Experiment-2

2 a) Using numpy model perfrom the following operation

1) Demonstrate array aggrigation function

2) Demonstarte the vectorized operations

3) Demonstarte the map , filter , reduce ,lambda functions with data frame.

2 B)Using pandas model perform the following operation

1) Aggrigation and grouping

2) Demonstrate map,filter,reduce & lambda function with dataframe

1) Demonstrate array aggrigation function

In [1]:
```
# Sum() :- Use to find the sum of the given array.
# max() :- It returns the maximum values among the elements of given array.
# min() :- It returns the minimum values among the elements of given array.
# mean() :- It returns the Mean(Averge) of the input array.
# std () :- It shows the standard deviation of array.
# median () :- It returns the median value of array.
```

In [2]:
```
import numpy as np
```

In [3]:
```
 a=np.array([20,26,73,84,34,97,45,72])
 b=np.array([45,23,66,63,44,85,23,41])
 print("set of a array is :",a)
 print("set of b array is :",b)
```

```
set of a array is : [20 26 73 84 34 97 45 72]
set of b array is : [45 23 66 63 44 85 23 41]
```

In [4]:
```
#1) Sum ()
```

In [5]:
```
 s=(a.sum())
 print("Sum of a array is :",s)
 # OR
 #print("sum of a array is :",np.sum(a))
```

```
Sum of a array is : 451
```

In [6]:
```
##2) Max ()
```

In [7]:
```
 m=(b.max())
 # print("Maximum values of array is :",m)
 # OR
 print("Maximum values of array is :",np.max(b))
```

```
Maximum values of array is : 85
```

In [8]:
```
 ##3) Min ()
```

In [9]:
```
 #mi=(a.min())
 # print("Minimum value of array is :-",mi)
 # OR
 print("Minimum value of array is :",np.min(a))
```

```
Minimum value of array is : 20
```

In [10]:
```
##4) Mean ()
```

In [11]:
```
# me=(a.mean())
 # print("Averge value of array is :",me)
 # OR
print("Averge value of array is :",np.mean(a))
```

```
Averge value of array is : 56.375
```

In [12]:
```
##5) std ()
```

In [13]:
```
# st=(b.std())
 # print("Standerd diviation of array is :",st)
 # OR
print("Standerd diviation of array is :",np.std(b))
```

```
Standerd diviation of array is : 20.116846174288852
```

In [14]:
```
##6) median ()
```

In [15]:
```
 # md=(a.std())
 # print("Median of a array is :",md)
 # OR
 print("Median of a array is :",np.median(a))
```

```
Median of a array is : 58.5
```

2) Demonstarte the vectorized operations

In [16]:
```
import numpy as np
 # creating arrays
a1= np.array([10, 20, 30])
b1= np.array([1, 2, 3])
```

In [17]:
```
 print("set of a1 array is :",a1)
 print("set of b1 array is :",b1)
```

```
set of a1 array is : [10 20 30]
set of b1 array is : [1 2 3]
```

In [18]:

```
 # Arithmetic Operations
 # Addition
 # Subtraction
 # Multiplication
 # Division
print("Addition:", a + b)
print("Subtraction:", a - b)
print("Multiplication:", a * b)
print("Division:", a / b)
```

```
Addition: [ 65  49 139 147  78 182  68 113]
Subtraction: [-25   3   7  21 -10  12  22  31]
Multiplication: [ 900  598 4818 5292 1496 8245 1035 2952]
Division: [0.44444444 1.13043478 1.10606061 1.33333333 0.77272727 1.14117647
 1.95652174 1.75609756]
```

In [19]:

```
 #  Mathematical Functions
 print("Square root of a:", np.sqrt(a))
```

```
Square root of a: [4.47213595 5.09901951 8.54400375 9.16515139 5.83095189 9.84885
78
 6.70820393 8.48528137]
```

In [20]:

```
print("Sum of a:", np.sum(a))
print("Max of b:", np.max(b))
print("Mean of a:", np.mean(a))
```

```
Sum of a: 451
Max of b: 85
Mean of a: 56.375
```

3) Demonstarte the map , filter , reduce ,lambda functions with data frame.

In [21]:

```
import pandas as pd
from functools import reduce
data = {
 'Name': ['Alice', 'Bob', 'Charlie', 'David'],
 'Age': [25, 30, 35, 40],
 'Salary': [50000, 60000, 70000, 80000]
}
df = pd.DataFrame(data)
df
```

Out[21]:

	Name	Age	Salary
0	Alice	25	50000
1	Bob	30	60000
2	Charlie	35	70000
3	David	40	80000

In [22]:

```
##1) map ()
```

In [23]:

```
def add(x):
 return x + 2000
```

```
Salary_List = df['Salary'].map(add)
print("Added Salaries:\n", Salary_List)
```

```
Added Salaries:
 0    52000
1    62000
2    72000
3    82000
Name: Salary, dtype: int64
```

