NAME – KAMLESH PAWAR

ROLL NO. –379

PRN –202201070138

“ 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**

CODE :

```
import numpy as np

# Read the dataset into a NumPy array
sales_data = np.array([
    [10, 15, 12],
    [5, 9, 7],
    [13, 6, 10]
])

# Perform matrix operations
transpose = np.transpose(sales_data)
matrix_product = np.dot(sales_data, transpose)
inverse = np.linalg.inv(sales_data)

# Display the results
print("Transpose:\n", transpose)
print("Matrix Product:\n", matrix_product)
print("Inverse:\n", inverse)

# Horizontal and vertical stacking of arrays
horizontal_stack = np.hstack((sales_data, sales_data))
print(horizontal_stack)
```

```
vertical_stack = np.vstack((sales_data, sales_data))
print(vertical_stack)

# Custom sequence generation
custom_sequence = np.arange(1, 6) * 2
print(custom_sequence)

# Arithmetic and Statistical Operations, Mathematical Operations,
Bitwise Operators
sum_all = np.sum(sales_data)
print(sum_all)
sum_axis0 = np.sum(sales_data, axis=0)
print(sum_axis0)
mean = np.mean(sales_data)
print(mean)
max_value = np.max(sales_data)
print(max_value)
min_value = np.min(sales_data)
print(min_value)
sqrt = np.sqrt(sales_data)
print(sqrt)
bitwise_and = np.bitwise_and(sales_data, 5)
print(bitwise_and)

# Copying and viewing arrays
copied_array = sales_data.copy()
print(copied_array)
view_array = sales_data.view()
print(view_array)

# Data Stacking, Searching, Sorting, Counting, Broadcasting
stacked_data = np.stack((sales_data, sales_data))
print(stacked_data)
search_index = np.where(sales_data == 10)
print(search_index)
sorted_data = np.sort(sales_data)
print(sorted_data)
count_value_10 = np.count_nonzero(sales_data == 10)
print(count_value_10)
broadcasted_sum = sales_data + 5
print(broadcasted_sum)
```

OUTPUT :

```
Transpose:
 [[10  5 13]
 [15  9  6]
 [12  7 10]]
Matrix Product:
```

```
 [[469 269 340]
 [269 155 189]
 [340 189 305]]
Inverse:
 [[ 0.94117647 -1.52941176 -0.05882353]
 [ 0.80392157 -1.09803922 -0.19607843]
 [-1.70588235  2.64705882  0.29411765]]
[[10 15 12 10 15 12]
 [ 5  9  7  5  9  7]
 [13  6 10 13  6 10]]
[[10 15 12]
 [ 5  9  7]
 [13  6 10]
 [10 15 12]
 [ 5  9  7]
 [13  6 10]]
[ 2  4  6  8 10]
87
[28 30 29]
9.666666666666666
15
5
[[3.16227766 3.87298335 3.46410162]
 [2.23606798 3.         2.64575131]
 [3.60555128 2.44948974 3.16227766]]
[[0 5 4]
 [5 1 5]
 [5 4 0]]
[[10 15 12]
 [ 5  9  7]
 [13  6 10]]
[[10 15 12]
 [ 5  9  7]
 [13  6 10]]
[[[10 15 12]
  [ 5  9  7]
  [13  6 10]]

 [[10 15 12]
  [ 5  9  7]
  [13  6 10]]]
(array([0, 2]), array([0, 2]))
[[10 12 15]
 [ 5  7  9]
 [ 6 10 13]]
2
[[15 20 17]
 [10 14 12]
 [18 11 15]]
```