ASSIGNMENT 3 – EDS

Name: Ashraf Shaikh

Roll No:160 **Batch:**A3

PRN:202201050001

Prepare/Take dataset for any real life application. Read a dataset into an array. Perform following operations on it as:

- Perform all matrix operations

- Horizontal and vertical stacking of Numpy Arrays

- Custom sequence generation

- Arithmetic and Statistical Operations, Mathematical Operations, Bitwise Operators

- Copying and viewing arrays

- Data Stacking, Searching, Sorting, Counting, Broadcasting.

```
import numpy as np
arr=np.loadtxt("/content/salary.csv",delimiter=',',dtype=str,s
kiprows=1) print(arr) sal=[] exp=[] for i in arr:
  sal.append(int(i[1]))
exp.append(int(i[2]))
print(sal) print(exp)

#converting list to numpyarray
arr_sal=np.array(sal)
arr_exp=np.array(exp)
#displaying the array
print("A1:",arr_sal) print("A2:",arr_exp)
```

output:

```
 [['raj' '25000' '12000']
 ['vijay' '20000' '15000']
 ['kishor' '15000' '7000']
['kiran' '18000' '8000']
 ['sahil' '21000' '10000']
 ['priyank' '30000' '20000']
 ['ramesh' '28000' '25000']
 ['Aditya' '23000' '21000']
 ['Shardul ' '12000' '11000']
 ['om' '13000' '11000']
 ['Jaggy' '45000' '12000']
 ['Ishwar' '98000' '21000']
 ['Ashraf' '87000' '52000']
 ['Aniruddha' '56999' '51500']]
[25000, 20000, 15000, 18000, 21000, 30000, 28000, 23000, 12000, 13000,
45000, 98000, 87000, 56999]
[12000, 15000, 7000, 8000, 10000, 20000, 25000, 21000, 11000, 11000,
12000, 21000, 52000, 51500]
A1: [25000 20000 15000 18000 21000 30000 28000 23000 12000 13000 45000
98000
 87000 56999]
A2: [12000 15000  7000  8000 10000 20000 25000 21000 11000 11000 12000
21000
 52000 51500]
```

1.ALL MATRIX OPERATIONS

1.Addition

```
#Addition resultarray=np.add(arr_sal,arr_exp)
print("\nAddition using Numpy
Function:\n",resultarray)
```

Output:

```
Addition using Numpy Function:
 [ 37000  35000  22000  26000  31000  50000  53000  44000  23000  24000
  57000 119000 139000 108499]
```

2.Substraction

```
#Substraction
resultarray=np.subtract(arr_sal,arr_exp)
print("\nsubstraction using Numpy
Function:\n",resultarray)
```

Output:

substraction using Numpy Function:

[13000 5000 8000 10000 11000 10000 3000 2000 1000 2000 33000 77000

35000 5499]

```


#Multiplication
```

```
resultarray=np.multiply(arr_sal,arr_exp)
print("\nmultiplication using Numpy Function:\n",resultarray)
```

multiplication using Numpy Function:

[300000000 300000000 105000000 144000000 210000000 600000000

 700000000 483000000 132000000 143000000 540000000 2058000000

4524000000 2935448500]

4.division

```
#Division resultarray=np.subtract(arr_sal,arr_exp)
print("\nDivision using Numpy
Function:\n",resultarray)
```

Output:

```
Division using Numpy Function:
 [13000  5000  8000 10000 11000 10000  3000  2000  1000  2000 33000
77000
 35000  5499]
```

5.Mod of two array

```
# mod of two array
resultarray=np.mod(arr_sal,arr_exp) print("\nthe
mod of two array is:\n",resultarray)
```

Output:

```
the mod of two array is:
 [ 1000  5000  1000  2000  1000 10000  3000  2000  1000  2000  9000
14000
 35000  5499
```

6.Dot Product

```
# The dot function of two array
resultarray=np.dot(arr_sal,arr_exp)
print("\nThe dot function of two array is
:\n",resultarray)
```

```
The dot function of two array is :
13174448500
```

2.HORIZONTAL AND VERTICAL STACKING OF NUMPY ARRAYS

1.Horizontal stacking

```
#Horizontal stacking
resultarray=np.hstack((arr_sal,arr_exp))
resultarray
```

Output: array([25000, 20000, 15000, 18000, 21000, 30000, 28000,

23000, 12000,   13000, 45000, 98000, 87000, 56999, 12000, 15000,

7000, 8000,

      10000, 20000, 25000, 21000, 11000, 11000, 12000, 21000, 52000,

51500])

2.Vertical stacking

```
#Vertical stacking
resultarray=np.vstack((arr_sal,arr_exp))
resultarray
```

Output:

```
array([[25000, 20000, 15000, 18000, 21000, 30000, 28000, 23000, 12000,
13000, 45000, 98000, 87000, 56999], [12000, 15000, 7000, 8000, 10000,
20000, 25000, 21000, 11000, 11000, 12000, 21000, 52000, 51500]])
```

3.CUSTOM SEQUENCE GENERATION

1.Range.

```
import numpy as np
nparray=np.arange(0,12,1).reshape(3,4)
nparray
```

Output:

```
array([[ 0, 1, 2, 3], [ 4, 5, 6, 7], [ 8, 9, 10, 11]])
```

2.Linearly seperable

```
nparray=np.linspace(start=0,stop=24,num=12).reshape(3,4)
nparray
```

Output:

array([[0.   , 2.18181818, 4.36363636, 6.54545455],

      [8.72727273, 10.90909091, 13.09090909, 15.27272727],

```
[17.45454545, 19.63636364, 21.81818182, 24.        ]])
```

4.STATISTICAL OPERATIONS,MATHEMATICAL OPERATIONS,BINARY OPERATORS.

1.Statistical operation for salary

```
#Statistical operations for salary  #standard
deviation print("The standard deviation
is\n",np.std(arr_sal))
#minimum print("The minimum value of salary
is\n",np.min(arr_sal))
#summation print("The summation of all salaries
is\n",np.sum(arr_sal))
#Median print("The medain of the salary
is\n",np.median(arr_sal))
#mean print("The mean of the salary
is\n",np.mean(arr_sal))
```

Output:

```
The standard deviation is
 26319.018886888003
The minimum value of salary is
 12000
The summation of all salaries is
 491999
The medain of the salary is
 24000.0
The mean of the salary is
35142.78571428572
```

2.Statistical operation for expense

```
#for expense #standard deviation print("The
standard deviation is\n",np.std(arr_exp))
#minimum print("The minimum value of expense
is\n",np.min(arr_exp))
#summation print("The summation of
everyone's expense is\n",np.sum(arr_exp))
#Median print("The medain of the expense
is\n",np.median(arr_exp)) #mean
```

```
print("The mean of the expense is\n",np.mean(arr_exp))
```

Output:

```
The standard deviation is
 14067.12225621959
The minimum value of expense is
 7000
The summation of everyone's expense is
 276500
The medain of the expense is
 13500.0
The mean of the expense is
 19750.0
```

3.Bitwise

```
array1=np.array([1,2,3],dtype=np.uint8)
array2=np.array([4,5,6])
# AND
resultarray=np.bitwise_and(array1,array2)
print(resultarray)
# OR
resultarray=np.bitwise_or(array1,array2)
print(resultarray)
#LeftShift
resultarray=np.left_shift(array1,2)
print(resultarray) #RightShift
resultarray=np.right_shift(array1,2)
print(resultarray)
```

Output:

```
[0 0 2]

[5 7 7]

[ 4  8 12]

[0 0 0]
```

5.COPYING AND VIEWING ARRAYS

1.copy

```
#Copying two arrays
array1=np.arange(1,10)
print(array1)
newarray=arr_sal.copy()
print(newarray)
##modification in Original Array
arr_sal[0]=100 print(arr_sal)
print(newarray)
```

Output:

```
[1 2 3 4 5 6 7 8 9]
[25000 20000 15000 18000 21000 30000 28000 23000 12000 13000 45000
98000
 87000 56999]
[  100 20000 15000 18000 21000 30000 28000 23000 12000 13000 45000
98000
 87000 56999]
[25000 20000 15000 18000 21000 30000 28000 23000 12000 13000 45000
98000
 87000 56999]
```

2.Viewing

```
newarray=arr_sal.view()
print(newarray)
##modification in Original
Array arr_sal[0]=100
print(arr_sal) print(newarray)
```

Output:

```
[  100 20000 15000 18000 21000 30000 28000 23000 12000 13000 45000
98000
 87000 56999]
[  100 20000 15000 18000 21000 30000 28000 23000 12000 13000 45000
98000
 87000 56999]
[  100 20000 15000 18000 21000 30000 28000 23000 12000 13000 45000
98000
 87000 56999]
```

6.SEARCHING,SORTING,STACKING,BROADCASTING

1.searching

```
 x=np.where(arr_sal%2==0)
print(x
```

Output:

```
(array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12]),)
```

2.sorting

```
np.sort(arr_sal,axis=0)#Horizontally Sort
```

Output:

```
array([[ 0. , 2.18181818, 4.36363636, 6.54545455], [ 8.72727273,
10.90909091, 13.09090909, 15.27272727], [17.45454545, 19.63636364,
21.81818182, 24. ]])
```

3.stacking

```
#Stacking z=np.stack((arr_sal,arr_exp),axis=0)
print(z)
```

Output:

```
[[  100 20000 15000 18000 21000 30000 28000 23000 12000 13000 45000
98000
  87000 56999]
 [12000 15000  7000  8000 10000 20000 25000 21000 11000 11000 12000
21000
  52000 51500]]
```

4.Broadcasting

```
#Broadcasting from numpy import array a =
array([1.0, 2.0, 3.0]) b = array([2.0, 2.0,
2.0]) a * b
```

Output:

```
array([2., 4., 6.])
```