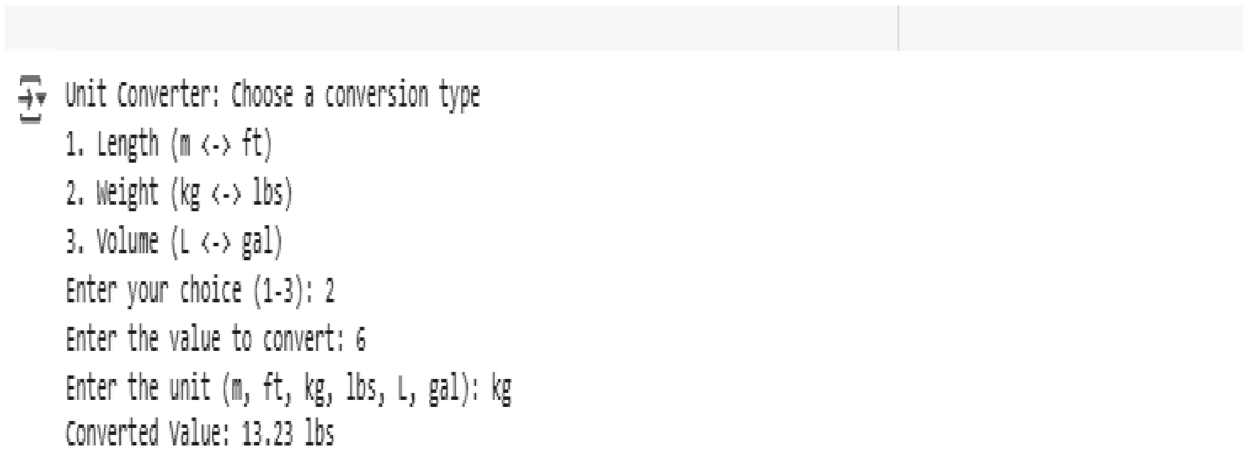
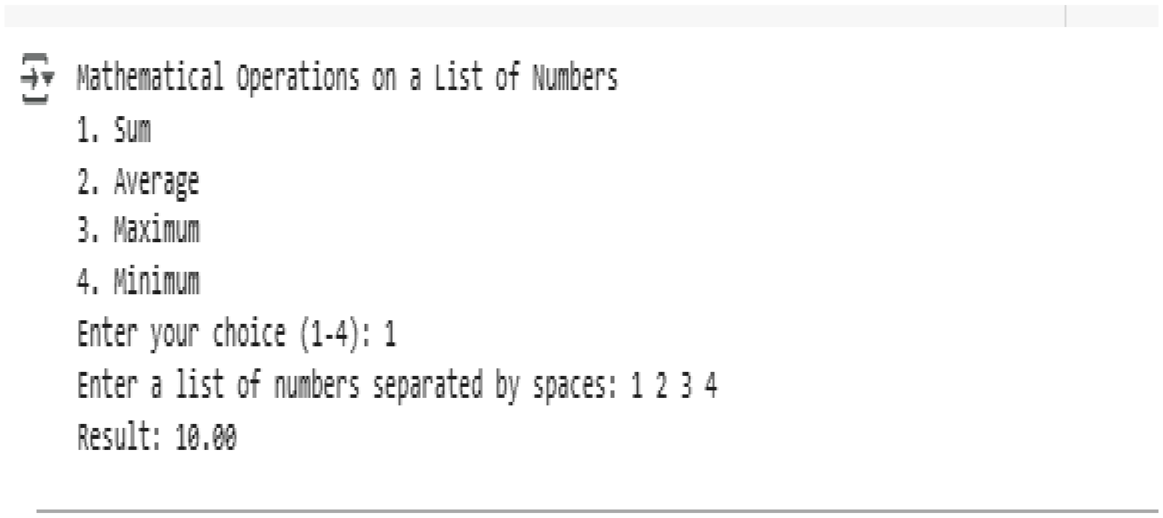
4.1 Exercise on Functions:

