Problem1: Array Creation

```
import numpy as np
a = np.empty((2,2))
a.shape
```

```
(2, 2)
```

```
import numpy as np
b = np.ones((4,2))
b
```

```
array([[1., 1.],
       [1., 1.],
       [1., 1.],
       [1., 1.]])
```

```
import numpy as np
a = np.full((4,2),fill_value=5.5)
a
```

```
array([[5.5, 5.5],
       [5.5, 5.5],
       [5.5, 5.5],
       [5.5, 5.5]])
```

```
import numpy as np
existing_arr = np.zeros((4,2))
a = np.zeros_like(existing_arr)
a
```

```
array([[0., 0.],
       [0., 0.],
       [0., 0.],
       [0., 0.]])
```

```
import numpy as np
existing_arr = np.ones((4,2))
a = np.ones_like(existing_arr)
a
```

```
array([[1., 1.],
       [1., 1.],
       [1., 1.],
       [1., 1.]])
```

```
import numpy as np
new_list = [1,2,3,4]
```

```
array = np.array(new_list)
array
```

```
array([1, 2, 3, 4])
```

Problem 2: Array Manipulation: Numerical Ranges and Array indexing

```
import numpy as np
a = np.arange (10 ,50)
a
```

```
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])
```

```
import numpy as np
a = np.arange(0,9).reshape(3,3)
a
```

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

```
import numpy as np
a = np.eye(3,3)
a
```

```
array([[1., 0., 0.],
       [0., 1., 0.],
       [0., 0., 1.]])
```

```
import numpy as np
a = np.random.random(30)
m = a.mean()
print(a)
print("The mean is:",m)
```

```
[0.81700416 0.97431255 0.92235548 0.42799125 0.14164381 0.20053312
 0.58786436 0.4621679  0.97871663 0.92196296 0.40186368 0.75462049
 0.50091149 0.71976045 0.96313422 0.06390503 0.61010921 0.88930923
 0.73870881 0.72678688 0.30456119 0.33123911 0.55310483 0.01666787
 0.94093591 0.97355072 0.52298147 0.69084844 0.52378982 0.17742677]
The mean is: 0.5946255949249415
```

```
import numpy as np
a = np.random.random(100).reshape(10,10)
max= (a.max)
min = (a.min)
```

```
import numpy as np
array = np.zeros(10, dtype=int)
array[4] = 1
print(array)
```

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

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

```
[0, 4, 0, 0, 2, 1]
```

```
import numpy as np
array = np.ones((5, 5), dtype=int)
array[1:-1, 1:-1] = 0
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]]
```

```
import numpy as np
matrix = np.zeros((8, 8), dtype=int)
matrix[::2, 1::2] = 1
matrix[1::2, ::2] = 1
print(matrix)
```

```
[[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

```
import numpy as np
x = np.array([[1, 2], [3, 5]])
y = np.array([[5, 6], [7, 8]])
result = x + y
print(result)
```

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

```
result = x - y
print(result)
```

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

```
result = x * 3
print(result)
```

```
[[ 3  6]
 [ 9 15]]
```

```
result = np.square(x)
print(result)
```

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

```
import numpy as np

x = np.array([[1, 2], [3, 5]])
y = np.array([[5, 6], [7, 8]])
v = np.array([9, 10])
w = np.array([11, 12])

dot_vw = np.dot(v, w)
dot_xv = np.dot(x, v)
dot_xy = np.dot(x, y)

print(dot_vw)
print(dot_xv)
print(dot_xy)
```

```
219
[29 77]
[[19 22]
 [50 58]]
```

```
import numpy as np

x = np.array([[1, 2], [3, 5]])
y = np.array([[5, 6], [7, 8]])
v = np.array([9, 10])
w = np.array([11, 12])

concat_xy = np.concatenate((x, y), axis=1)
concat_vw = np.vstack((v, w))

print(concat_xy)
print(concat_vw)
```

```
[[1 2 5 6]
 [3 5 7 8]]
[[ 9 10]
 [11 12]]
```

```
import numpy as np

x = np.array([[1, 2], [3, 5]])
v = np.array([9, 10])

v_reshaped = v.reshape(1, -1)
concat_xv = np.concatenate((x, v_reshaped), axis=0)

print(concat_xv)
```

```
[[ 1  2]
 [ 3  5]
 [ 9 10]]
```

Problem - 4: Matrix Operations:

```
import numpy as np

A = np.array([[3, 4], [7, 8]])
B = np.array([[5, 3], [2, 1]])

A_inv = np.linalg.inv(A)
print("A.A⁻¹ = I:\n", np.dot(A, A_inv))

AB = np.dot(A, B)
BA = np.dot(B, A)
print("AB = BA?", np.array_equal(AB, BA))

AB_transpose = np.transpose(AB)
print("(AB)ᵀ = Bᵀ Aᵀ:\n", AB_transpose, "\n", np.dot(np.transpose(B), np.transpos
```

```
A.A⁻¹ = I:
 [[1.00000000e+00 0.00000000e+00]
 [1.77635684e-15 1.00000000e+00]]
AB = BA? False
(AB)ᵀ = Bᵀ Aᵀ:
 [[23 51]
 [13 29]]
 [[23 51]
 [13 29]]
```

```
import numpy as np

A = np.array([[2, -3, 1], [1, -1, 2], [3, 1, -1]])
B = np.array([-1, -3, 9])

A_inv = np.linalg.inv(A)
solution = np.dot(A_inv, B)

print("Solution (x, y, z):", solution)
```

```
Solution (x, y, z): [ 2.  1. -2.]
```

```
import numpy as np

A = np.array([[2, -3, 1], [1, -1, 2], [3, 1, -1]])
B = np.array([-1, -3, 9])

solution = np.linalg.solve(A, B)

print("Solution (x, y, z):", solution)
```

```
Solution (x, y, z): [ 2.  1. -2.]
```