NumPy Basics HW

In [1]:
```
import numpy as np
```

1. Array Creation:

In [1]:
```
#a. Create a 1D array containing numbers from 0 to 9
```

In [2]:
```
a = np.array([0,1,2,3,4,5,6,7,8,9])
a
```

Out[2]:
```
array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])
```

In [8]:
```
#b. Create a 3x3 matrix with numbers from 1 to 9
```

In [3]:
```
b=np.array([[1,2,3],[4,5,6],[7,8,9]])
b
```

Out[3]:
```
array([[1, 2, 3],
       [4, 5, 6],
       [7, 8, 9]])
```

In [10]:
```
#c. Generate a 1D array of 10 random integers between 50 and 100
```

In [5]:
```
rng = np.random.default_rng()
```

In [93]:
```
c=rng.integers(50,100, size=(1,10))
np.array(c)
c
```

Out[93]:
```
array([[91, 92, 52, 61, 56, 54, 78, 53, 72, 91]])
```

In [41]:
```
#d. Create a 3x4 (row,column) matrix of random floating-point numbers between (
```

In [7]:
```
d= np.random.rand(3,4)
d
```

Out[7]:
```
array([[0.5213726 , 0.76934817, 0.89799248, 0.01347922],
       [0.69785647, 0.60355742, 0.74375589, 0.35340709],
       [0.11780711, 0.87921352, 0.91458159, 0.09581511]])
```

2. Array Indexing

In [34]:
```
#a. From the 1D array created in 1a, extract all even numbers.
```

In [37]:
```
empty=[]
for i in a:
    if i % 2 == 0:
        empty.append(i)
print("Values that are even are", empty)
```

```
Values that are even are [0, 2, 4, 6, 8]
```

In [52]:
```
#b. From the 3x3 matrix created in 1b, extract the second row.
```

In [9]:
```
b[1,]
```

Out[9]:
```
array([4, 5, 6])
```

In [ ]:
```
#c. From the 3x3 matrix, extract the element
#at the third row and second column.
```

In [10]:
```
b[2,1]
```

Out[10]:
```
8
```

In [ ]:
```
#d. From the 3x4 matrix created in 1d, extract all elements greater than 0.5.
```

In [35]:
```
print("Values bigger than .5 =", d[d>.5])
```

```
Values bigger than .5 = [0.5213726  0.76934817 0.89799248 0.69785647 0.6035574
2 0.74375589
 0.87921352 0.91458159]
```

3. Basic Math Operators:

In [38]:
```
#a. Create two arrays A and B of shape (4, 4)
#with random integers between 1 and 10. Compute the element-wise sum and produ
```

In [61]:
```
#first array
A = rng.integers(1,10, size=(4,4))
np.array(A)
A
```

Out[61]:
```
array([[6, 2, 3, 5],
       [1, 5, 4, 8],
       [9, 4, 2, 1],
       [9, 1, 8, 5]])
```

In [63]:
```
#second array
B = rng.integers(1,10, size=(4,4))
np.array(B)
B
```

Out[63]:
```
array([[1, 6, 2, 9],
       [9, 7, 2, 8],
       [5, 6, 9, 7],
       [9, 6, 3, 7]])
```

In [64]:
```
#element-wise sum:
np.add(A,B)
```

Out[64]:
```
array([[ 7,  8,  5, 14],
       [10, 12,  6, 16],
       [14, 10, 11,  8],
       [18,  7, 11, 12]])
```

In [67]:
```
#element-wise prod:
A*B
```

Out[67]:
```
array([[ 6, 12,  6, 45],
       [ 9, 35,  8, 64],
       [45, 24, 18,  7],
       [81,  6, 24, 35]])
```

In [68]:
```
#b. Multiply the 1D array created in 1a by 5.
```

In [69]:
```
a * 5
```

Out[69]:
```
array([ 0,  5, 10, 15, 20, 25, 30, 35, 40, 45])
```

In [70]:
```
#c. Subtract the mean of the 1D array created in 1a from each of its elements
```

In [71]:
```
a.mean()
```

Out[71]:
```
4.5
```

In [72]:
```
a-4.5
```

Out[72]:
```
array([-4.5, -3.5, -2.5, -1.5, -0.5,  0.5,  1.5,  2.5,  3.5,  4.5])
```

4. Basic Statistical Calculations:

In [73]:
```
#a. Compute the mean, median, and standard deviation of the 1D array from 1a
```

In [86]:
```
print(np.mean(a))
print(np.median(a))
print(np.std(a))
```

```
4.5
4.5
2.8722813232690143
```

In [88]:
```
#b. Find the minimum and maximum values of the 3x4 matrix created in 1d
```

In [90]:
```
print(np.max(a))
print(np.min(a))
```

```
9
0
```

In [91]:
```
#c. Find the position (index) of the minimum and maximum values in the 1D array
```

In [95]:
```
print(np.argmax(c))
print(np.argmin(c))
```

```
1
2
```

5. Bonus

In [98]:
```
#a. Compute the dot product of two 1D arrays of length 5.
#Remember, the dot product is the sum of the products of
#corresponding entries of the two sequences of numbers
```

In [99]:
```
e = np.array([23,35,667,34,420])
f = np.array([1,2,3,4,5])

np.dot(e,f)
```

Out[99]:
```
4330
```

In [100…
```
# b. Reshape the 3x3 matrix from 1b into a 1D array of length 9.
```

In [101…
```
b.flatten()
```

Out[101]:
```
array([1, 2, 3, 4, 5, 6, 7, 8, 9])
```