NAME: GUNGUN DHIRAJ WAHANE

ROLL NO: 666

BATCH: F4

PRN: 202201050029

ASSIGNMENT 3:

```
import numpy as np
array3=np.loadtxt('/content/marks2.csv',delimiter=',', dtype=str,
skiprows=1)
print(array3)

physics=[]
sic=[]
EEE=[]
EGR=[]
EDS=[]
for i in array3:
  physics.append(int(i[1]))
  sic.append(int(i[2]))
  EEE.append(int(i[3]))
  EGR.append(int(i[4]))
  EDS.append(int(i[5]))
print(physics)
print(sic)
print(EEE)
print(EGR)
print(EDS)

#converting list to array
arr_physics = np.array(physics)
arr_sic = np.array(sic)
arr_EEE = np.array(EEE)
```

```
arr_EGR = np.array(EGR)
arr_EDS = np.array(EDS)

#displaying array
print("array 1:" ,arr_physics)
print("array 2:" ,arr_sic)
print("array 3:" ,arr_EEE)
print("array 4:" ,arr_EGR)
print("array 5:" ,arr_EDS)
```

OUTPUT:

```
[['Kaustubh' ' 88' ' 91 ' ' 93 ' ' 99 ' ' 95']
 ['Gungun' ' 91' ' 92 ' ' 93 ' ' 97 ' ' 95']
 ['Rutuja' ' 89' ' 91 ' ' 92 ' ' 94 ' ' 95']
 ['Siddhi' ' 89' ' 93 ' ' 94 ' ' 88 ' ' 96']
 ['Sandesh' ' 77 ' '87' ' 89 ' ' 85 ' ' 86']
 ['Rohan' ' 92 ' ' 89 ' ' 87 ' ' 96 ' ' 95']
 ['Yash' ' 75 ' ' 89 ' ' 84 ' ' 86 ' ' 85 ']
 ['muskan ' '65 ' ' 74 ' ' 84 ' ' 94 ' ' 89 ']
 ['Tanmay' ' 95 ' ' 89 ' ' 87 ' ' 84' ' 86']
 ['Sanket' ' 87' ' 85 ' ' 86 ' ' 89 ' ' 88']]
[88, 91, 89, 89, 77, 92, 75, 65, 95, 87]
[91, 92, 91, 93, 87, 89, 89, 74, 89, 85]
[93, 93, 92, 94, 89, 87, 84, 84, 87, 86]
[99, 97, 94, 88, 85, 96, 86, 94, 84, 89]
[95, 95, 95, 96, 86, 95, 85, 89, 86, 88]
array 1: [88 91 89 89 77 92 75 65 95 87]
array 2: [91 92 91 93 87 89 89 74 89 85]
array 3: [93 93 92 94 89 87 84 84 87 86]
array 4: [99 97 94 88 85 96 86 94 84 89]
array 5: [95 95 95 96 86 95 85 89 86 88]
```

#1. PERFORM ALL MATRIX OPERATION

```
#1. matrix
#addition
resultarray= arr1+arr2
print("using operator: \n", resultarray)


resultarray=np.add(arr1,arr2)
print("using numpy: \n", resultarray)
```

OUTPUT:

```
using operator:
 [[183 186 184 185 163 187 160 154 181
                   175]

 [186 187 186 189 173 184 174 163 175
                   173]

 [190 190 190 192 172 190 170 178 172
                   176]]
using numpy:
 [[183 186 184 185 163 187 160 154 181
                   175]

 [186 187 186 189 173 184 174 163 175
                   173]

 [190 190 190 192 172 190 170 178 172
                   176]]
```

```
#substraction
resultarray= arr1-arr2
print("using operator: \n", resultarray)


resultarray=np.subtract(arr1,arr2)
print("using numpy: \n", resultarray)
```

