05/08/2025

Experiment 2:

Study of Python Basic Libraries such as Statistics, Math, Numpy and Scipy.

#1. Implementation of Numpy

```
import numpy as np
arr=np.array([1,0,4])
print("Array with Rank1:\n",arr)
arr=np.array([[4,3,2],[5,6,7]])
print("Array with Rank2:\n",arr)
arr=np.array((9,8,7))
print("\n Array created using" "passed tuple:\n",arr)
```

Output :

```
Array with Rank1:
 [1 0 4]
Array with Rank2:
 [[4 3 2]
 [5 6 7]]
 Array created using passed tuple:
 [9 8 7]
```

#2. Slicing of an Array

```
import numpy as np
arr= np.array([[2,4,-9,0],
                    [2,1,3.7,8],
                    [5,6,7,8],
                    [1.7,4,6,3]])
print("Initial Array")
print(arr)
sliced_arr=arr[:2,::2]
print("Array with first 2 rows and"
        "alternate columns(0 and 2) :\n",sliced_arr)
Index_arr=arr[[1,1,0,3],
                   [3,2,1,0]]
print("\n Elements at indices(1,3),"
"(1,2),(0,1),(3,0):\n",Index_arr)
```

05/08/2025

Output :

```
Initial Array
[[ 2.   4.  -9.   0. ]
 [ 2.   1.   3.7  8. ]
 [ 5.   6.   7.   8. ]
 [ 1.7  4.   6.   3. ]]
Array with first 2 rows and alternate columns(0 and 2) :
 [[ 2.  -9. ]
 [ 2.   3.7]]

 Elements at indices(1,3),(1,2),(0,1),(3,0):
 [8.  3.7 4.  1.7]
```

#3.Basic Operations on Array

```
a=np.array([[9,8],[5,6]])
b=np.array([[4,3],[1,8]])
print("adding 1 to every element",a+1)
print("\n Sybtracting 2 from every element",b-2)
print("\n Sum of all array elements:",a.sum())
print("\nArray sum:\n",a+b)
```

Output :

```
adding 1 to every element [[10  9]
 [ 6  7]]

 Sybtracting 2 from every element [[ 2  1]
 [-1  6]]

 Sum of all array elements: 28

Array sum:
 [[13 11]
 [ 6 14]]
```

#4.String methods

```
s="MACHINELEARNING"
print(s)
print(s[1])
print(s[12])
print(s[-14])
print(s[-4])
print(s[10])
print(s[-10])
print(s[1:4])
print(s[-4:-1])
print(s[-2:-7:-2])
```

Output :

```
MACHINELEARNING
A
I
A
N
R
N
ACH
NIN
NNA
```

#5. Flattening and ravel

```
Import numpy as np
x=np.array([[4,3,2,0],[3,5,7,3],[9,6,3,1]])
a=x.flatten()
print(x)
print(a)
a[0]=99
print(a)
a2=x.ravel()
a2[0]=98
print(x)
print(a2)
```

Output :

```
[[4 3 2 0]
 [3 5 7 3]
 [9 6 3 1]]
[4 3 2 0 3 5 7 3 9 6 3 1]
[99  3  2  0  3  5  7  3  9  6  3  1]
[[98  3  2  0]
 [ 3  5  7  3]
 [ 9  6  3  1]]
[98  3  2  0  3  5  7  3  9  6  3  1]
```

#6. Array Creation methods

```
import numpy as np
a=np.array([4,1,7])
a
```

Out: array([4, 1, 7])

```
a.dtype
```

Out: dtype('int32')

```
b=np.array([1.8,6,7.9,4])
b.dtype
```

```
Out: dtype('float64')
```

```
np.zeros((2,5))
```

```
Out: array([[0., 0., 0., 0., 0.],
       [0., 0., 0., 0., 0.]])
```

```
np.ones((2,5,3),dtype=np.int16)
```

```
Out: array([[[1, 1, 1],
        [1, 1, 1],
        [1, 1, 1],
        [1, 1, 1],
        [1, 1, 1]],

       [[1, 1, 1],
        [1, 1, 1],
        [1, 1, 1],
        [1, 1, 1],
        [1, 1, 1]]], dtype=int16)
```

```
np.empty((2,3))
```

```
Out: array([[2.12199579e-314, 6.36598737e-314, 1.06099790e-313],
       [1.48539705e-313, 1.90979621e-313, 2.33419537e-313]])
```

```
np.arange(10,50,5)
```

```
Out:  array([10, 15, 20, 25, 30, 35, 40, 45])
```

```
np.arange(1,6,0.5)
```

```
Out: array([1. , 1.5, 2. , 2.5, 3. , 3.5, 4. , 4.5, 5. , 5.5])
```

