Assignment-1 On Machine Learning

```
pip install numpy
```

```
Requirement already satisfied: numpy in /usr/local/lib/python3.10/dist-packages (1.26.4)
```

```
import numpy as av
```

1. Perform arithmetic operations on NumPy arrays.

```
a=av.array([3,4,5])
b=av.array([6,7,8])
print("Addition : ",av.add(a,b))
print("Substraction : ",av.subtract(a,b))
print("Division : ",av.divide(a,b))
print("Multiplication : ",av.multiply(a,b))
```

```
Addition :  [ 9 11 13]
Substraction :  [-3 -3 -3]
Division :  [0.5        0.57142857 0.625     ]
Multiplication :  [18 28 40]
```

Double-click (or enter) to edit

2. Write a script to compute the mean and standard deviation of a NumPy array.

```
a=av.array([4,5,6,7,8,9,10])
print("Mean Deviation : ",av.mean(a))
print("Standard Deviation : ",av.std(a))
```

```
Mean Deviation :  7.0
Standard Deviation :  2.0
```

3. Create a NumPy array of integers from 10 to 50 with a step size of 5.

```
b=av.arange(10,51,5)
print("Numpy Array : ",b)
```

```
Numpy Array :  [10 15 20 25 30 35 40 45 50]
```

4. Reshape the array into a 2x4 matrix.

```
c=av.array([4,5,6,7,8,9,1,2])
print("Noraml Array : ",c)
c=c.reshape(2,4)
print("Reshaped Array : ",c)
```

```
Noraml Array :  [4 5 6 7 8 9 1 2]
Reshaped Array :  [[4 5 6 7]
 [8 9 1 2]]
```

5. Given the array arr = np.array([[1, 2, 3], [4, 5, 6], [7, 8, 9]])

• Extract the sub-matrix containing the last two rows and first two columns.

• Replace all elements greater than 4 with 0.

```
x= av.array([[1, 2, 3], [4, 5, 6], [7, 8, 9]])
print(x[1:,:2])
x[x>4]=0
print("Modified Array : ",x)
```

```
[[4 5]
 [7 8]]
Modified Array :  [[1 2 3]
 [4 0 0]
 [0 0 0]]
```

6. Create two 1D arrays

• a = np.array([1, 2, 3])

• b = np.array([4, 5, 6])

• Perform element-wise addition, multiplication, and division.

```
x= av.array([1, 2, 3])
y=av.array([4, 5, 6])
#using Normal operations
print(x+y)
print(x*y)
print(x/y)
print(" ")
#using Methods
print("Addition : ",av.add(x,y))
print("Substraction : ",av.subtract(x,y))
print("Division : ",av.divide(x,y))
print("Multiplication : ",av.multiply(x,y))
```

```
[5 7 9]
[ 4 10 18]
[0.25 0.4  0.5 ]

Addition :  [5 7 9]
Substraction :  [-3 -3 -3]
Division :  [0.25 0.4  0.5 ]
Multiplication :  [ 4 10 18]
```

7. Create a random NumPy array of size (4, 4). Compute

• Mean of each column.

• Maximum value of the entire array.

• Sum of all elements.

```
arr=av.random.randint(10,size=(4,4))
print("Random array: ",arr)
print("   ")
print("Mean : ",av.mean(arr,axis=0))
print("Maximum : ",av.max(arr))
print("Sum : ",av.sum(arr))
```

```
Random array:  [[1 4 8 9]
 [0 0 6 0]
 [1 7 9 3]
 [8 7 9 3]]

Mean :  [2.5  4.5  8.   3.75]
Maximum :  9
Sum :  75
```

8. Given the array arr = np.array([1, 2, 3, 4]), add 10 to each element and multiply the result by 2.

Double-click (or enter) to edit

```
z = av.array([1, 2, 3, 4])
z=z+10
z=z*2
print("Array after operations : ",z)
```

```
Array after operations :  [22 24 26 28]
```

9. Perform advanced slicing and indexing on a 3D array. Create a 3D NumPy array arr of shape (3, 4, 5) with random integers ranging from 1 to 100. Perform the following:

• Extract the second "layer" (i.e., the second 2D array) in the first dimension.

• Extract the first two rows and the last three columns from each "layer."

• Replace all elements divisible by 3 with -1.

```
x =av.random.randint(1,101,(3,4,5))
print("Random Array : ",x)
print("---------- ")
print("Second Layer of Array :",x[1])
print("---------- ")
print("Extraction of Array :",x[:,0:2,2:])
```

```
print("--------- ")
x[x%3==0]=-1
print("Divisible by 3 with -1",x)
```

```
Random Array :  [[[65 14  2 31 23]
  [29 68 69 30 42]
  [63 20 91 22 85]
  [70 45  4 79 12]]

 [[75 67 79 69 66]
  [84 75 59 39 32]
  [89 68 37 66 28]
  [94 19 13 56 42]]

 [[90 39 67 26 87]
  [92 92 85 94 27]
  [36 57 93 62 64]
  [ 5 26 36 79  5]]]
----------
Second Layer of Array : [[75 67 79 69 66]
 [84 75 59 39 32]
 [89 68 37 66 28]
 [94 19 13 56 42]]
----------
Extraction of Array : [[[ 2 31 23]
  [69 30 42]]

 [[79 69 66]
  [59 39 32]]

 [[67 26 87]
  [85 94 27]]]
---------
Divisible by 3 with -1 [[[65 14  2 31 23]
  [29 68 -1 -1 -1]
  [-1 20 91 22 85]
  [70 -1  4 79 -1]]

 [[-1 67 79 -1 -1]
  [-1 -1 59 -1 32]
  [89 68 37 -1 28]
  [94 19 13 56 -1]]

 [[-1 -1 67 26 -1]
  [92 92 85 94 -1]
  [-1 -1 -1 62 64]
  [ 5 26 -1 79  5]]]
```

10. Work with broadcasting and advanced operations in NumPy. Create a NumPy array arr of shape (3, 3) with random integers between 1 and 20. Perform the following:

• Subtract the mean of each row from its respective elements (row-wise normalization).

• Create a new array by squaring all the elements in arr.

• Find the indices of the top 3 maximum values in the entire array.

```
n = av.random.randint(1,21,(3,3))
print(n)
print("---------- ")
for i in range(3):
  n[i] = n[i]-av.mean(n[i])
print("Normalisation :\n",n)
print("---------- ")
print("Squared array :\n",av.square(n))
print("---------- ")
f = n.flatten()
t = av.argsort(f)[-3:]
row,col = av.unravel_index(t,n.shape)
print("Indices of top 3 maximum values :\n",list(zip(row,col)))
```

```
[[19 13  1]
 [19  5 12]
 [ 7 20  7]]
----------
Normalisation :
 [[  8   2 -10]
 [  7  -7   0]
 [ -4   8  -4]]
----------
Squared array :
 [[ 64   4 100]
 [ 49  49   0]
 [ 16  64  16]]
```

```
----------
Indices of top 3 maximum values :
 [(1, 0), (0, 0), (2, 1)]
```