Task- 1:



```
Unit Converter: Choose a conversion type
1. Length (m <-> ft)
2. Weight (kg <-> lbs)
3. Volume (L <-> gal)
Enter your choice (1-3): 2
Enter the value to convert: 6
Enter the unit (m, ft, kg, lbs, L, gal): kg
Converted Value: 13.23 lbs
```

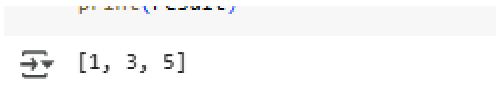
Task- 2:



```
Mathematical Operations on a List of Numbers
1. Sum
2. Average
3. Maximum
4. Minimum
Enter your choice (1-4): 1
Enter a list of numbers separated by spaces: 1 2 3 4
Result: 10.00
```

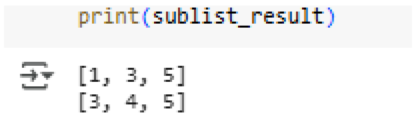
4.2 Exercise on List Manipulation:

1. Extract Every Other Element:



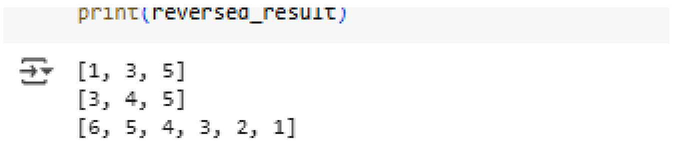
```
print(result)
[1, 3, 5]
```

2. Slice a Sublist:



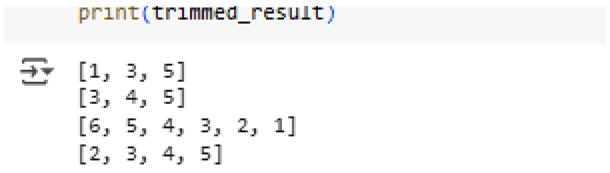
```
print(sublist_result)
[1, 3, 5]
[3, 4, 5]
```

3. Reverse a List Using Slicing:



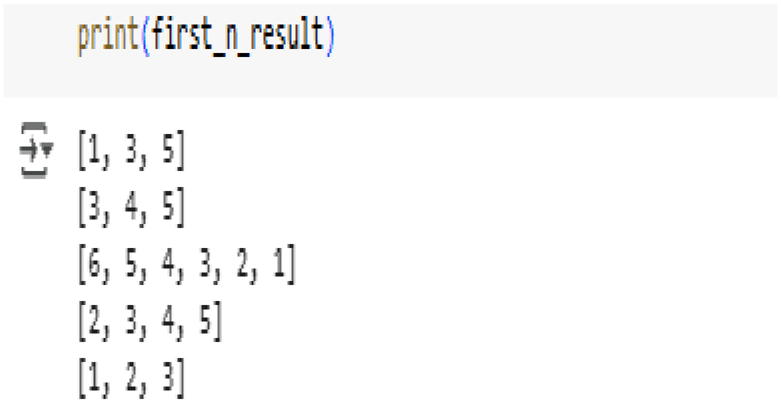
```
print(reversed_result)
[1, 3, 5]
[3, 4, 5]
[6, 5, 4, 3, 2, 1]
```

4. Remove the First and Last Elements:



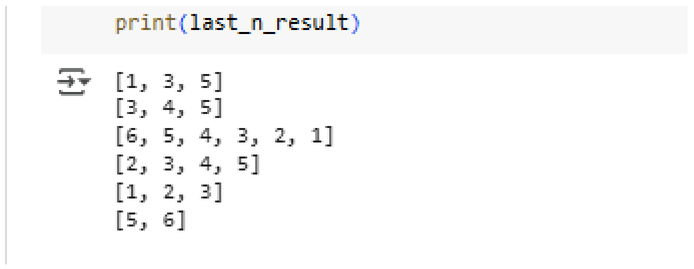
```
print(trimmed_result)
[1, 3, 5]
[3, 4, 5]
[6, 5, 4, 3, 2, 1]
[2, 3, 4, 5]
```

5. Get the First n Elements:



```
print(first_n_result)
[1, 3, 5]
[3, 4, 5]
[6, 5, 4, 3, 2, 1]
[2, 3, 4, 5]
[1, 2, 3]
```

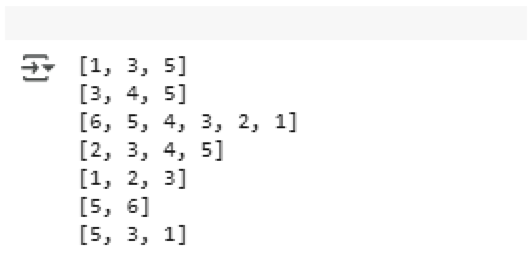
6. Extract Elements from the End:



```
print(last_n_result)
```

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

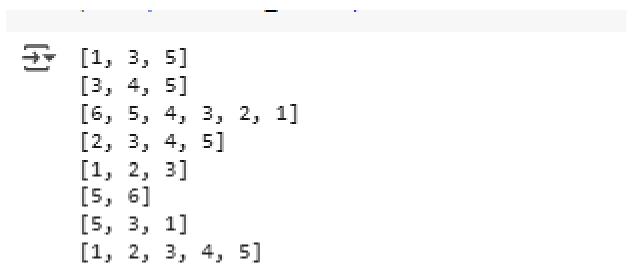
7. Extract Elements in Reverse Order:



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

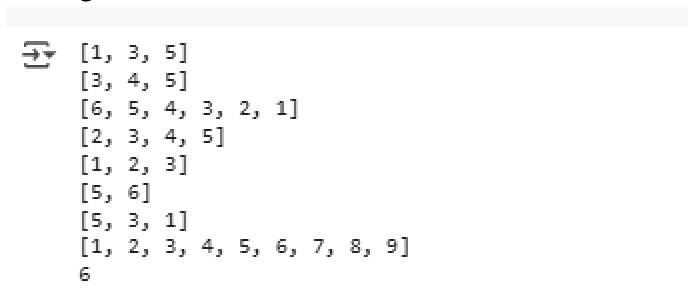
4.3 Exercise on Nested List:

1. Flatten a Nested List:



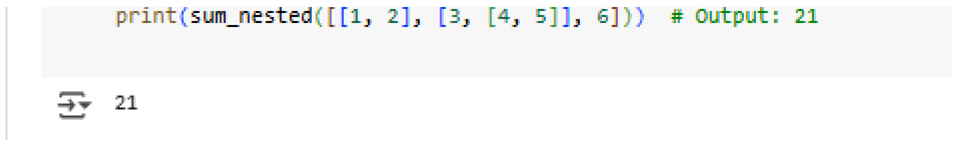
```
[1, 3, 5]
[3, 4, 5]
[6, 5, 4, 3, 2, 1]
[2, 3, 4, 5]
[1, 2, 3]
[5, 6]
[5, 3, 1]
[1, 2, 3, 4, 5]
```

2. Accessing Nested List Elements:



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

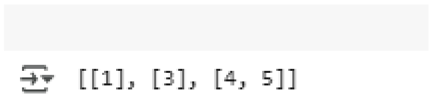
3. Sum of All Elements in a Nested List:



```
print(sum_nested([[1, 2], [3, [4, 5]], 6]))  # Output: 21
```

