Exercise on Functions:

Task - 1:

Create a Python program that converts between different units of measurement.

```
Welcome to the Unit Converter!
Available conversions:
1. Length: meters to feet, feet to meters
2. Weight: kilograms to pounds, pounds to kilograms
3. Volume: liters to gallons, gallons to liters
Choose the type of conversion (1, 2, or 3): 2
Enter the value to be converted: 2
Choose conversion (kg_to_lbs or lbs_to_kg): kg_to_lbs
Converted value: 4.40924
```

Task - 2:

Create a Python program that performs various mathematical operations on a list of numbers.

```
Welcome to the Mathematical Operations Program!
Available operations:
1. Sum
2. Average
3. Maximum
4. Minimum
Choose the operation (1, 2, 3, or 4): 1
Enter a list of numbers separated by spaces: 3
Sum: 3.0
```

4.2 Exercise on List Manipulation:

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

1. Extract Every Other Element:

Write a Python function that extracts every other element from a list, starting from the first element.

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

2. Slice a Sublist:

Write a Python function that returns a sublist from a given list, starting from a specified index and ending at another specified index.

```
remove_first_last([1
[2, 3, 4]
```

3. Reverse a List Using Slicing:

Write a Python function that reverses a list using slicing.

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

4. Remove the First and Last Elements:

Write a Python function that removes the first and last elements of a list and returns the resulting sublist.

```
[2, 3, 4]
```

5. Get the First n Elements:

Write a Python function that extracts the first n elements from a list.

```
get_first_n([1,
[1, 2, 3]
```

6. Extract Elements from the End:

Write a Python function that extracts the last n elements of a list using slicing.

```
[4, 5]
```

7. Extract Elements in Reverse Order:

Write a Python function that extracts a list of elements in reverse order starting from the second-to-last element and skipping one element in between.

```
reverse_skip(
[5, 3, 1]
```

4.3 Exercise on Nested List:

1. Flatten a Nested List:

Write a Python function that takes a nested list and flattens it into a single list, where all the elements are in a single dimension.

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

2. Accessing Nested List Elements:

Write a Python function that extracts a specific element from a nested list given its indices.

```
access_nested
6
```

3. Sum of All Elements in a Nested List:

Write a Python function that calculates the sum of all the numbers in a nested list (regardless of depth).

```
sum_nested

21
```

4. Remove Specific Element from a Nested List:

Write a Python function that removes all occurrences of a specific element from a nested list.

```
remove_element([[1, 2], [3, 2],

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

5. Find the Maximum Element in a Nested List:

Write a Python function that finds the maximum element in a nested list (regardless of depth).

```
find_max([[1, 2], [3,

6
```

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

Write a Python function that counts how many times a specific element appears in a nested list.

```
count_occurrences([[1

3
```

7. Flatten a List of Lists of Lists:

Write a Python function that flattens a list of lists of lists into a single list, regardless of the depth.

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

8. Nested List Average:

Write a Python function that calculates the average of all elements in a nested list.

```
average_nested([[1,
3.5
```

Basic Vector and Matrix Operation with Numpy.

Problem - 1: Array Creation:

```
Empty 2x2 Array:
 [[3.12035597e-316 0.00000000e+000]
 [5.34030023e-317 0.00000000e+000]]
---------------
All Ones 4x2 Array:
 [[1. 1.]
 [1. 1.]
 [1. 1.]
 [1. 1.]]
---------------
Array Filled with Value 5:
 [[5 5 5]
 [5 5 5]
 [5 5 5]]
---------------
Zeros Array with Same Shape as Given Array:
 [[0 0]
 [0 0]]
---------------
Ones Array with Same Shape as Given Array:
 [[1 1]
 [1 1]]
---------------
Converted NumPy Array from List:
 [1 2 3 4]
```

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

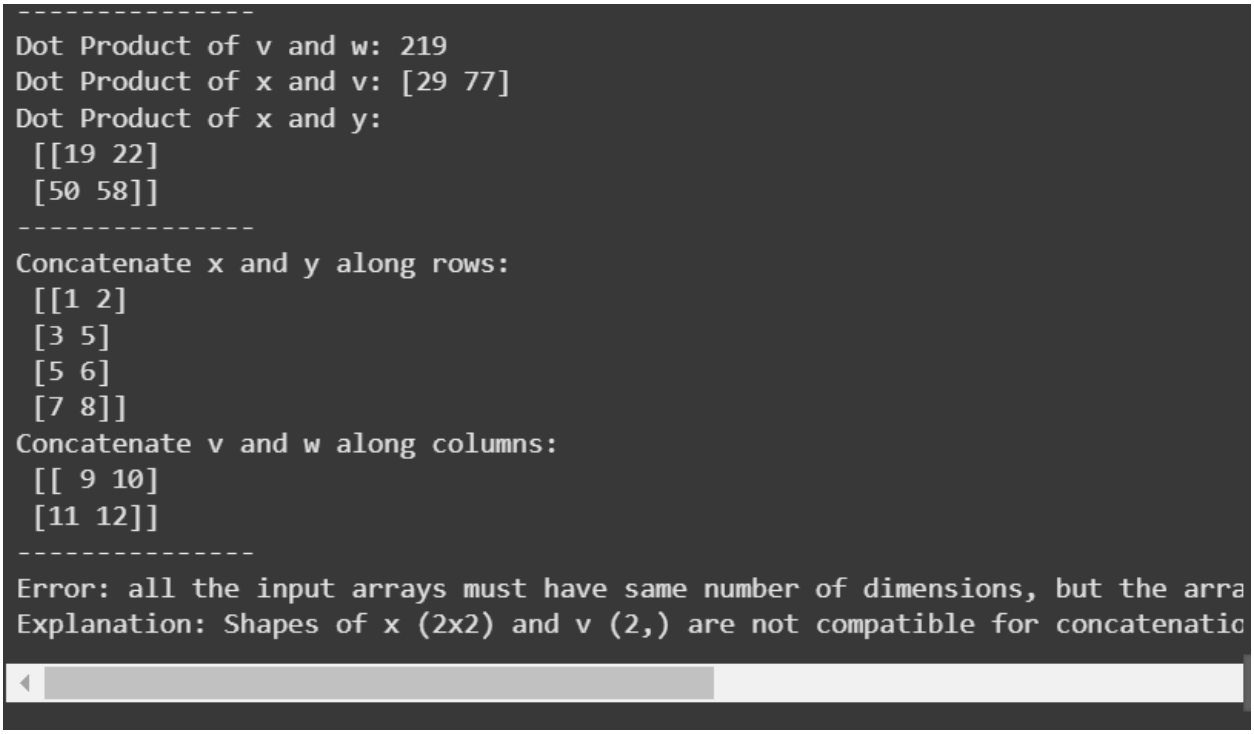
```
Array with Values from 10 to 49:
 [10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33
 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49]
--------------
3x3 Matrix with Values 0 to 8:
 [[0 1 2]
 [3 4 5]
 [6 7 8]]
--------------
3x3 Identity Matrix:
 [[1. 0. 0.]
 [0. 1. 0.]
 [0. 0. 1.]]
--------------
```

```
 ---------------
 Random Array of Size 30:
  [0.50510942 0.61681236 0.25448661 0.2841971  0.43678798 0.88717003
  0.38928236 0.49495408 0.94008693 0.8380229  0.45313628 0.13177205
  0.7386537  0.8884329  0.02001845 0.44028784 0.1738057  0.16394457
  0.34719434 0.41342257 0.13122954 0.91735794 0.358095   0.60672089
  0.69173343 0.98853044 0.8721694  0.97327489 0.30206778 0.86345804]
 Mean of Random Array: 0.5374071835286774
 ---------------
 10x10 Random Array:
  [[0.85524771 0.45155804 0.3112074  0.56122871 0.68350384 0.00729479
   0.04154751 0.15373054 0.73733868 0.94671553]
  [0.19501902 0.10038393 0.24015427 0.33895314 0.34480605 0.32688577
   0.98916903 0.25472996 0.44492496 0.89921157]
  [0.8087475  0.98751545 0.41910868 0.36841182 0.01519977 0.35185461
   0.23574412 0.3798041  0.70795015 0.06051163]
  [0.72812417 0.28397922 0.82774804 0.60259937 0.40674894 0.48234696
   0.04641422 0.84745369 0.49029411 0.81360283]
```

```
Zero Array with 5th Element Replaced:
 [0. 0. 0. 0. 1. 0. 0. 0. 0. 0.]
---------------
Reversed Array:
 [0 4 0 0 2 1]
---------------
2D Array with 1 on Border and 0 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:
 [[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.]
 [1. 0. 1. 0. 1. 0. 1. 0.]]
```

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 2:
 [[ 2  4]
 [ 6 10]]
---------------
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]]
```



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 != BA? True
---------------
(AB)^T:
 [[23 51]
 [13 29]]
B^T A^T:
 [[23 51]
 [13 29]]
Is (AB)^T == B^T A^T? True
---------------
---------------
Solution using Inverse Method: [ 2.  1. -2.]
---------------
Solution using np.linalg.solve: [ 2.  1. -2.]
```