EXPERIMENT - 2

NUMPY OPERATIONS

AIM:
To study and practice various NumPy operations including **array creation, attributes, indexing, slicing, broadcasting, arithmetic, statistical operations, concatenation, reshaping, sorting and splitting** using a case study.

Case Study: A company manager wants to analyze bike sales over 4 weeks in January from their three branches (branch a, b and c). Using NumPy, extract meaningful insights from the raw sales data.

PREREQUISITES & REQUIREMENTS:

1. Computer with Python Installed
2. Jupytor Notebook
3. Knowledge on Python & Numpy Library

Step 1: Install and Import the NumPy Library using commands:

pip install numpy

In [203…

```
# importing NumPy
import numpy as np
```

Step 2: Create arrays for three branches a, b and c for four week bike sales

In [204…

```
# [Week1, Week2, Week3, Week4] for branch a, b and c

branch_a = np.array([150, 200, 180, 200])
branch_b = np.array([160, 210, 160, 230])
branch_c = np.array([170, 220, 200, 240])
```

In [205…

```
# Created arrays
branch_a, branch_b, branch_c
```

Out[205…

```
(array([150, 200, 180, 200]),
 array([160, 210, 160, 230]),
 array([170, 220, 200, 240]))
```

Step 3: Check the properties of the arrays with different attributes

In [206…

```
# Dimension of the array
branch_a.ndim
```

Out[206…

```
1
```

In [207…

```
# Shape of the array
branch_b.shape
```

Out[207…

```
(4,)
```

In [208…

```
# Size of the array
branch_c.size
```

Out[208…

```
4
```

In [209…

```
# Data Type of the array
branch_a.dtype
```

Out[209…

```
dtype('int64')
```

In [210…

```
# Space used by each element in array
branch_a.itemsize, branch_b.itemsize, branch_c.itemsize
```

Out[210…

```
(8, 8, 8)
```

Step 4: Retretive Sales data by Indexing & Slicing the arrays

In [211…

```
# Created arrays
branch_a, branch_b, branch_c
```

Out[211…

```
(array([150, 200, 180, 200]),
 array([160, 210, 160, 230]),
 array([170, 220, 200, 240]))
```

In [212…

```
# Index start with 0
# Get the branch_c week 1 sales data
branch_c[0]
```

Out[212…

```
np.int64(170)
```

In [213…

```
# Get the unique values in the branch_b
np.unique(branch_b)
```

Out[213…

```
array([160, 210, 230])
```

In [214…

```
# Get branch_a week 1,2 sales data
branch_a[0:2]
```

Out[214…

```
array([150, 200])
```

In [215…

```
# Get branch_b last two weeks sales data
branch_b[2:]
```

Out[215…

```
array([160, 230])
```

In [216…

```
# Get branch_c last three weeks sales data (negative slicing)
branch_c[-3:]
```

Out[216…

```
array([220, 200, 240])
```

Step 5: Assign new values to the branch sales (Broadcasting)

In [217…

```
branch_a
```

Out[217…

```
array([150, 200, 180, 200])
```

In [218…

```
# Making a new copy for branhc_a
branch_a_new = branch_a
```

In [219…

```
branch_a_new
```

Out[219…

```
array([150, 200, 180, 200])
```

In [220…

```
# Updating week 1 sales data
branch_a_new[0] = 156
branch_a_new
```

Out[220…

```
array([156, 200, 180, 200])
```

In [221…

```
# Adding week 5,6 sales data to the branch_a_new array
branch_a_new = np.append(branch_a_new, [280,190])
branch_a_new
```

Out[221…

```
array([156, 200, 180, 200, 280, 190])
```

In [222…

```
# Updating week 5, 6 sales data using slicing
branch_a_new[4:6] = [300, 170]
branch_a_new
```

Out[222…

```
array([156, 200, 180, 200, 300, 170])
```

Step 6: Arithmetic Operations on the sales data

In [223…

```
branch_c
```

Out[223…

```
array([170, 220, 200, 240])
```

In [224…

```
# Adding +2 sales to all 4 weeks
branch_c + 2
```

```
Out[224… array([172, 222, 202, 242])
```

```
In [225… # Subtracting 100 sales from the week 4 data with indexing in branch_c
         branch_c[3] - 10
```

```
Out[225… np.int64(230)
```

```
In [226… # Double the sales in week 1 with indexing in branch_c
         branch_c[0] * 2
```

```
Out[226… np.int64(340)
```

```
In [227… # Half sales in week 2 & 3
         branch_c[1:3] / 2
```

```
Out[227… array([110., 100.])
```