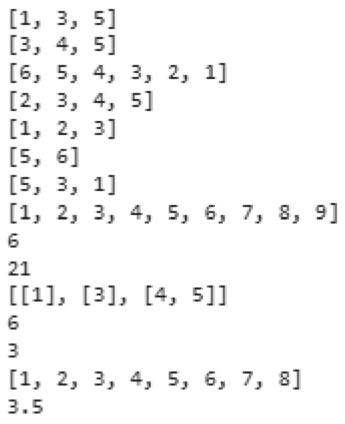
```
21
```

4. Remove Specific Element from a Nested List:



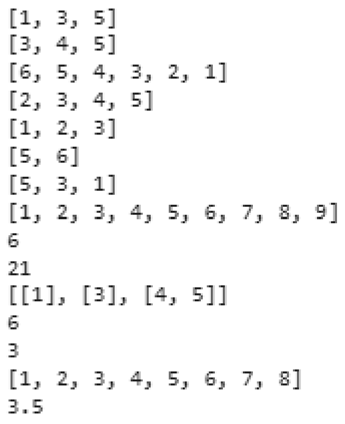
```
[[1], [3], [4, 5]]
```

5. Find the Maximum Element in a Nested List



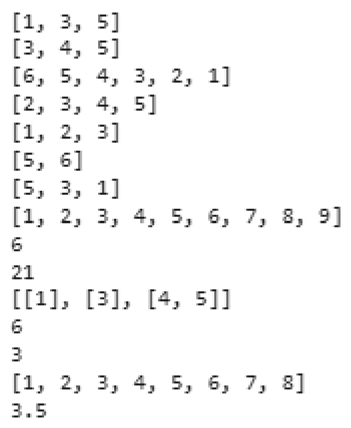
```
[1, 3, 5]
[3, 4, 5]
[6, 5, 4, 3, 2, 1]
[2, 3, 4, 5]
[1, 2, 3]
[5, 6]
[5, 3, 1]
[1, 2, 3, 4, 5, 6, 7, 8, 9]
6
21
[[1], [3], [4, 5]]
6
3
[1, 2, 3, 4, 5, 6, 7, 8]
3.5
```

6. Count Occurrences of an Element in a Nested List:



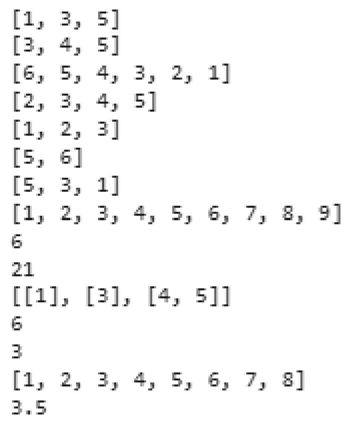
```
[1, 3, 5]
[3, 4, 5]
[6, 5, 4, 3, 2, 1]
[2, 3, 4, 5]
[1, 2, 3]
[5, 6]
[5, 3, 1]
[1, 2, 3, 4, 5, 6, 7, 8, 9]
6
21
[[1], [3], [4, 5]]
6
3
[1, 2, 3, 4, 5, 6, 7, 8]
3.5
```

7. Flatten a List of Lists of Lists:



```
[1, 3, 5]
[3, 4, 5]
[6, 5, 4, 3, 2, 1]
[2, 3, 4, 5]
[1, 2, 3]
[5, 6]
[5, 3, 1]
[1, 2, 3, 4, 5, 6, 7, 8, 9]
6
21
[[1], [3], [4, 5]]
6
3
[1, 2, 3, 4, 5, 6, 7, 8]
3.5
```

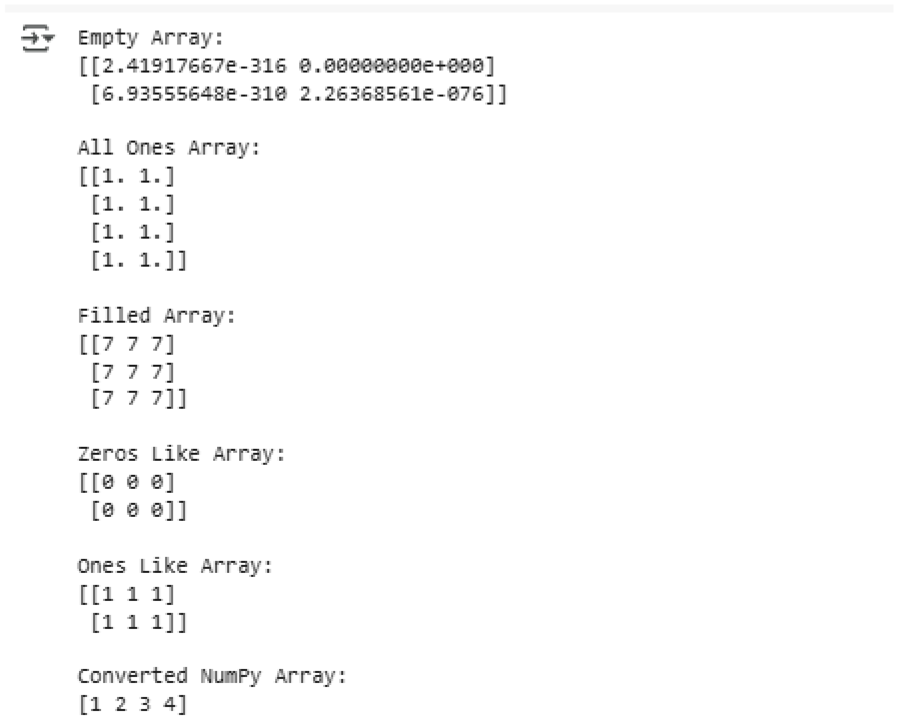
8. Nested List Average:



```
[1, 3, 5]
[3, 4, 5]
[6, 5, 4, 3, 2, 1]
[2, 3, 4, 5]
[1, 2, 3]
[5, 6]
[5, 3, 1]
[1, 2, 3, 4, 5, 6, 7, 8, 9]
6
21
[[1], [3], [4, 5]]
6
3
[1, 2, 3, 4, 5, 6, 7, 8]
3.5
```

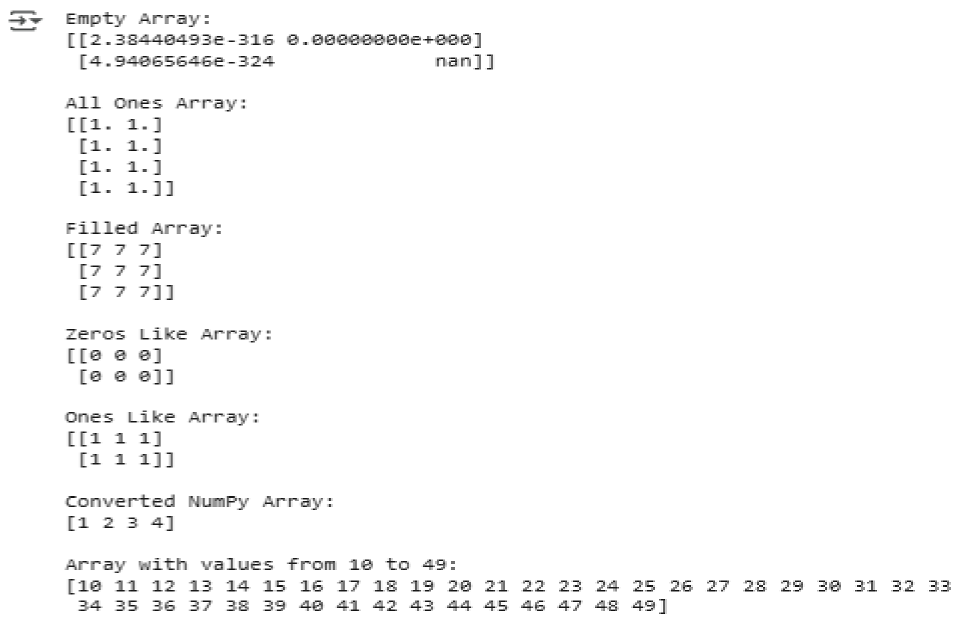
10.1 Basic Vector and Matrix Operation with Numpy

Problem- 1: Array Creation:

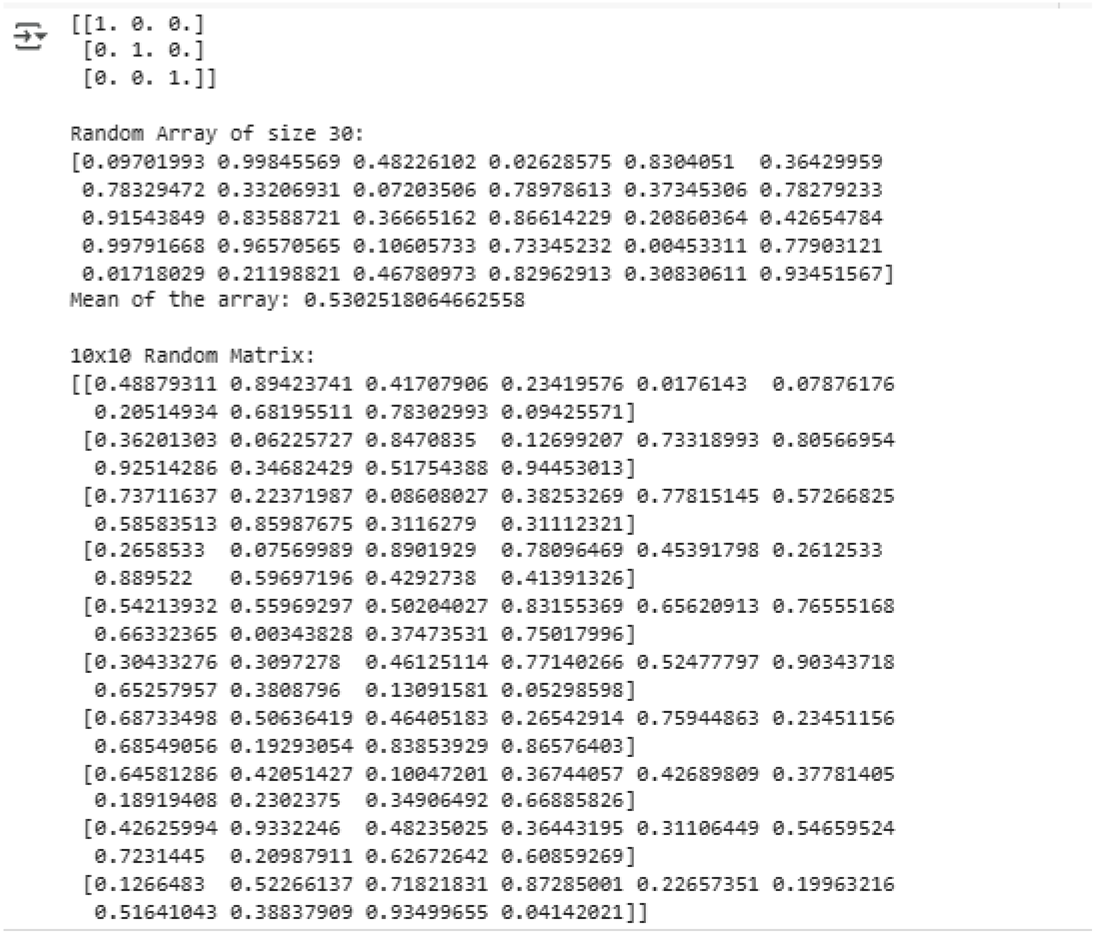


```
Empty Array:
[[2.41917667e-316 0.00000000e+000]
 [6.93555648e-310 2.26368561e-076]]

All Ones Array:
[[1. 1.]
 [1. 1.]
 [1. 1.]
 [1. 1.]]

Filled Array:
[[7 7 7]
 [7 7 7]
 [7 7 7]]

Zeros Like Array:
[[0 0 0]
 [0 0 0]]

Ones Like Array:
[[1 1 1]
 [1 1 1]]

Converted NumPy Array:
[1 2 3 4]
```

