Q.N.1: Unit Conversion Program

Taking Length as input

```
Choose conversion type (length/weight/volume): length
Enter the value to convert: 10
Enter the current unit (m/ft for length, kg/lbs for weight, L/gal for volume): m
Converted value: 32.8084
```

Taking weight as Input

```
Choose conversion type (length/weight/volume): weight
Enter the value to convert: 5
Enter the current unit (m/ft for length, kg/lbs for weight, L/gal for volume): kg
Converted value: 11.0231
```

Taking volume as Input

```
Choose conversion type (length/weight/volume): volume
Enter the value to convert: 10
Enter the current unit (m/ft for length, kg/lbs for weight, L/gal for volume): L
Converted value: 2.6417
```

Q.N.2: Mathematical operations on a list of numbers

Finding Average of the numbers

```
Choose operation (sum/avg/max/min): avg
Enter numbers (space-separated): 10 20 30 40 50
Result: 30.0000
```

Finding sum of the numbers

```
Choose operation (sum/avg/max/min): sum
Enter numbers (space-separated): 10 20 30 40
Result: 100.0000
```

Finding Maximum Number

```
Choose operation (sum/avg/max/min): max
Enter numbers (space-separated): 10 20 30 62 89 45
Result: 89.0000
```

Finding Minimum number

```
Choose operation (sum/avg/max/min): min
Enter numbers (space-separated): 10 20 30 62 89 45
Result: 10.0000
```

Q.N.3. List Manipulation Solutions

```
Output of task-1
[1, 3, 5]
['a', 'c', 'e']

Output of task-2
[20, 30, 40]
['apple', 'banana', 'cherry']

Output of task-3
[4, 3, 2, 1]
['z', 'y', 'x']

Output of task-4
[2, 3]
['b']
[]

Output of task-5
[10, 20, 30]
['x', 'y']

Output of task-6
[40, 50]
['b', 'c', 'd']

Output of task-7
[5, 3, 1]
['e', 'c', 'a']
```

Q.N.4. Nested List Solutions

```
Output of Task-1:
[1, 2, 3, 4, 5, 6]
['a', 'b', 'c', 'd']

Output of Task-2:
4
2

Output of Task-3:
21
100

Output of Task-4:
[1, [3, [4], 5]]
[['b', 'c']]

Output of Task-5:
9
-1

Output of Task-6:
1
1

Output of Task-7:
[1, 2, 3, 4, 5, 6]
['x', 'y', 'z']

Output of Task-8:
3.5
25.0
0
```

To - Do - NumPy

Basic Vector and Matrix Operation with Numpy.

```
Empty 2x2 Array:
 [[4.9e-324 9.9e-324]
 [1.5e-323 2.0e-323]]

4x2 Ones Array:
 [[1. 1.]
 [1. 1.]
 [1. 1.]
 [1. 1.]]

3x3 Full Array (Filled with 5s):
 [[5 5 5]
 [5 5 5]
 [5 5 5]]

Zeros Like 'full' (Same Shape as 'full'):
 [[0 0 0]
 [0 0 0]
 [0 0 0]]

Ones Like 'full' (Same Shape as 'full'):
 [[1 1 1]
 [1 1 1]
 [1 1 1]]

NumPy Array from List:
 [1 2 3 4]
```

Problem - 2: Array Manipulation: Numerical Ranges and Array indexing:

```
Array from 10 to 49:
 [10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33
 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49]

3x3 Matrix:
 [[0 1 2]
 [3 4 5]
 [6 7 8]]

3x3 Identity Matrix:
 [[1. 0. 0.]
 [0. 1. 0.]
 [0. 0. 1.]]

Mean of random 30-element array: 0.4956859350058425

Min and Max of 10x10 random matrix: (0.001335103630851986, 0.998660996384065)

Zero array with fifth element as 1:
 [0. 0. 0. 0. 1. 0. 0. 0. 0. 0.]

Reversed array:
 [0 4 0 0 2 1]

5x5 Matrix with a border of 1s and 0s inside:
 [[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.]]

8x8 Checkerboard Pattern:
 [[1. 0. 1. 0. 1. 0. 1. 0.]
 [0. 1. 0. 1. 0. 1. 0. 1.]
 [1. 0. 1. 0. 1. 0. 1. 0.]
 [0. 1. 0. 1. 0. 1. 0. 1.]
 [1. 0. 1. 0. 1. 0. 1. 0.]
 [0. 1. 0. 1. 0. 1. 0. 1.]
 [1. 0. 1. 0. 1. 0. 1. 0.]
 [0. 1. 0. 1. 0. 1. 0. 1.]]
```

Problem - 3: Array Operations

```
Addition of x and y:
 [[ 6  8]
 [10 13]]

Subtraction of x and y:
 [[-4 -4]
 [-4 -3]]

Multiplication of x by 3:
 [[ 3  6]
 [ 9 15]]

Square of x:
 [[ 1  4]
 [ 9 25]]

Dot Product of v and w: 219

Dot Product of x and v:
 [29 77]

Dot Product of x and y:
 [[19 22]
 [50 58]]

Concatenation of x and y along rows:
 [[1 2]
 [3 5]
 [5 6]
 [7 8]]

Concatenation of v and w along columns:
 [[ 9 11]
 [10 12]]
```

Problem - 4: Matrix Operations:

```
A * A_inv (Should be Identity Matrix):
 [[1.00000000e+00 0.00000000e+00]
 [1.77635684e-15 1.00000000e+00]]

Is A * B not equal to B * A?:
 [[ True  True]
 [ True  True]]

Is (A * B)^T equal to B^T * A^T?:
 [[ True  True]
 [ True  True]]

Solution to the linear system Ax = b:
 [ 2.  1. -2.]
```

10.2 Experiment: How Fast is Numpy?

1. Element-wise Addition:

```
Python Lists Addition Time: 0.113466 seconds
NumPy Addition Time: 0.006588 seconds
```

2. Element-wise Multiplication

```
Python Lists Multiplication Time: 0.103454 seconds
NumPy Multiplication Time: 0.009201 seconds
```

3. Dot Product

```
Python Lists Dot Product Time: 0.078112 seconds
NumPy Dot Product Time: 0.001974 seconds
```

4. Matrix Multiplication

```
Python Lists Matrix Multiplication Time: 172.813901 seconds
NumPy Matrix Multiplication Time: 1.352483 seconds
```