]
```

28. Create a 3x3 matrix with values ranging from 0 to 8.

```
import numpy as np
matrix = np.arange(9).reshape(3, 3)
print(matrix)
```

```
[[0 1 2]
 [3 4 5]
 [6 7 8]]
```

29. Calculate the exponential of all elements in a 1D array.

```
import numpy as np
arr = np.array([1, 2, 3, 4, 5])
exponential_arr = np.exp(arr)
print(exponential_arr)
```

```
[  2.71828183   7.3890561   20.08553692  54.59815003 148.4131591 ]
```

30. Swap two rows in a 2D array.

```
import numpy as np
matrix = np.random.random((3, 4))
matrix[[0, 1]] = matrix[[1, 0]]
print(matrix)
```

```
Before:  [[0.87490494 0.12585716 0.48168281 0.7754226 ]
 [0.4633068  0.86857742 0.09155646 0.62340317]
 [0.34543925 0.42384049 0.27701921 0.48128202]]
After:  [[0.4633068  0.86857742 0.09155646 0.62340317]
 [0.87490494 0.12585716 0.48168281 0.7754226 ]
 [0.34543925 0.42384049 0.27701921 0.48128202]]          }
```

31. Create a random 3x3 matrix and replace all values greater than 0.5 with 1 and all others with 0.

```
import numpy as np
matrix = np.random.random((3, 3))
matrix[matrix > 0.5] = 1
matrix[matrix <= 0.5] = 0
print(matrix)
```

```
[[0. 1. 0.]
 [0. 1. 1.]
 [1. 1. 1.]]
```

32. Find the indices of the top N maximum values in a 1D array.

```
import numpy as np
arr = np.array([10, 5, 8, 20, 15])
top_indices = np.argsort(arr)[-2:]  # Replace 2 with desired top N
print(top_indices)
```

```
[4 3]
```

33. Calculate the mean of each column in a 2D array.

```
import numpy as np
matrix = np.random.random((4, 3))
column_means = np.mean(matrix, axis=0)
print(column_means)
```

```
[0.23277862 0.70152452 0.65349524]
```

34. Normalize the values in each column of a 2D array.

```
import numpy as np
matrix = np.random.random((4, 3))
normalized_matrix = (matrix - np.mean(matrix, axis=0)) /
np.std(matrix, axis=0)
print(normalized_matrix)
```

```
[[-0.44020306  0.79626839 -1.10490715]
 [ 1.63633814 -1.15826084  1.55402006]
 [-1.05262732 -0.80866766 -0.58742741]
 [-0.14350777  1.17066011  0.1383145 ]]
```

35. Concatenate two 1D arrays.

```
import numpy as np
arr1 = np.array([1, 2, 3])
arr2 = np.array([4, 5, 6])
concatenated_arr = np.concatenate((arr1, arr2))
print(concatenated_arr)
```

```
[1 2 3 4 5 6]
```

36. Create a 2D array with random values and sort each row.

```
import numpy as np
matrix = np.random.random((3, 4))
sorted_matrix = np.sort(matrix, axis=1)
print(sorted_matrix)
```

```
[[0.11021656 0.1652999  0.42275732 0.48843359]
 [0.13627556 0.16730659 0.1908463  0.38244478]
 [0.05364408 0.16755983 0.31251631 0.3440856 ]]
```

37. Compute the mean squared error between two arrays.

```
import numpy as np
arr1 = np.array([1, 2, 3, 4])
arr2 = np.array([2, 3, 4, 5])
mse = np.mean((arr1 - arr2) ** 2)
print(mse)
```

```
1.0
```

38. Replace all negative values in an array with 0.

```
import numpy as np
arr = np.array([-1, 2, -3, 4, -5])
arr[arr < 0] = 0
print(arr)
```

```
[0 2 0 4 0]
```

39. Find the 5th and 95th percentiles of an array.

```
import numpy as np
arr = np.random.random(10)
percentile_5th = np.percentile(arr, 5)
percentile_95th = np.percentile(arr, 95)
print("5th Percentile:", percentile_5th)
print("95th Percentile:", percentile_95th)
```

```
5th Percentile: 0.0857261314474971
95th Percentile: 0.8997378443068365
```

40. Create a random 2x2 matrix and compute its determinant.

```
import numpy as np
matrix = np.random.random((2, 2))
determinant = np.linalg.det(matrix)
print(determinant)
```

```
0.014310277186241978
```

41. Count the number of elements in an array that are greater than the mean.

```
import numpy as np
arr = np.array([1, 2, 3, 4, 5])
count_above_mean = np.sum(arr > np.mean(arr))
print(count_above_mean)
```

```
2
```

42. Calculate the square root of each element in a 1D array.

```
import numpy as np
arr = np.array([4, 9, 16, 25])
sqrt_arr = np.sqrt(arr)
print(sqrt_arr)
```

```
[2. 3. 4. 5.]
```

43. Create a 3x3 matrix and compute the matrix square root.

```
import numpy as np
matrix = np.random.random((3, 3))
matrix_sqrt = np.linalg.matrix_power(matrix, 2)
print(matrix_sqrt)
```

```
[[0.72212697 1.33546075 0.64087534]
 [0.24119656 0.74694008 0.30417958]
 [0.22067009 0.63082856 0.40891365]]
```

44. Convert the data type of an array to float.

```
import numpy as np
arr = np.array([1, 2, 3, 4], dtype=int)
float_arr = arr.astype(float)
print(float_arr)
```

```
[1. 2. 3. 4.]
```

45. Calculate the element-wise absolute values of an array.

```
import numpy as np
arr = np.array([-1, -2, 3, -4])
abs_values = np.abs(arr)
print(abs_values)
```

```
[1 2 3 4]
```

46. Find the indices where elements of two arrays match.

```
import numpy as np
arr1 = np.array([1, 2, 3, 4, 5])
arr2 = np.array([3, 2, 8, 4, 5])
matching_indices = np.where(arr1 == arr2)
print(matching_indices)
```

```
(array([1, 3, 4]),)
```

47. Calculate the cumulative sum of elements in a 1D array.

```
import numpy as np
arr = np.array([1, 2, 3, 4, 5])
cumulative_sum = np.cumsum(arr)
print(cumulative_sum)
```

```
[ 1  3  6 10 15]
```

48. Compute the inverse of a 2x2 matrix.

```
import numpy as np
matrix = np.random.random((2, 2))
inverse_matrix = np.linalg.inv(matrix)
print(inverse_matrix)
```

```
[[-5.62064541  5.62930613]
 [ 4.92778083 -3.78428758]]
```

49. Count the number of non-zero elements in a 2D array.

```
import numpy as np
matrix = np.array([[0, 1, 0], [2, 0, 3], [0, 4, 0]])
non_zero_count = np.count_nonzero(matrix)
print(non_zero_count)
```

```
4
```

50. Create a 2D array and replace all nan values with 0.

```
import numpy as np
matrix = np.array([[1, np.nan, 3], [4, 5, np.nan], [7, 8, 9]])
matrix[np.isnan(matrix)] = 0
print(matrix)
```

```
[[1. 0. 3.]
 [4. 5. 0.]
 [7. 8. 9.]]
```

51. Find the correlation coefficient between two arrays.

```
import numpy as np
arr1 = np.array([1, 2, 3, 4, 5])
arr2 = np.array([3, 4, 5, 6, 7])
correlation_coefficient = np.corrcoef(arr1, arr2)[0, 1]
print(correlation_coefficient)
```

```
0.9999999999999999
```

52. Create a 1D array and remove all duplicate values.

```
import numpy as np
arr = np.array([1, 2, 3, 2, 4, 5, 1])
unique_arr = np.unique(arr)
print(unique_arr)
```

```
[1 2 3 4 5]
```

53. Compute the element-wise product of two arrays.

```
import numpy as np
arr1 = np.array([1, 2, 3])
arr2 = np.array([4, 5, 6])
elementwise_product = np.multiply(arr1, arr2)
print(elementwise_product)
```

```
[ 4 10 18]
```

54. Calculate the standard deviation of each column in a 2D array.

```
import numpy as np
matrix = np.random.random((4, 3))
column_stddev = np.std(matrix, axis=0)
print(column_stddev)
```

```
[0.27114438 0.31897695 0.19336942]
```

55. Create a 2D array and set all values above a certain threshold to that threshold.

```
import numpy as np
matrix = np.random.random((3, 4))
threshold = 0.7
matrix[matrix > threshold] = threshold
print(matrix)
```

```
[[0.37211233 0.22337052 0.39620909 0.7       ]
 [0.62085658 0.3639095  0.03106835 0.00347155]
 [0.7        0.0414794  0.35080181 0.7       ]]
```

56. Create a random 5x5 matrix and replace the maximum value by -1.

```
import numpy as np
matrix = np.random.random((5, 5))
max_value_index = np.unravel_index(np.argmax(matrix), matrix.shape)
matrix[max_value_index] = -1
print(matrix)
```

```
[[ 0.55617628  0.26016978  0.73642064  0.09653032  0.87217427]
 [ 0.92254568  0.71061232  0.90702465  0.51715347  0.57695979]
 [ 0.09236627  0.89672075  0.57159179 -1.          0.80806633]
 [ 0.58795416  0.52558551  0.21656962  0.31068263  0.9190084 ]
 [ 0.52299652  0.92788408  0.57852415  0.69386469  0.82377198]]
```

57. Convert a 1D array of Fahrenheit temperatures to Celsius.

```
import numpy as np
fahrenheit_temps = np.array([32, 68, 100, 212])
celsius_temps = (fahrenheit_temps - 32) * 5/9
print(celsius_temps)
```

```
[  0.          20.          37.77777778 100.        ]
```

58. Compute the outer product of two arrays.

```
import numpy as np
arr1 = np.array([1, 2, 3])
arr2 = np.array([4, 5, 6])
outer_product = np.outer(arr1, arr2)
print(outer_product)
```

```
[[ 4  5  6]
 [ 8 10 12]
 [12 15 18]]
```

59. Create a 1D array with 10 equidistant values between 0 and 1.

```
import numpy as np
equidistant_arr = np.linspace(0, 1, 10)
print(equidistant_arr)
```

```
[0.         0.11111111 0.22222222 0.33333333 0.44444444 0.55555556
 0.66666667 0.77777778 0.88888889 1.        ]
```

60. Compute the cross product of two 3D arrays.

```
import numpy as np
arr1 = np.array([1, 2, 3])
arr2 = np.array([4, 5, 6])
cross_product = np.cross(arr1, arr2)
print(cross_product)
```

```
[0.         0.11111111 0.22222222 0.33333333 0.44444444 0.55555556
 0.66666667 0.77777778 0.88888889 1.        ]
```

61. Calculate the percentile along a specific axis of a 2D array.

```
import numpy as np
```

```
matrix = np.random.random((3, 4))
percentiles_axis1 = np.percentile(matrix, 75, axis=1)
print(percentiles_axis1)
```

```
[0.78124402 0.34172572 0.77345595]
```

62. Create a 1D array and add a border of 0s around it.

```
import numpy as np
arr = np.array([1, 2, 3, 4])
arr_with_border = np.pad(arr, (1, 1), mode='constant',
constant_values=0)
print(arr_with_border)
```

```
[0 1 2 3 4 0]
```

63. Compute the histogram of a 1D array.

```
import numpy as np
arr = np.array([1, 1, 2, 2, 2, 3, 3, 3, 3])
hist, bins = np.histogram(arr, bins=[1, 2, 3, 4])
print("Histogram:", hist)
print("Bin edges:", bins)
```

```
Histogram: [2 3 4]
Bin edges: [1 2 3 4]
```

64. Create a 2D array with random values and normalize each row.

