OUTPUT

```
1. 5x5 matrix:
 [[ 1  2  3  4  5]
 [ 6  7  8  9 10]
 [11 12 13 14 15]
 [16 17 18 19 20]
 [21 22 23 24 25]]
2. 4x4 Identity matrix:
 [[1. 0. 0. 0.]
 [0. 1. 0. 0.]
 [0. 0. 1. 0.]
 [0. 0. 0. 1.]]

3. 1D array 100 to 200 step 10:
 [100 110 120 130 140 150 160 170 180 190
200]

4. Random 3x3 matrix:
 [[2 5 3]
 [1 5 7]
 [5 5 7]]
Determinant: 79.99999999999997

5. 10 random integers:
 [ 1 81 74 30 61 35 57 93 40 31]

6. 3x4 matrix:
 [[ 0  1  2  3]
 [ 4  5  6  7]
 [ 8  9 10 11]]

7. Matrix A:
 [[7 5 8]
 [3 4 2]
 [5 2 8]]
Matrix B:
 [[8 4 1]
 [8 7 5]
 [9 7 3]]
Matrix multiplication:
 [[168 119  56]
 [ 74  54  29]
 [128  90  39]]

8. 2x2 matrix:
 [[4 2]
 [1 3]]
Eigenvalues: [5. 2.]
Eigenvectors:
 [[ 0.89442719 -0.70710678]
 [ 0.4472136   0.70710678]]

9. 5x5 random matrix:
 [[0.41565666 0.19537384 0.56510639 0.65413812
0.3659473 ]
 [0.872268   0.53047973 0.76894145 0.88125921
0.57883316]
 [0.08209867 0.57915092 0.6895216  0.50792415
0.10694109]
 [0.44636677 0.89013347 0.83313624 0.57966317
0.77894507]
 [0.10835876 0.92208756 0.35957753 0.70785097
0.97164215]]
Diagonal elements: [0.41565666 0.53047973
0.6895216  0.57966317 0.97164215]

10:
Original: [23 34 11 78 53 14 57 36 70 93]
Normalized: [0.14634146 0.2804878  0.
0.81707317 0.51219512 0.03658537
 0.56097561 0.30487805 0.7195122  1.]

11:
Original:
 [[50 75 73  7]
 [50 12 34  9]
 [76 14  1 42]
 [21 57 47 34]]
Row-wise:
 [[ 7 50 73 75]
 [ 9 12 34 50]
 [ 1 14 42 76]
 [21 34 47 57]]
Col-wise:
 [[21 12  1  7]
```

12:

Array: [14 21 33 1 46 91 37 97 22 19]

Max index: 7

Min index: 3

13:

Ravel: [0 1 2 3 4 5 6 7 8]

Flatten: [0 1 2 3 4 5 6 7 8]

14:

Matrix:

[[2 1 5]

[4 7 7]

[4 6 9]]

Inverse:

[[1.5   1.5   -2.   ]

[-0.57142857 -0.14285714 0.42857143]

[-0.28571429 -0.57142857
0.71428571]]

15:

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

16:

[-1 1 -1 3 -1 5 -1 7 -1 9 -1 11 -1 13 -1 15 -
1 17 -1 19 -1]

17: 32

18: Trace: 2.4145219298603497

19:

[array([0, 1, 2]), array([3, 4, 5]), array([6,
7, 8])]

20:

[[0.1709877 0.59093265 0.13787577]

[0.67320209 0.17664985 0.31463425]

[0.79049726 0.45015421 0.30196355]]

21: [1 3 6 10 15 21 28 36 45 55]

22:

Original:

[[4 4 8 4]

[2 1 2 2]

[6 5 5 4]

[8 8 7 5]]

Upper triangular:

[[4 4 8 4]

[0 1 2 2]

[0 0 5 4]

[0 0 0 5]]

23:

[[0 1 0 1 0 1]

[1 0 1 0 1 0]

[0 1 0 1 0 1]

[1 0 1 0 1 0]

[0 1 0 1 0 1]

[1 0 1 0 1 0]]

24:

[[0.96376028 0.88437542 0.6911788]

[0.6424991 0.75239603 0.64015981]

[0.38483325 0.92871836 0.95758287]]

```
25: [19 18 17 16 15 14 13 12 11 10  9  8  7
6  5  4  3  2  1  0]

26:
Vertical:
 [[1 2]
 [3 4]
 [5 6]
 [7 8]]
Horizontal:
 [[1 2 5 6]
 [3 4 7 8]]

27: Array:
 [[8 1 2]
 [2 4 9]
 [1 3 4]]
Row sum: [11 15  8]
Col sum: [11  8 15]

28:
 [[1. 5. 3.]
 [4. 5. 3.]]

29:  0.9746318461970762

30:
 [[ 3  7 11 15]
 [ 2  6 10 14]
 [ 1  5  9 13]
 [ 0  4  8 12]]


31: [('Alice', 25, 88.5) ('Bob', 30, 92. )]

32:  3

33:
 [[0.4912274  0.30077321 0.4645492  0.50087474
0.44894282]
 [0.00633849 0.79887794 0.26212172 0.22569684
0.49204371]
 [0.61438992 0.10758315 0.09220629 0.66751416
0.39607289]
 [0.03340259 0.46737941 0.4416962  0.5140721
0.56663489]
 [0.37414324 0.79533724 0.07793247 0.30849838
0.35526157]]

# 34. Check arrays equal element-wise
arr6 = np.array([1, 2, 3])
arr7 = np.array([1, 2, 3])
print("\n34:\n", np.array_equal(arr6, arr7))

35:
Hist: [  2  13  48 152 262 251 169  90  11   2]
Bins: [-3.81559052 -3.07917613 -2.34276174 -
1.60634735 -0.86993296 -0.13351857
  0.60289582  1.33931021  2.0757246   2.81213899
3.54855338]

36:
 [[2. 3. 4.]
 [2. 3. 4.]
 [2. 3. 4.]]

37:
 {np.int64(1): np.int64(1), np.int64(2): np.int64(2),
np.int64(3): np.int64(3), np.int64(4): np.int64(1)}

38: -0.9999999999999999

39:  [1.  1.5 2.5 3.5 4. ]
```

40:
U:
[[-0.46624291 0.59658423 0.65322646]
[-0.6783633   0.23285749 -0.6968505]
[-0.56783869 -0.76802646 0.29613269]]
S:
[1.66139597 0.47255546 0.25288375]
Vt:
[[-0.60670895 -0.5020996 -0.61627936]
[-0.53291951 -0.31833703 0.78400149]
[-0.58983137 0.80408801 -0.07444075