Step 7: Statistical Operations on the sales data

```
In [228… branch_a, branch_b, branch_c
```

```
Out[228… (array([156, 200, 180, 200]),
          array([160, 210, 160, 230]),
          array([170, 220, 200, 240]))
```

```
In [229… # What are the Total sales of branch_a
         np.sum(branch_a)
```

```
Out[229… np.int64(736)
```

```
In [230… # What are the Maximum and Minumun sales of the branch_a
         np.max(branch_a), np.min(branch_a)
```

```
Out[230… (np.int64(200), np.int64(156))
```

```
In [231… # What are the average (mean) sales of the each branch
         print(np.mean(branch_a), np.mean(branch_b), np.mean(branch_c))
```

```
184.0 190.0 207.5
```

Step 8: Combine all branches sales data (Concatenating)

```
In [232… branch_a, branch_b, branch_c
```

```
Out[232… (array([156, 200, 180, 200]),
          array([160, 210, 160, 230]),
          array([170, 220, 200, 240]))
```

```
In [233… # While concatenating, dimension must be same
         branch_a.ndim, branch_b.ndim, branch_c.ndim
```

```
Out[233… (1, 1, 1)
```

```
In [234… # Combining all 3 branches sales
         all_branches_sales = np.concatenate([branch_a, branch_b, branch_c])
```

```
In [235… all_branches_sales
```

```
Out[235… array([156, 200, 180, 200, 160, 210, 160, 230, 170, 220, 200, 240])
```

```
In [236… # Total sales of all branches
         np.sum(all_branches_sales)
```

```
Out[236… np.int64(2326)
```

Step 9: Reshape the all_branches_sales into 3x4 matrix

```
In [237… all_branches_sales.ndim, all_branches_sales.shape
```

```
Out[237… (1, (12,))
```

```
In [238… all_branches_sales = all_branches_sales.reshape(3,4)
         all_branches_sales
```

```
Out[238… array([[156, 200, 180, 200],
                [160, 210, 160, 230],
                [170, 220, 200, 240]])
```

```
In [239… # Check the shape and dimension
         all_branches_sales.ndim, all_branches_sales.shape
```

```
Out[239… (2, (3, 4))
```

Step 9: Sort and Split the all_branches_sales array

```
In [240… all_branches_sales
```

```
Out[240… array([[156, 200, 180, 200],
                [160, 210, 160, 230],
                [170, 220, 200, 240]])
```

```
In [241… # Indexing 2-D array
         all_branches_sales[0]
```

```
Out[241… array([156, 200, 180, 200])
```

```
In [242… # Sort Row wise sales
         np.sort(all_branches_sales, axis=1)
```

```
Out[242… array([[156, 180, 200, 200],
                [160, 160, 210, 230],
                [170, 200, 220, 240]])
```

```
In [243… # Sort Column wise sales
         np.sort(all_branches_sales, axis=0)
```

```
Out[243… array([[156, 200, 160, 200],
                [160, 210, 180, 230],
                [170, 220, 200, 240]])
```

```
In [244… # Splitting the all_branches_sales array into 3 different arrays
         # Splitting based on rows (axis=0)
         split_a, split_b, split_c = np.split(all_branches_sales, [1,2], axis=0)
```

```
In [245… split_a
```

```
Out[245… array([[156, 200, 180, 200]])
```

```
In [246… split_b
```

```
Out[246… array([[160, 210, 160, 230]])
```

```
In [247… split_c
```

```
Out[247… array([[170, 220, 200, 240]])
```

```
In [248… all_branches_sales
```

```
Out[248… array([[156, 200, 180, 200],
                [160, 210, 160, 230],
                [170, 220, 200, 240]])
```

```
In [249… # What are the Week 2 sales of all three branches?

         # Slice only column 2
         week_2_sales = all_branches_sales[:3, 1:2]
```

```
In [250… print(week_2_sales)
```

```
[[200]
 [210]
 [220]]
```

RESULT:

By using the NumPy library, various operations on arrays such as creation, attribute access, indexing, slicing, broadcasting, arithmetic operations, statistical computations, concatenation, sorting, and splitting were successfully implemented and demonstrated using a bike sales data.