3) Using Numpy module , Perform the follwing operations .

a) Demonstarte Array aggregations functions.

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.

In [1]:

```
import numpy as np
```

In [2]:

```
a=np.array([20,26,73,84,34,97,45,72])
a
```

Out[2]:

```
array([20, 26, 73, 84, 34, 97, 45, 72])
```

1) Sum () :-

In [4]:

```
s=(a.sum())
print("Sum of array is :",s)
```

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

2) Max () :-

In [5]:

```
m=(a.max())
print("Maximum values of array is :",m)
```

```
Maximum values of array is : 97
```

3) Min () :-

```
In [6]: mi=(a.min())
        print("Minimum value of array is :-",mi)
```

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

4) Mean () :-

```
In [7]: me=(a.mean())
        print("Averge value of array is :",me)
```

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

b) Demonstarate vectorized operations.

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

        # creating arrays

        a = np.array([10, 20, 30])
        b = np.array([1, 2, 3])
```

```
In [3]: a,b
```

```
Out[3]: (array([10, 20, 30]), array([1, 2, 3]))
```

```
In [ ]:
```

```
In [4]: # Arithmetic Operations
         # Addition
         # Subtraction
         # Multiplication
         # Division

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

```
Addition: [11 22 33]
Subtraction: [ 9 18 27]
Multiplication: [10 40 90]
Division: [10. 10. 10.]
```

```
In [ ]:
```

In [38]:

```
#  Mathematical Functions

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

```
Square root of a: [3.16227766 4.47213595 5.47722558]
```

In [39]:

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

```
Sum of a: 60
Max of b: 3
Mean of a: 20.0
```

In [ ]:

In [ ]:

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

In [2]:

```
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[2]:

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

1) map () :-

In [3]:

```
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 [ ]:

In [ ]:

2) filter () :-

In [4]:

```
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 [ ]:

3) reduce () :-

In [5]:

```


def add(x, y):
    return x + y

total_salary = reduce(add, df['Salary'])
print("Total Salary:", total_salary)
```

```
Total Salary: 260000
```

In [30]:

```


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
```

4) lambda () :-

In [24]:

```
# Age grater then 30 years using lambda

older_than_30 = df[df['Age'].apply(lambda x: x > 30)]
print(older_than_30)
```

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

In [28]:

```
# Sum of salary using Lambda

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

```
Total Salary: 260000
```

In [ ]:

```
```