```
import numpy as np
matrix = np.random.random((4, 3))
normalized_rows = matrix / np.linalg.norm(matrix, axis=1,
keepdims=True)
print(normalized_rows)
```

```
[[0.74859861 0.48996093 0.44669722]
 [0.23086597 0.93092462 0.28298455]
 [0.32148259 0.4357115  0.84071662]
 [0.0122106  0.89782833 0.44017631]]
```

65. Create a random 2D array and sort it by the second column.

```
import numpy as np
matrix = np.random.random((3, 4))
sorted_matrix_by_column2 = matrix[matrix[:, 1].argsort()]
print(sorted_matrix_by_column2)
```

```
[[0.5402868  0.37393552 0.13810053 0.13239314]
 [0.67377544 0.81814656 0.13407787 0.24517383]
 [0.84241354 0.98386653 0.89637111 0.94639661]]
```

66. Calculate the determinant of a 3x3 matrix.

```
import numpy as np
matrix = np.random.random((3, 3))
determinant = np.linalg.det(matrix)
print(determinant)
```

```
-0.40708046108130447
```

67. Calculate the element-wise exponentiation of a 1D array.

```
import numpy as np
arr = np.array([2, 3, 4])
exponentiated_arr = np.exp(arr)
print(exponentiated_arr)
```

```
[ 7.3890561  20.08553692 54.59815003]
```

68. Calculate the Frobenius norm of a 2D array.

```
import numpy as np
matrix = np.random.random((3, 4))
frobenius_norm = np.linalg.norm(matrix)
print(frobenius_norm)
```

```
1.962261365982598
```

69. Create a 2D array with random values and replace the maximum value with the minimum.

```
import numpy as np
matrix = np.random.random((3, 4))
max_value_index = np.unravel_index(np.argmax(matrix), matrix.shape)
min_value = np.min(matrix)
matrix[max_value_index] = min_value
print(matrix)
```

```
[[0.75364287 0.34378423 0.25473139 0.58185362]
 [0.69773434 0.12844785 0.39774801 0.6265039 ]
 [0.31228892 0.78129872 0.06542822 0.06542822]]
```

70. Compute the matrix multiplication of two 2D arrays.

```
import numpy as np
matrix1 = np.random.random((3, 4))
```

```
matrix2 = np.random.random((4, 5))
matrix_multiplication = np.dot(matrix1, matrix2)
print(matrix_multiplication)
```

```
[[0.82413759 0.91433369 0.75364353 0.54813274 1.14861562]
 [0.89875187 0.9510833  0.85455169 0.41236335 1.1746858 ]
 [1.19476796 1.48191497 1.44578463 0.88921017 1.79637794]]
```

71. Create a 1D array and set the values between 10 and 20 to 0.

```
import numpy as np
arr = np.array([5, 15, 12, 18, 25])
arr[(arr >= 10) & (arr <= 20)] = 0
print(arr)
```

```
[ 5  0  0  0 25]
```

72. Compute the inverse hyperbolic sine of each element in a 1D array.

```
import numpy as np
arr = np.array([1, 2, 3, 4])
inverse_sineh_arr = np.arcsinh(arr)
print(inverse_sineh_arr)
```

```
[0.88137359 1.44363548 1.81844646 2.09471255]
```

73. Compute the Kronecker product of two arrays.

```
import numpy as np
arr1 = np.array([1, 2])
arr2 = np.array([3, 4])
kronecker_product = np.kron(arr1, arr2)
print(kronecker_product)
```

```
[3 4 6 8]
```

74. Calculate the mean absolute deviation of a 1D array.

```
import numpy as np
arr = np.array([1, 2, 3, 4, 5])
mean_absolute_deviation = np.mean(np.abs(arr - np.mean(arr)))
print(mean_absolute_deviation)
```

```
1.2
```

75. Create a 3x3 matrix and set all values above the main diagonal to zero.

```
import numpy as np
matrix = np.random.random((3, 3))
matrix[np.triu_indices(3, 1)] = 0
```

```
print(matrix)
```

```
[[0.65034521 0.         0.        ]
 [0.51892408 0.31739196 0.        ]
 [0.62292293 0.30477768 0.97759478]]
```

76. Count the number of occurrences of each unique value in a 1D array.

```
import numpy as np
arr = np.array([2, 2, 1, 3, 3, 3, 4])
unique_values, counts = np.unique(arr, return_counts=True)
print("Unique values:", unique_values)
print("Counts:", counts)
```

```
Unique values: [1 2 3 4]
Counts: [1 2 3 1]
```

77. Compute the cumulative product of elements along a given axis in a 2D array.

```
import numpy as np
matrix = np.random.random((3, 4))
cumulative_product_axis0 = np.cumprod(matrix, axis=0)
print(cumulative_product_axis0)
```

```
[[0.80002244 0.5269954  0.9215188  0.31338015]
 [0.79847231 0.30228644 0.46744854 0.15522858]
 [0.01012439 0.26532087 0.08731155 0.04599822]]
```

78. Round elements of a 1D array to the nearest integer.

```
import numpy as np
arr = np.array([1.2, 2.7, 3.5, 4.9])
rounded_arr = np.round(arr)
print(rounded_arr)
```

```
[1. 3. 4. 5.]
```

79. Create a 1D array and append a new element to the end.

```
import numpy as np
arr = np.array([1, 2, 3])
new_element = 4
arr = np.append(arr, new_element)
print(arr)
```

```
[1 2 3 4]
```

80. Calculate the element-wise absolute difference between two arrays.

```
import numpy as np
```

```
arr1 = np.array([3, 7, 1, 10, 4])
arr2 = np.array([2, 5, 8, 1, 7])
absolute_difference = np.abs(arr1 - arr2)
print(absolute_difference)
```

```
[1 2 7 9 3]
```

81. Create a 2D array with random values and replace the maximum value in each row with -1.

```
import numpy as np
matrix = np.random.random((3, 4))
max_values_indices = np.argmax(matrix, axis=1)
matrix[np.arange(matrix.shape[0]), max_values_indices] = -1
print(matrix)
```

```
[[ 0.31852732  0.68487224  0.42844784 -1.        ]
 [ 0.17127598  0.3087585   0.16594259 -1.        ]
 [ 0.91290646 -1.          0.23629088  0.37944377]]
```

82. Normalize the columns of a 2D array to have a sum of 1.

```
import numpy as np
matrix = np.random.random((3, 4))
normalized_columns = matrix / np.sum(matrix, axis=0, keepdims=True)
print(normalized_columns)
```

```
[[0.26562076 0.14433391 0.42604834 0.36197828]
 [0.2575003  0.45049902 0.13053922 0.43164451]
 [0.47687894 0.40516707 0.44341244 0.20637721]]
```

83. Find the indices of the top N minimum values in a 1D array.

```
import numpy as np
arr = np.array([10, 5, 8, 1, 7])
top_indices = np.argsort(arr)[:2]  # Replace 2 with desired top N
print(top_indices)
```

```
[3 1]
```

84. Convert the elements of a 1D array to strings.

```
import numpy as np
arr = np.array([1, 2, 3, 4])
string_arr = arr.astype(str)
print(string_arr)
```

```
['1' '2' '3' '4']
```

85. Compute the percentile rank of each element in a 1D array.

```
import numpy as np
from scipy.stats import percentileofscore
arr = np.array([1, 2, 3, 4, 5])
percentile_rank = np.array([percentileofscore(arr, value) for value
in arr])
print(percentile_rank)
```

```
[ 20.  40.  60.  80. 100.]
```

86. Create a 1D array and shuffle its elements randomly.

```
import numpy as np
arr = np.array([1, 2, 3, 4, 5])
np.random.shuffle(arr)
print(arr)
```

```
[1 4 5 3 2]
```

87. Check if all elements in a 1D array are non-zero.

```
import numpy as np
arr = np.array([1, 2, 3, 4, 5])
all_nonzero = np.all(arr != 0)
print(all_nonzero)
```

```
True
```

88. Find the indices of the maximum value in each row of a 2D array.

```
import numpy as np
matrix = np.random.random((3, 4))
max_indices_per_row = np.argmax(matrix, axis=1)
print(max_indices_per_row)
```

```
[2 1 3]
```

89. Create a 2D array and replace all nan values with the mean of the array.