OUTPUT:

```
using operator:
 [[[ -7  -4  -6  -7  -9  -3 -10 -24   9
                    -1]]

  [[ -4  -3  -4  -3   1  -6   4 -15   3
                    -3]]

  [[  0   0   0   0   0   0   0   0   0
                     0]]]
using numpy:
 [[[-7  -4  -6  -7  -9  -3 -10 -24   9
                    -1]]

  [[ -4  -3  -4  -3   1  -6   4 -15   3
                    -3]]

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

```
#multiplication
resultarray= arr1*arr2
print("using operator: \n", resultarray)


resultarray=np.multiply(arr1,arr2)
print("using numpy: \n", resultarray)
```

OUTPUT:

```
using operator:
 [[[8360 8645 8455 8544 6622 8740 6375
                  5785 8170 7656]]

  [[8645 8740 8645 8928 7482 8455 7565
                  6586 7654 7480]]

  [[9025 9025 9025 9216 7396 9025 7225
                  7921 7396 7744]]]
using numpy:
 [[[8360 8645 8455 8544 6622 8740 6375
                  5785 8170 7656]]

  [[8645 8740 8645 8928 7482 8455 7565
                  6586 7654 7480]]

  [[9025 9025 9025 9216 7396 9025 7225
                  7921 7396 7744]]]
```

```
#division
resultarray= arr1/arr2
print("using operator: \n", resultarray)


resultarray=np.divide(arr1,arr2)
print("using numpy: \n", resultarray)
```

OUTPUT:

```
using operator:
 [[[0.92631579 0.95789474 0.93684211
                  0.92708333 0.89534884 0.96842105
                  0.88235294 0.73033708 1.10465116
                  0.98863636]]

                [[0.95789474 0.96842105 0.95789474
                  0.96875    1.01162791 0.93684211
                  1.04705882 0.83146067 1.03488372
                  0.96590909]]

                [[1.         1.         1.
                  1.         1.         1.
                  1.         1.         1.
                  1.        ]]]
using numpy:
 [[[0.92631579 0.95789474 0.93684211
                  0.92708333 0.89534884 0.96842105
                  0.88235294 0.73033708 1.10465116
                  0.98863636]]

                [[0.95789474 0.96842105 0.95789474
                  0.96875    1.01162791 0.93684211
                  1.04705882 0.83146067 1.03488372
                  0.96590909]]

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

```
 #mod

resultarray= arr1%arr2
print("using operator: \n", resultarray)


resultarray=np.mod(arr1,arr2)
print("using numpy: \n", resultarray)
```

OUTPUT:

```
using operator:
 [[[88 91 89 89 77 92 75 65  9 87]]

  [[91 92 91 93  1 89  4 74  3 85]]

  [[ 0  0  0  0  0  0  0  0  0  0]]]
using numpy:
 [[[88 91 89 89 77 92 75 65  9 87]]

  [[91 92 91 93  1 89  4 74  3 85]]

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

#2 STACKING

```
# horizontal stacking OR #column stacking


import numpy as np
arr1=np.array([physics])
arr2=np.array([EDS])
arr3=np.hstack((arr1,arr2))
print("horizontal stacking: \n",arr3)
```

OUTPUT:

horizontal stacking:
[[88 91 89 89 77 92 75 65 95 87 95 95 95 96 86 95 85 89 86 88]]

```
# vertical stacking  OR  #row stacking


arr3=np.vstack((arr1,arr2))
print( "vertical stacking: \n" ,arr3)
```

OUTPUT:

```
vertical stacking:
 [[88 91 89 89 77 92 75 65 95 87]
 [95 95 95 96 86 95 85 89 86 88]]
```

#3 custom range sequence

```
#range
nparray=np.arange(0,12,1).reshape(3,4)
nparray
```

output:

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

```
#empty array
nparray=np.empty((3,3),int)
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. ]])
```

```
#empty array
nparray=np.empty((3,3),int)
nparray
```

output:

```
array([[ 140064252272736, 0, 72368284412984948 37], [ 49,
34968880, 140064843160576], [73094688341692537 34,
11422086359395, 160]])
```

```
#empty like some other array

nparray=np.empty_like(arr1)
nparray
```

output:

```
array([[         34451984,                0,                0,
                       0,     140063178489856],
       [8319683848551211643, 3180222411935070754, 8391722768137527840,
        7959390389040738153, 2318283116239266420],
       [8675445202132104482, 7957695011165139568, 2318365875964093043,
        7233184988217307170,                128]])
```

#4. ARITHMETIC AND STATISTICAL OPERATONS, MATHEMATICAL OPERATIONS, BITWISE OPERATORS

```
#arithmetic operation
#1. addition
resultarray=np.add(arr_EEE,arr_EGR)
print(resultarray)
```

OUTPUT:

```
[192 190 186 182 174 183 170 178 171 175]
```

```
#2. subtraction
resultarray=np.subtract(arr_physics,arr_EDS)
print(resultarray)
```

OUTPUT:

```
[ -7  -4  -6  -7  -9  -3 -10 -24   9  -1]
```

```
#3. multiplication
resultarray=np.multiply(arr_physics,arr_EDS)
print(resultarray)
```

OUTPUT:

```
[8360 8645 8455 8544 6622 8740 6375 5785 8170 7656]
```

```
#4. division
resultarray=np.divide(arr_physics,arr_EDS)
print(resultarray)
```

OUTPUT:

```
[0.92631579 0.95789474 0.93684211 0.92708333 0.89534884
0.96842105
 0.88235294 0.73033708 1.10465116 0.98863636]
```

```
#5. mod
resultarray=np.mod(arr_physics,arr_EDS)
print(resultarray)
```

OUTPUT:

```
[88 91 89 89 77 92 75 65  9 87]
```

```
#6 dot product
resultarray=np.dot(arr_physics,arr_EDS)
print(resultarray)
```

OUTPUT:

```
77352
```

```
#7. transpose physics
resultarray=np.transpose(arr_physics)
print("transpose Physics:\n", resultarray)


 # transpose eds
```

```
resultarray=np.transpose(arr_EDS)
print("transpose EDS:\n", resultarray)
```

OUTPUT:

```
transpose Physics:
 [88 91 89 89 77 92 75 65 95 87]
transpose EDS:
 [95 95 95 96 86 95 85 89 86 88]
```

```
# statistical operatios
#1. standad deviation
resultarray=np.std(arr_physics)
print(resultarray)
```

OUTPUT:

```
8.908422980528035
```

```
#2. variance
resultarray=np.var(arr_physics)
print(resultarray)
```

OUTPUT:

```
79.36000000000001
```

```
#3. mean
resultarray=np.mean(arr_physics)
print(resultarray)
```

OUTPUT:

```
84.8
```

```
#4 median
resultarray=np.median(arr_physics)
print(resultarray)
```

OUTPUT:

```
88.5
```

```
#5 average
resultarray=np.average(arr_physics)
print(resultarray)
```

OUTPUT:

```
84.8
```

```
#6 max value
resultarray=np.amax(arr_physics)
print(resultarray)
```

OUTPUT:

95

```
#7 min value
resultarray=np.amin(arr_physics)
print(resultarray)
```

OUTPUT:

65

```
#MATHEMATICAL

#addition
arr1=np.array([physics])
arr2=np.array([sic])
resultarray= arr1+arr2
print("using additionr: \n", resultarray)

#substaction
arr1=np.array([physics])
arr2=np.array([sic])
resultarray= arr1-arr2
```

```
print("using substaction: \n", resultarray)

#multiplication
arr1=np.array([physics])
arr2=np.array([sic])
resultarray= arr1*arr2
print("using multiplication: \n", resultarray)

#division
arr1=np.array([physics])
arr2=np.array([sic])
resultarray= arr1/arr2
print("using division: \n", resultarray)

#mod
arr1=np.array([physics])
arr2=np.array([sic])
resultarray= arr1%arr2
print("using mod: \n", resultarray)
```

output:

```
using additionr:
 [[179 183 180 182 164 181 164 139 184 172]]
using substaction:
```