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

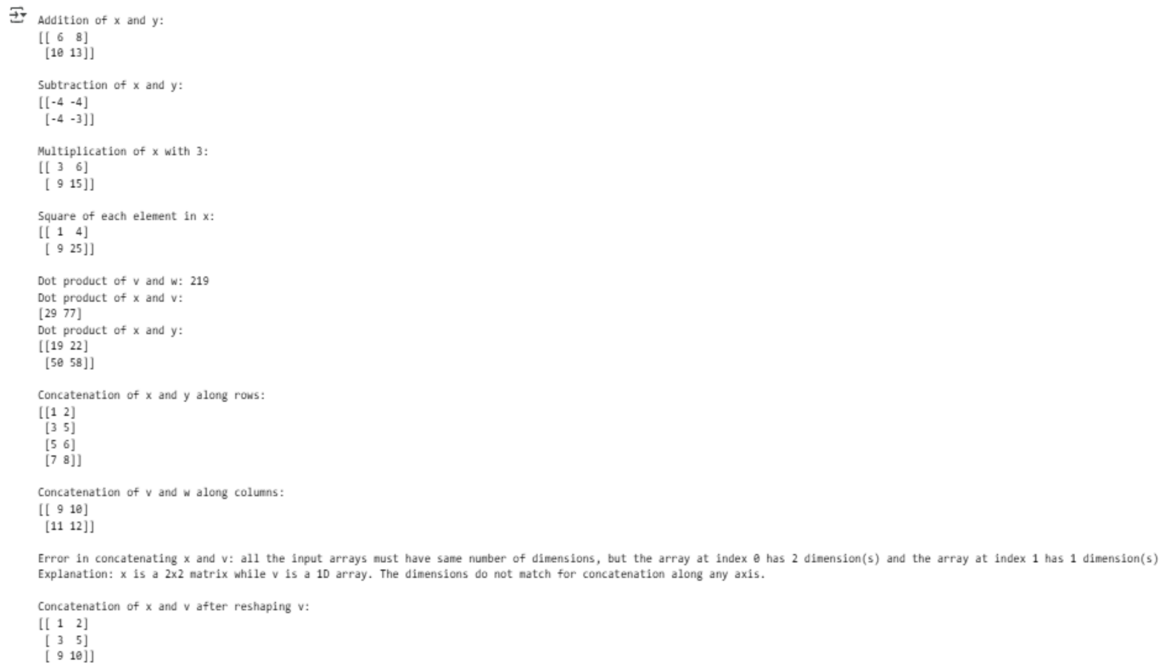


```
Empty Array:
[[2.38440493e-316 0.00000000e+000]
 [4.94065646e-324             nan]]

All Ones Array:
[[1. 1.]
 [1. 1.]
 [1. 1.]
 [1. 1.]]

Filled Array:
[[7 7 7]
 [7 7 7]
 [7 7 7]]

Zeros Like Array:
[[0 0 0]
 [0 0 0]]

Ones Like Array:
[[1 1 1]
 [1 1 1]]

Converted NumPy Array:
[1 2 3 4]

Array with values 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]
```



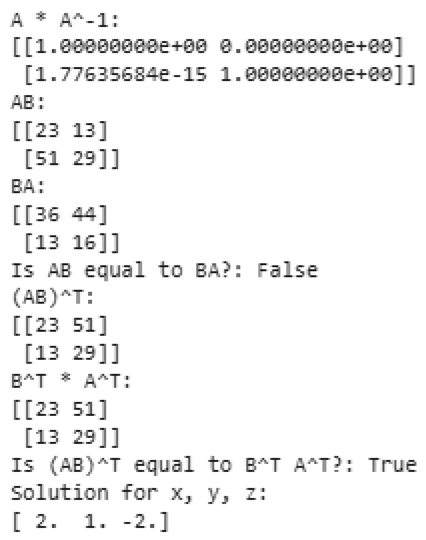
```
[[1. 0. 0.]
 [0. 1. 0.]
 [0. 0. 1.]]

Random Array of size 30:
[0.09701993 0.99845569 0.48226102 0.02628575 0.8304051  0.36429959
 0.78329472 0.33206931 0.07203506 0.78978613 0.37345306 0.78279233
 0.91543849 0.83588721 0.36665162 0.86614229 0.20860364 0.42654784
 0.99791668 0.96570565 0.10605733 0.73345232 0.00453311 0.77903121
 0.01718029 0.21198821 0.46780973 0.82962913 0.30830611 0.93451567]
Mean of the array: 0.5302518064662558

10x10 Random Matrix:
[[0.48879311 0.89423741 0.41707906 0.23419576 0.0176143  0.07876176
  0.20514934 0.68195511 0.78302993 0.09425571]
 [0.36201303 0.06225727 0.8470835  0.12699207 0.73318993 0.80566954
  0.92514286 0.34682429 0.51754388 0.94453013]
 [0.73711637 0.22371987 0.08608027 0.38253269 0.77815145 0.57266825
  0.58583513 0.85987675 0.3116279  0.31112321]
 [0.2658533  0.07569989 0.8901929  0.78096469 0.45391798 0.2612533
  0.889522   0.59697196 0.4292738  0.41391326]
 [0.54213932 0.55969297 0.50204027 0.83155369 0.65620913 0.76555168
  0.66332365 0.00343828 0.37473531 0.75017996]
 [0.30433276 0.3097278  0.46125114 0.77140266 0.52477797 0.90343718
  0.65257957 0.3808796  0.13091581 0.05298598]
 [0.68733498 0.50636419 0.46405183 0.26542914 0.75944863 0.23451156
  0.68549056 0.19293054 0.83853929 0.86576403]
 [0.64581286 0.42051427 0.10047201 0.36744057 0.42689809 0.37781405
  0.18919408 0.2302375  0.34906492 0.66885826]
 [0.42625994 0.9332246  0.48235025 0.36443195 0.31106449 0.54659524
  0.7231445  0.20987911 0.62672642 0.60859269]
 [0.1266483  0.52266137 0.71821831 0.87285001 0.22657351 0.19963216
  0.51641043 0.38837909 0.93499655 0.04142021]]
```

Problem- 3: Array Operations:



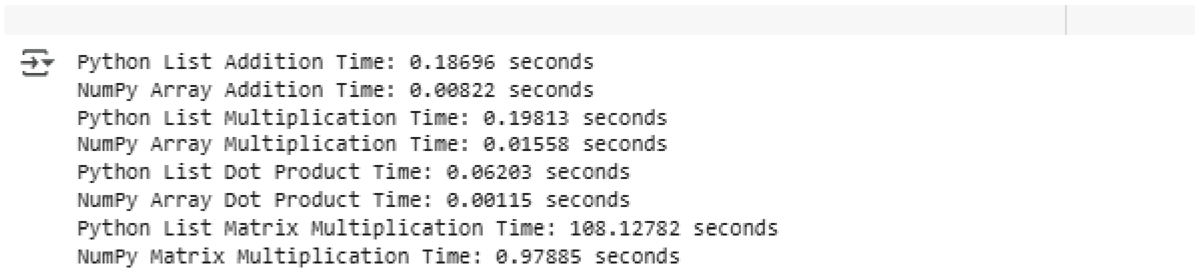
```
Addition of x and y:
[[ 6  8]
 [10 13]]

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

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

Square of each element in 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 10]
 [11 12]]

Error in concatenating x and v: all the input arrays must have same number of dimensions, but the array at index 0 has 2 dimension(s) and the array at index 1 has 1 dimension(s)
Explanation: x is a 2x2 matrix while v is a 1D array. The dimensions do not match for concatenation along any axis.

Concatenation of x and v after reshaping v:
[[ 1  2]
 [ 3  5]
 [ 9 10]]
```

Problem- 4: Matrix Operations:



```
A * A^-1:
[[1.00000000e+00 0.00000000e+00]
 [1.77635684e-15 1.00000000e+00]]
AB:
[[23 13]
 [51 29]]
BA:
[[36 44]
 [13 16]]
Is AB equal to BA?: False
(AB)^T:
[[23 51]
 [13 29]]
B^T * A^T:
[[23 51]
 [13 29]]
Is (AB)^T equal to B^T A^T?: True
Solution for x, y, z:
[ 2.  1. -2.]
```

10.2 Experiment: How Fast is Numpy?



```
Python List Addition Time: 0.18696 seconds
NumPy Array Addition Time: 0.00822 seconds
Python List Multiplication Time: 0.19813 seconds
NumPy Array Multiplication Time: 0.01558 seconds
Python List Dot Product Time: 0.06203 seconds
NumPy Array Dot Product Time: 0.00115 seconds
Python List Matrix Multiplication Time: 108.12782 seconds
NumPy Matrix Multiplication Time: 0.97885 seconds
```