#7. Printing arrays

```
a=np.arange(5)
print(a)
b=np.arange(12).reshape(4,3)
print(b)
c=np.arange(20).reshape(2,2,5)
print(c)
```

Output :

```
[0 1 2 3 4]
[[ 0  1  2]
 [ 3  4  5]
```

05/08/2025

```
 [ 6  7  8]
 [ 9 10 11]]
[[[ 0  1  2  3  4]
  [ 5  6  7  8  9]]

 [[10 11 12 13 14]
  [15 16 17 18 19]]]
```

#8. Multiplication of 2 arrays

```
a=np.array([[9,7],[5,4]])
b=np.array([[2,5],[2,4]])
print(a*b)
print(a@b)
```

Output:

```
[[18 35]
 [10 16]]
[[32 73]
 [18 41]]
```

#9. Basic operations

```
a=np.array([12,14,16,18,20])
b=np.arange(5)
print('a=',a)
print('b=',b)
c=a-b
print('c=',c)
print('b**2=',b**2)
print('a<16=',a<16)
```

Output:

```
a= [12 14 16 18 20]
b= [0 1 2 3 4]
c= [12 13 14 15 16]
b**2= [ 0  1  4  9 16]
a<16= [ True  True False False False]
```

# Implementation of scipy

```
from scipy import special as sp
a=sp.exp10(3)
b=sp.exp2(8)
c=sp.sindg(90)
print("a=",a)
print("b=",b)
print("c=",c)
```

Output:

```
a= 1000.0
b= 256.0
c= 1.0
```

#Linear Algebra

```
from scipy import linalg
import numpy as np
a=np.array([[1,3,9],[8,2,1],[2,0,1]])
print("a=",a)
b=linalg.inv(a)
print("b=",b)
```

Output:

```
a= [[1 3 9]
 [8 2 1]
 [2 0 1]]
b= [[-0.03846154  0.05769231  0.28846154]
 [ 0.11538462  0.32692308 -1.36538462]
 [ 0.07692308 -0.11538462  0.42307692]]
```

05/08/2025

```
# Derivatives

from scipy.misc import derivative
def my_function(x):
    return x**2+2*x+3
a=derivative(func= my_function, x0=2)
print(a)
```

Output:

```
6.0
```

```
# Determinant

from scipy import linalg
import numpy as np
a = np.array([[3, 4, 5],
              [1, 1, 2],
              [5, 0, 8]])
print("a =\n", a)
b = linalg.det(a)
print("Determinant b =", b)
c = linalg.inv(a)
print("Inverse c =\n", c)
```

Output:

```
a =
 [[3 4 5]
 [1 1 2]
 [5 0 8]]
Determinant b = 7.000000000000004
Inverse c =
 [[ 1.14285714 -4.57142857  0.42857143]
 [ 0.28571429 -0.14285714 -0.14285714]
 [-0.71428571  2.85714286 -0.14285714]]
```

05/08/2025

```
#Eigen values and eigen vectors

from scipy import linalg
import numpy as np
arr=np.array([[1,3],[9,8]])
eg_val,eg_vect=linalg.eig(arr)
print("eigen values:",eg_val)
print("eigen vectors:\n",eg_vect)
```

Output:

```
eigen values: [-1.76498204+0.j 10.76498204+0.j]
eigen vectors:
 [[-0.73532226 -0.29367359]
 [ 0.67771762 -0.95590576]]
```

```
# Integration

from scipy import integrate
import numpy as np
from math import sqrt
a= lambda x:x**3
b=integrate.quad(a,0,1)
print("inte\n",b)
```

Output:

```
inte
 (0.25, 2.7755575615628914e-15)
```

```
#Double integral

from scipy import integrate
import numpy as np
from math import sqrt
f = lambda x, y: 32 * x * y
p = lambda x: 0
q = lambda y: sqrt(1 - 2 * y**2)
```

```
integration = integrate.dblquad(f, 0, 2 / 4, p, q)
print("integration\n",integration)
```

Output:

```
integration
 (1.7999999999999998, 6.661275421441794e-14)
```