```
import numpy as np
matrix = np.array([[1, np.nan, 3], [4, 5, np.nan], [7, 8, 9]])
nan_mean = np.nanmean(matrix)
matrix[np.isnan(matrix)] = nan_mean
print(matrix)
```

```
[[1.         5.28571429 3.        ]
 [4.         5.         5.28571429]
 [7.         8.         9.        ]]
```

90. Calculate the mean of each row in a 2D array ignoring nan values.

```
import numpy as np
matrix = np.array([[1, 2, np.nan], [4, np.nan, 6], [7, 8, 9]])
row_means_ignore_nan = np.nanmean(matrix, axis=1)
print(row_means_ignore_nan)
```

```
[1.5 5.  8. ]
```

91. Compute the sum of diagonal elements in a 2D array.

```
import numpy as np
matrix = np.random.random((3, 3))
diagonal_sum = np.trace(matrix)
print(diagonal_sum)
```

```
1.984795918371303
```

92. Convert radians to degrees for each element in a 1D array.

```
import numpy as np
arr_in_radians = np.array([np.pi/2, np.pi, 3*np.pi/2])
arr_in_degrees = np.degrees(arr_in_radians)
print(arr_in_degrees)
```

```
[ 90. 180. 270.]
```

93. Calculate the pairwise Euclidean distance between two arrays.

```
import numpy as np
arr1 = np.array([1, 2, 3])
arr2 = np.array([4, 5, 6])
euclidean_distance = np.linalg.norm(arr1 - arr2)
print(euclidean_distance)
```

```
5.196152422706632
```

94. Create a 1D array and set the values between the 25th and 75th percentile to 0.

```
import numpy as np
arr = np.array([10, 20, 30, 40, 50])
percentile_25th = np.percentile(arr, 25)
percentile_75th = np.percentile(arr, 75)
arr[(arr >= percentile_25th) & (arr <= percentile_75th)] = 0
print(arr)
```

```
[10  0  0  0 50]
```

95. Calculate the element-wise square of the difference between two arrays.

```
import numpy as np
arr1 = np.array([1, 2, 3])
arr2 = np.array([4, 5, 6])
squared_difference = (arr1 - arr2)**2
print(squared_difference)
```

```
[9 9 9]
```

96. Replace all even numbers in a 1D array with the next odd number.

```
import numpy as np
arr = np.array([2, 5, 8, 12, 15])
arr[arr % 2 == 0] += 1
print(arr)
```

```
[ 3  5  9 13 15]
```

97. Create a 2D array and normalize each column by its range..

```
import numpy as np
matrix = np.random.random((3, 4))
normalized_columns_range = (matrix - np.min(matrix, axis=0)) /
(np.max(matrix, axis=0) - np.min(matrix, axis=0))
print(normalized_columns_range)
```

```
[[0.         0.         0.         0.        ]
 [1.         0.36396856 0.6441989  1.        ]
 [0.4005366  1.         1.         0.28767957]]
```

98. Compute the cumulative sum of elements along a given axis in a 2D array.

```
import numpy as np
matrix = np.random.random((3, 4))
cumulative_sum_axis1 = np.cumsum(matrix, axis=1)
print(cumulative_sum_axis1)
```

```
[[0.75153179 1.70329048 2.13725996 2.40057035]
 [0.56986966 1.04643009 1.40574359 1.75522435]
 [0.5794329  0.8892024  1.85296892 1.86521685]]
```

99. Check if any element in a 1D array is non-zero.

```
import numpy as np
arr = np.array([0, 0, 0, 1, 0])
any_nonzero = np.any(arr != 0)
print(any_nonzero)
```

```
True
```

100. Create a 2D array with random integers and replace all values greater than a certain threshold with that threshold.

```
import numpy as np
matrix = np.random.randint(0, 100, size=(3, 4))
threshold = 75
matrix[matrix > threshold] = threshold
print(matrix)
```

```
[[66 55 75 45]
 [19 73 62 43]
 [59 75 45 27]]
```