```
 [[ -3  -1  -2  -4 -10   3 -14  -9   6   2]]
using multiplication:
 [[8008 8372 8099 8277 6699 8188 6675 4810 8455 7395]]
using division:
 [[0.96703297 0.98913043 0.97802198 0.95698925 0.88505747
1.03370787
  0.84269663 0.87837838 1.06741573 1.02352941]]
using mod:
 [[88 91 89 89 77  3 75 65  6  2]]
```

```
#bitwise


#and
arr1=np.array([arr1])
arr2=np.array([arr2])
c=np.bitwise_and(arr1,arr2)
print("for bitwise AND: \n", c)
```

```
#or
arr1=np.array([arr1])
arr2=np.array([arr2])
c=np.bitwise_or(arr1,arr2)
print("for bitwise OR: \n", c)

#xor
arr1=np.array([arr1])
arr2=np.array([arr2])
c=np.bitwise_xor(arr1,arr2)
print("for bitwise XOR: \n", c)


#left shift

arr1=np.array([arr1])
arr2=np.array([arr2])
c=np.left_shift(arr1,arr2)
print("for left shift: \n", c)

#right shift
arr1=np.array([arr1])
arr2=np.array([arr2])
c=np.right_shift(arr1,arr2)
print("for right_shift: \n", c)
```

output:

```
for bitwise AND:
 [[88 91 89 64 68 92 65 65 86 80]
for bitwise OR:
 [ 95  95  95 121  95  95  95  89  95  95]
for bitwise XOR:
 [ 7  4  6 57 27  3 30 24  9 15]
for left shift:
 [0 0 0 0 0 0 0 0 0 0]
```

```
for right_shift:
 [0 0 0 0 0 0 0 0 0 0]
```

```
# 5. Copying and Viewing arrays

#  Copy
import copy
arr1=np.array([physics])
print("copying: \n")
print(arr1)
arr2=copy.copy(arr1)



print(arr2)


#view
arr1=np.array([sic])
arr2=arr1.view()
print("\n viewing:\n")
print(arr1)
print(arr2)
```

output:

copying:

[[88 91 89 89 77 92 75 65 95 87]]
[[88 91 89 89 77 92 75 65 95 87]]

viewing:

[[91 92 91 93 87 89 89 74 89 85]]
[[91 92 91 93 87 89 89 74 89 85]]

#searching

```
array1=np.array([1,2,3,4,5])
np.searchsorted(array1,4,side="left")

output:
3
```

```
#6. data stacking
arr1=np.array(np.arange(1,5).reshape(2,2))
print(arr1)

arr2=np.array(np.arange(100,104).reshape(2,2))
print("1.\n", arr2)




array=np.stack([arr1,arr2],axis=0)
print("2.\n",array)


array=np.stack([arr1,arr2],axis=1)
print("3.\n",array)
```

output:

```
[[1 2]
 [3 4]]
1.
 [[100 101]
 [102 103]]
2.
 [[[ 1  2]
  [ 3  4]]

 [[100 101]
  [102 103]]]
3.
 [[[ 1  2]
  [100 101]]

 [[ 3  4]
  [102 103]]]
```

```
#counting
arr1=np.array([1,2,3,4,0,5,6])
print("count:\n", np.count_nonzero(arr1))
print(np.nonzero(arr1))
print("size:\n", arr1.size)
```

```
output:
count:
 6
(array([0, 1, 2, 3, 5, 6]),)
size:
 7
```

```
#sorting

arr1=np.array([[1,2,12,4,5],[10,20,120,40,50],[100,200,1200
,400,500]])
print(arr1)
```

```
np.sort(arr1,axis=0)
```

```
np.sort(arr1,axis=1)
```

```
output:
```

```
output:
```

```
[[   1    2   12    4    5]
 [  10   20  120   40   50]
 [ 100  200 1200  400  500]]
array([[   1,    2,    4,    5,   12],
       [  10,   20,   40,   50,  120],
       [ 100,  200,  400,  500, 1200]])
```