Problem 1: Array Creation

Initialize an empty array with size 2x2

```
import numpy as np
```

```
array = np.empty((2, 2))
print(array)
```

```
[[2.68242994e-315 0.00000000e+000]
 [6.62846392e-310 5.76318485e-317]]
```

Initialize an all one array with size 4x2

```
array = np.ones((4, 2))
print(array)
```

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

Return a new array of given shape and type, filled with fill_value using np.full

+ Code + Text

```
array = np.full((4, 2), 8) #this fills the 4x2 array with 8
print(array)
```

```
[[8 8]
 [8 8]
 [8 8]
 [8 8]]
```

Return a new array of zeros with same shape and type as a given array using np.zers_like

+ Code + Text

```
a = np.array([[1, 2], [4, 5], [3, 6], [7, 8]])
b = np.zeros_like(a)
print(b)
```

```
[[0 0]
 [0 0]
 [0 0]
 [0 0]]
```

Return a new array of zeros with same shape and type as given array using np.ones_like

```
a = np.array([[1, 2], [3, 4], [5, 6], [1, 5]])
n = np.ones_like(a)
print(n)
```

```
[[1 1]
 [1 1]
 [1 1]
 [1 1]]
```

For an existing list new_list = [1, 2, 3, 4] convert to an numpy array using np.array()

```
new_list = [1, 2, 3, 4]
a = np.array(new_list)
print(a)
```

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

Problem 2: Array Manipulation: Numerical Rnages and Array indexing

Create an array with values renging from 10 to 49. Hint:np.arange()

+ Code + Text

```
array = np.arange(10, 50) #here 50 is excluded as this excludes the last number where it ends
print (array)
```

```
[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]
```

+ Code + Text

Create a 3x3 matrix with values ranging from 0 to 8. Hint: look for np.reshape()

+ Code + Text

```
array = np.arange(0, 9). reshape(3, 3)
print(array)
```

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

Create a 3x3 identidy matrix. Hint: np.eye()

```
matrix = np.eye(3)
print(matrix)
```

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

Create a random array of size 30 and find the mean of the array. Hint: Check for np.random.random() and array.mean() function

```
array = np.random.random(30)  #Creates random numbers between 0 to 1
mean_value = array.mean()  #Finds the mean of all elements in the array
print("Array: ", array)
print("MEan: ", mean_value)
```

```
Array:  [0.07942926 0.36162116 0.93722763 0.31614307 0.95061519 0.55672155
 0.79868591 0.05446485 0.76973648 0.85943729 0.26730591 0.60772488
 0.30349418 0.31553909 0.52750939 0.72447172 0.18661104 0.60754867
 0.23044505 0.43583424 0.7702863  0.88952337 0.01179216 0.82853467
 0.22753539 0.49778    0.58726889 0.92031724 0.20316012 0.88088502]
MEan:  0.5235883235553733
```

Create a 10x10 array with random values and find the minimun and maximun values.

```
array = np.random.random(10)
min_value = array.min()
max_value = array.max()
print("Array: ", array)
print("Minimun Value: ", min_value)
print("Maximum Value: ", max_value)
```

```
Array:  [0.34376329 0.73790823 0.98130428 0.64363399 0.03465041 0.44878176
 0.76416756 0.29629041 0.67493242 0.24523846]
Minimun Value:  0.034650410777556595
Maximum Value:  0.981304281932785
```

Create a zero array of size 10 and replace 5th element with 1

```
array = np.zeros(10)
array[4]=1
print(array)
```

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

Reverse an array arr = [1, 2, 0, 0, 4, 0]

```
arr = [1, 2, 0, 0, 4, 0]
reversed_arr = arr[::-1]
print("Reversed array: ", reversed_arr)
```

```
Reversed array:  [0, 4, 0, 0, 2, 1]
```

```
##Or it can also be done in this way
arr = [1, 2, 0, 0, 4, 0]
arr.reverse()
print("Reversed array: ", arr)
```

```
Reversed array:  [0, 4, 0, 0, 2, 1]
```

Create a 2d array with 1 on border and 0 inside

+ Code

```
n, m = 5, 5
array = np.zeros((n, m))
array[0, :] = 1 #top row
array[-1, :] = 1 #bottom row
array[:, 0]=1 #first column
array[:, -1]=1 #last column
print(array)
```

```
[[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.]]
```

Problem 3: Array Operations

+ Code + Text

For the following arrays: x = np.array([[1,2],[3,5]]) and y = np.array([[5,6],[7,8]]); v = np.array([9,10]) and w = np.array([11,12]); Complete all the task using numpy:

Add the two array.

```
x = np.array([[1, 2], [3, 5]])
y = np.array([[5, 6], [7, 8]])
#Adding arrays x and y
add = x+y
print(add)
```

```
[[ 6  8]
 [10 13]]
```

```
v = np.array([9, 10])
w = np.array([11, 12])
#Adding arrays v and w
add = v+w
print(add)
```

```
[20 22]
```

Subtract the two array

```
x = np.array([[1, 2], [3, 5]])
y = np.array([[5, 6], [7, 8]])
#Subtracting arrays x and y
sub = x - y
print(sub)
```

```
[[-4 -4]
 [-4 -3]]
```

```
v = np.array([9, 10])
w = np.array([11, 12])
#Subtracting arrays v and w
sub = v-w
print(sub)
```

```
[-2 -2]
```

+ Code

Multiply the array with any integers of your choice

+ Code

```
mul_x = x*5 #multiplying array x with 5
print(mul_x)
```

```
[[ 5 10]
 [15 25]]
```

```
mul_y = y*3  #Multiplying array x with 3
print(mul_y)
```

```
[[15 18]
 [21 24]]
```

```
mul_v = v*4    #Multiplying array v with 4
print(mul_v)
```

```
[36 40]
```

```
mul_w = w * 10  #Multiplying array w with 10
print(mul_w)
```

```
[110 120]
```

Find the square of each element of the array

```
square_x = x ** 2
print(square_x)
```

```
[[ 1  4]
 [ 9 25]]
```

+ Code

```
square_y = y**2
print(square_y)
```

```
[[25 36]
 [49 64]]
```

+ Code

```
square_v = v**2
print(square_v)
```

```
[ 81 100]
```

```
square_w = w**2
print(square_w)
```

```
[121 144]
```

Find the dot product between: v(and)w ; x(and)v ; x(and)y.

```
dot_vw = np.dot(v, w)   #9x11 + 10x12 = 99 + 120 = 219
print(dot_vw)
```

```
219
```

```
dot_xv = np.dot(x, v)  #First row: 1x9 + 2x10 = 29
print(dot_xv)          #Second row: 3x9 + 5x10 = 77
```

```
[29 77]
```

+ Code + Text

```
dot_xy = np.dot(x, y)
print(dot_xy)
```

```
[[19 22]
 [50 58]]
```

+ Code + Text

Concatenate x(and)y along row and Concatenate v(and)w along column. {Hint:try np.concatenate() or np.vstack() functions.

```
concat_xy = np.concatenate((x, y), axis=0)
print(concat_xy)
```

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

```
concat_vw = np.column_stack((v, w))
print(concat_vw)
```

```
[[ 9 11]
 [10 12]]
```

Concatenate x(and)y; if you get an error, observe and explain why did you get the error?

```
concat_xv = np.concatenate((x, v), axis = 0)
print(concat_xv)

#This raises a valueError because arrays must have same
#number of dimensions, but here we have x as 2D array and v as 1D array.
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
/tmp/ipython-input-3555463519.py in <cell line: 0>()
----> 1 concat_xv = np.concatenate((x, v), axis = 0)
      2 print(concat_xv)

ValueError: 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)
```

Problem 4: Matrix Operations

*For the following arrays: A = np.array([[3,4],[7,8]]) and B = np.array([[5,3],[2,1]]); Prove following with Numpy:

1. Prove A.A–1 = I.
2. Prove AB ≠ BA.
3. Prove (AB)

T = BTAT*

Prove: A.A-1 = I

```
A = np.array([[3, 4], [7, 8]])
B = np.array([[5, 3], [2, 1]])
A_inv = np.linalg.inv(A)
I = np.dot(A, A_inv)
print(I)
```

```
[[1.00000000e+00 0.00000000e+00]
 [1.77635684e-15 1.00000000e+00]]
```

Prove that AB/=BA

```
AB = A @ B
BA = B @ A
print("AB = \n", AB)
print("BA = \n", BA)
```

```
AB =
 [[23 13]
 [51 29]]
BA =
 [[36 44]
 [13 16]]
```

Prove that (AB)^T = B^T.A^T

```
left = (A@B).T
right = B.T @ A.T
print("Left = \n", left)
print("Right = \n", right)
```

```
Left =
 [[23 51]
 [13 29]]
Right =
 [[23 51]
 [13 29]]
```

*Solve the following system of Linear equation using Inverse Methods.

2x – 3y + z = –1 x – y + 2z = –3 3x + y – z = 9

{Hint: First use Numpy array to represent the equation in Matrix form. Then Solve for: AX = B}*

```
A = np.array ([
    [2, -3, 1],
    [1, -1, 2],
    [3, 1, -1]
])
B = np.array([[-1],
              [-3],
              [9]])
A_inv = np.linalg.inv(A)
X = A_inv @ B
print(X)
```

```
[[ 2.]
 [ 1.]
 [-2.]]
```