In [24]:
```
##2) filter ()
```

In [25]:
```
def get(age):
 if age > 30:
     return True
l1=(df['Age'])
res=list(filter(get,l1))
print("Grater then 30 years Age :",res)
```

```
Grater then 30 years Age : [35, 40]
```

In [26]:
```
##3)reduce ()
```

In [27]:
```
def add(x, y):
    return x + y
total_salary = reduce(add, df['Salary'])
print("Total Salary:", total_salary)
```

```
Total Salary: 260000
```

In [28]:
```
def max_value(x, y):
    return x if x > y else y
max_age = reduce(max_value, df['Age'])
print("Maximum Age:", max_age)
```

```
Maximum Age: 40
```

In [29]:
```
##4) lambda ()
```

In [30]:
```
# Age grater then 30 years using lambda
old= df[df['Age'].apply(lambda x: x > 30)]
print(old)
```

```
      Name  Age  Salary
2  Charlie   35   70000
3    David   40   80000
```

In [31]:
```
# Sum of salary using Lambda
total= reduce(lambda x, y: x + y, df['Salary'])
print("Total Salary:", total)
```

```
Total Salary: 260000
```

2 B)Using pandas model perform the following operation

1) Aggrigation and grouping

In [32]:

```
#* sum()
#* min()
#* max()
#* mean()
#* describe()
#* count()
#* std()
#* sum()
```

In [33]:

```
import pandas as pd

data = {
    'Department': ['HR', 'IT', 'HR', 'IT', 'Finance'],
    'Age': [25, 30, 45, 35, 40],
    'Salary': [50000, 60000, 52000, 58000, 70000]
}
df = pd.DataFrame(data)
print(df)
```

```
  Department  Age  Salary
0         HR   25   50000
1         IT   30   60000
2         HR   45   52000
3         IT   35   58000
4    Finance   40   70000
```

In [34]:

```
#Sum()
```

In [35]:

```
df['Age'].sum()
```

Out[35]:

```
np.int64(175)
```

In [36]:

```
## Min()
```

In [37]:

```
df['Age'].min()
```

Out[37]:

```
np.int64(25)
```

In [38]:

```
# max
```

In [39]:

```
df['Age'].max()
```

Out[39]:

```
np.int64(45)
```

In [40]:

```
# mean
```

In [41]:

```
df['Age'].mean()
```

Out[41]:

```
np.float64(35.0)
```

In [42]:

```
# standard Diviation
```

In [43]:

```
df['Age'].std()
```

Out[43]:

```
np.float64(7.905694150420948)
```

In [44]:

```
# Describe
```

In [45]:

```
df['Age']. describe()
```

Out[45]:

```
count     5.000000
mean     35.000000
std       7.905694
min      25.000000
25%      30.000000
50%      35.000000
75%      40.000000
max      45.000000
Name: Age, dtype: float64
```

In [46]:

```
# Count
```

In [47]:

```
df['Age']. count()
```

Out[47]:

```
np.int64(5)
```

◆ Grouping Examples

1. Sum of salaries by department

In [48]:

```
print(df.groupby('Department')['Salary'].sum())
```

```
Department
Finance     70000
HR         102000
IT         118000
Name: Salary, dtype: int64
```

In [49]:

```
#2. Average salary by department
```

In [50]:

```
print(df.groupby('Department')['Salary'].mean())
```

```
Department
Finance    70000.0
HR         51000.0
IT         59000.0
Name: Salary, dtype: float64
```

In [51]:

```
#3. Count of employees per departmen
```

In [52]:

```
print(df.groupby('Department')['Age'].count())
```

```
Department
Finance    1
HR         2
IT         2
Name: Age, dtype: int64
```

In [53]:

```
# 4. Multiple aggregations together
```

In [54]:

```
print(df.groupby('Department').agg({
    'Age': ['mean', 'max'],
    'Salary': ['sum', 'mean', 'min']
}))
```

```
               Age      Salary
              mean max     sum     mean    min
Department
Finance       40.0  40   70000  70000.0  70000
HR            35.0  45  102000  51000.0  50000
IT            32.5  35  118000  59000.0  58000
```

2) Demonstrate map,filter,reduce & lambda function with dataframe

In [55]:

```
# Map
```

In [56]:

```
df['Salary_upper'] = df['Salary'].map(lambda x: x+1000)
df
```

Out[56]:

	Department	Age	Salary	Salary_upper
0	HR	25	50000	51000
1	IT	30	60000	61000
2	HR	45	52000	53000
3	IT	35	58000	59000
4	Finance	40	70000	71000

In [57]:

```
# filter
```

In [58]:

```
filtered_df = df[df['Salary'] > 50000]
filtered_df
```

Out[58]:

	Department	Age	Salary	Salary_upper
1	IT	30	60000	61000
2	HR	45	52000	53000
3	IT	35	58000	59000
4	Finance	40	70000	71000

In [59]:

```
# reduce
from functools import reduce
```

In [60]:

```
m=reduce(lambda x,y: x+y,df['Salary'])
print(m)
```

```
290000
```

In [ ]: