NAME – Akshit Dhake

ROLL NO. – 322(C2)

PRN – 202201040177

EDS PRACTICAL 3

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

1. Perform all matrix operations

2. Horizontal and vertical stacking of Numpy Arrays

3. Custom sequence generation

4. Arithmetic and Statistical Operations, Mathematical Operations, Bitwise Operators

5. Copying and viewing arrays

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

A	B	C	D
Roll no.	German marks	Science marks	Maths marks
1	67	98	87
2	45	99	34
3	34	44	90
4	43	30	57
5	78	67	88

CODE :

```
import numpy as np
array = np.loadtxt('markss.csv', delimiter=',' ,skiprows=1)
print(array)
math_marks =[]
science_marks=[]
german_marks=[]
for i in array:
  math_marks.append(int(i[3]))
  science_marks.append(int(i[2]))
  german_marks.append(int(i[1]))

#converting list into array

arr_mm=np.array(math_marks)
arr_sm=np.array(science_marks)
arr_gm=np.array(german_marks)

#displaying the array

print("GERMAN MARKS: ",arr_gm)
print("* SCIENCE SCORES: ",arr_sm)
print("* MATHS SCORES: ",arr_mm)

# Addition

total_marks = arr_mm + arr_sm + arr_gm
print("1.Addition :",total_marks)

# Subtraction

math_minus_german = arr_mm - arr_gm
print("2.Subtraction :",math_minus_german)
```

```
# Multiplication

science_times_2 = arr_sm * 2
print("3.Multiplication :",science_times_2)

# Division

german_divided_by_math = arr_gm / arr_mm
print("4.Division :",german_divided_by_math)

# Transpose

german_transposed = np.transpose(arr_gm)
print("5.Transpose :",german_transposed)


# Horizontal stacking

horizontal_stack = np.hstack((arr_mm, arr_sm, arr_gm))
print("6.Horizontal stacking :",horizontal_stack)

# Vertical stacking

vertical_stack = np.vstack((arr_mm, arr_sm, arr_gm))
print("7.Vertical stacking :",vertical_stack)

# Generate sequence of science score indices 0 to 4 along with values

indices = np.arange(len(arr_sm))

# Access data using the generated indices

for i in indices:
  print("8.Science score at index", i, ":", arr_sm[i])


# Copying arrays

math_marks_copy = arr_mm.copy()
print("9.Copying arrays :",math_marks_copy)


# Viewing arrays
```

```
science_marks_view = arr_sm.view()
print("10.Viewing arrays :",science_marks_view)

# Data Stacking

data_stack = np.stack((arr_mm, arr_sm, arr_gm), axis=1)
print("11.Data Stacking :",data_stack)

# Searching

index_of_88 = np.where(arr_mm == 88)
print("12.Searching :",index_of_88)

# Sorting
sorted_math_marks = np.sort(arr_mm)
print("13.Sorting :",sorted_math_marks)
# Counting
count_67 = np.count_nonzero(arr_gm == 67)
print("14.Counting :",count_67)
# Broadcasting
broadcasted_sum = arr_mm + 10
print("15.Broadcasting :",broadcasted_sum)
```

Output:

```
[[ 1. 67. 98. 87.]
 [ 2. 45. 99. 34.]
 [ 3. 34. 44. 90.]
 [ 4. 43. 30. 57.]
 [ 5. 78. 67. 88.]]
GERMAN MARKS:  [67 45 34 43 78]
* SCIENCE SCORES:  [98 99 44 30 67]
* MATHS SCORES:  [87 34 90 57 88]
1.Addition : [252 178 168 130 233]
2.Subtraction : [ 20 -11  56  14  10]
3.Multiplication : [196 198  88  60 134]
4.Division : [0.77011494 1.32352941 0.37777778 0.75438596 0.88636364]
5.Transpose : [67 45 34 43 78]
6.Horizontal stacking : [87 34 90 57 88 98 99 44 30 67 67 45 34 43 78]
7.Vertical stacking : [[87 34 90 57 88]
 [98 99 44 30 67]
 [67 45 34 43 78]]
8.Science score at index 0 : 98
8.Science score at index 1 : 99
```

```
8.Science score at index 2 : 44
8.Science score at index 3 : 30
8.Science score at index 4 : 67
9.Copying arrays : [87 34 90 57 88]
10.Viewing arrays : [98 99 44 30 67]
11.Data Stacking : [[87 98 67]
 [34 99 45]
 [90 44 34]
 [57 30 43]
 [88 67 78]]
12.Searching : (array([4]),)
13.Sorting : [34 57 87 88 90]
14.Counting : 1
15.Broadcasting : [ 97  44 100  67  98]
```